Charon Version 3.4 Supervisor's Guide

"Charon (Kar'on). In Greek mythology, the ferryman, a son of Erubus, who transported the souls of the dead (whose bodies had been buried) over the river Styx to the lower world.

His fee was an obolus or other coin, and this was placed for him in the mouth of the dead at the time of burial."

"Charon (Kar'on). In Networking mythology, the gateway, a son of CUTE, who transports the mail messages and data files of the office dead (whose text has been relegated to queues) over the IPX-TCP/IP river to the other world.

His fee is a PC/XT/AT or other MS-DOS based processor with ethernet, and this is placed for him in the back closets and shelves of empty, sterile computer rooms at the time of installation."

August 14, 1991

Brad K. Clements
bkc@omnigate.clarkson.edu

Charon Version 3.4 Supervisor's Guide

Disclaimer

Clarkson University provides this program to educational and commercial institutions free of charge with the following limitations:

- This program is not for sale, and may not be combined with programs that are for sale.
- There is no warranty or claim of fitness or reliability. Although the author has made every effort to remove bugs, neither he nor Clarkson University shall be held liable for any loss of data, down time or other direct or indirect damage or claims caused by this program.
- You may NOT charge a distribution fee for giving this program to others.
- Please report bugs and other comments to:

cutcp-bugs@omnigate.clarkson.edu

- There is an internet email based list which is dedicated to the discussion of all CUTCP programs, including Charon and CUTE. You can subscribe to this discussion list by sending email to

cutcp-request@omnigate.clarkson.edu

Chapter 1. Introduction

Are you running a mixed networking environment consisting of Novell Netware LANS and TCP/IP based minicomputers/mainframes?

Do you need to transfer electronic mail messages and data between your two environments?

If so, then Charon is the program for you. With Charon you can:

- Transfer Electronic Mail between Novell Servers and TCP/IP mainframes
- Synchronize the time on your Novell Servers to other systems via RDate.

Charon transfers electronic mail messages between Novell Servers and TCP/IP based SMTP mailers. Charon can synchronize the server time on your Novell Servers with each other, and to a Unix host running RDate. Up to eight Novell Servers can be supported by each Charon gateway.

1.1 Installation Requirements

You will need the following to install and run Charon.

- An IBM PC/XT/AT or compatible with at least 640K of memory. This machine must be dedicated to the task of running Charon.
- A network interface card and Netware shells for the above gateway machine.
- An Ethernet card and its corresponding Packet Driver (see Appendix for a list of Ethernet cards which have Packet Drivers).

  Note: You may use a single ethernet card to run both Novell IPX and Charon. This is accomplished using:

  - Packet Driver and BYU's IPX for Packet Driver

    or
  - ODI Driver and the ODI to Packet Converter

- An on-site central SMTP mailer capable of forwarding messages to 'The Internet'.
- An on-site Domain Name Server (or /etc/hosts.txt files).
- An understanding of Novell Print Queues, and how to use Pconsole.

Chapter 2. Installation

This section of the manual will give you detailed instructions on how to install Charon.

Installation Overview

A. Prepare your Novell Servers
B. Generate Domain Name information
C. Configure your central SMTP mailer
D. Configure Charon
E. Prepare a boot disk

2.1 Prepare Your Novell Servers

Novell Server preparation involves several steps, most of which can be accomplished using the PConsole and Syscon Novell utilities. You will have to log in as Supervisor in order to perform the installation process.

Preparation Overview

A. Create a Print Server account.
B. Create a Print Server group.
C. Add the Print Server account to the Print Server group.
D. Create Queues for incoming and outgoing mail.
E. Grant the Print Server account access to the mail queues.
F. Grant the Print Server group access to the SYS:MAIL directories.
G. Choose a ’Master’ Novell Server.
H. Optionally create a log file and list file directory on the Master Server.

A Print Server Group is created to facilitate the maintenance of access permissions on the Novell Server. Most of the Novell maintenance utilities only work with groups and users. But because Charon runs as a Print Server, a method is required to grant the Print Server account access to various directories as if that account were a regular user account. This is accomplished by making the Print Server account security equivalent to the Print Server Group, and then manipulating the Print Server Group using the various Novell utilities. Although you will create a Print Server called ’Charon’, and a User Group called ’Charon’, Novell sees these as two separate bindery objects.

Each Charon gateway can service up to eight Novell Servers. Each collection of Novell Servers, serviced by a single gateway, will hereafter be referred to as a Cluster. If you have more than eight Novell Servers, you will require more than one Charon gateway.

You may wish to group your Novell Servers by function, by geography or by adjacency to each other.

Carry out the following steps on each Novell Server that will be serviced by the Charon gateway. Check off the steps as you go.

[ ] *Use PCONSOLE to Create a Print Server Account*

A. Select **Print Server Information** from the Main menu of PConsole.
B. Press **<INSERT>** to add a Print Server name to the list of existing Print Servers.
C. Enter the name **CHARON**
D. Exit **PCONSOLE** by pressing **<ALT-F10>.**

[ ] *Use SYSCON to Create a Print Server Group*

A. Select **Group Information** from the Main menu of Syscon.
B. Press **<INSERT>** to add a group to the list of existing groups.
C. Enter the name **CHARON.**
D. Exit **Syscon** by pressing **<ALT-F10>.**

[ ] *Use ADDQGRP to add the Print Server Account to the Group*

The ADDQGRP program is provided with the Charon package. Type the following command to add the Print Server Account to the Print Server Group.

ADDQGRP CHARON CHARON

Notice! If you subsequently run bindfix on this Novell Server, you will have to perform this step again to re-add the print server account to the Charon group.

[ ] *Use PCONSOLE to create Queues for Incoming and Outgoing Mail*

A. Select **Print Queue Information** from the Main menu.
B. Press **<INSERT>** to create a Print Queue.
C. Enter the name **MAILQUEUE**

[ ] *Grant Charon Access to the MailQueue*

A. Select **MAILQUEUE** from the list of available Print queues.
B. Select **Queue Operators**
C. Press **<INSERT>** to add a Print Queue Operator to the Queue.
D. Enter the name **Charon**

This enters the Group Charon as a Print Queue Operator for this Queue. You created this group using Syscon in a previous step.

E. Return to the **Print Queue Information** screen by pressing **<ESCAPE>**
F. Select **Queue Servers.**
G. Press **<INSERT>** to add a Print Server to the Queue.
H. Enter the name **Charon**

This enters the Print Server Account Charon as a Print Server for this Queue. You created this account in the first step of the installation procedure.

I. Return to the **Print Queue Information** screen by pressing **<ESCAPE>.**
J. Select **Queue Users.**
K. Press **<INSERT>** to add an account as a Print Queue User.
L. Enter the name **Charon**

This enters the Print Server Group Charon as a User of this Queue.

M. Modify the existing list of Queue Users as desired.

The default Queue Users list includes the group **EVERYONE.** If you do not want Everyone to have access to the SMTP gateway, remove this group from the list and enter a different group that corresponds to those users who you will grant SMTP access to.

If you will be using the Charon gateway for all electronic mail transmission (Pmail's **Use Gateway Always** option), you should enter whatever groups you desire to have any electronic mail access, not just SMTP mail access.

N. Exit **PCONSOLE** by pressing **<ALT-F10>**

[ ] *Grant the Print Server Group Access to SYS:MAIL*

The Charon gateway requires Create and Write access to the SYS:MAIL directory on your Novell Server. Without this access, the gateway will be unable to deliver mail to user mail accounts.

A. For NW386 servers, issue the following DOS command:

GRANT C FOR SYS:MAIL TO GROUP CHARON

B. For 2.15 servers, issue the following DOS command:

GRANT C W FOR SYS:MAIL TO GROUP CHARON

[ ] *Choose a Master Novell Server*

Each Charon gateway requires one Novell Server to act as a ’Master’ server. The Master server is used as the repository for mailing lists, log files and incoming SMTP mail messages.

If your gateway will be servicing more than one Novell server, you should select one of the servers as your Master. Choose the Novell server which is logically the ’closest’ to your Charon gateway, and the most capable in terms of disk space and performance. Future versions of Charon will make heavy use of the master server’s disk space. It is expected that when (if) MHS support is added to, Charon will eliminate the WORKQUEUE method, and move to a file based method of mail handling. Additionally, as Charon becomes more resistant to file server failures, it will come to rely heavily on the master server. Choose your most reliable server.

If your gateway is servicing only one Novell server, your choice is easy.

If you will be running more than one Charon Gateway, you will have more than one Master Novell Server: carry out the following tasks on each of those Master Novell Servers.

[ ] *Use PCONSOLE to create a Workqueue on each Master Novell Server*

A. Select **Print Queue Information** from the Main menu.
B. Press **<INSERT>** to create a Print Queue.
C. Enter the name **WORKQUEUE**

[ ] *Grant Charon Access to the WorkQueue*

A. Select **WORKQUEUE** from the list of available Print queues.
B. Select **Queue Operators**
C. Press **<INSERT>** to add a Print Queue Operator to the Queue.
D. Enter the name **Charon**

This enters the Group Charon as a Print Queue Operator for this Queue.

E. Return to the **Print Queue Information** screen by pressing **<ESCAPE>**
F. Select **Queue Servers.**
G. Press **<INSERT>** to add a Print Server to the Queue.
H. Enter the name **Charon**

This enters the Print Server Account Charon as a Print Server for this Queue.

I. Return to the **Print Queue Information** screen by pressing **<ESCAPE>.**
J. Select **Queue Users.** .item Press **<INSERT>** to add an account as a Print Queue User.
K. Enter the name **Charon**

This enters the Print Server Group Charon as a User of this Queue.

L. Remove the group **EVERYONE** from the list of Queue Users by scrolling to the group EVERYONE and pressing **<DELETE>**

[ ] *Create a List File Directory on your Master Novell Server*

Create a directory on your Master Novell server.

mkdir SYS:PUBLIC\LISTS

In this directory you will place membership lists for each system/cluster wide mailing list.

[ ] *Create a Log File Directory*

Each Charon gateway requires one or more directories in which to place (optional) log files. You may place these log files on a local disk (local to the gateway machine) or on any of your Novell Servers (or several of them).

It is suggested that you create a log file directory on your Master Novell Server. You will need to grant Create, Read and Write access to this directory.

Example:

mkdir SYS:USR\CHARON

grant ALL FOR SYS:USR\CHARON TO GROUP CHARON

2.2 Generate Domain Name Information

To accomplish this step you will need to have a detailed understanding of how the Domain Name Service works, and how your local DNS server is configured.

If you do not have this information, I suggest that you simply hand over these instructions to your local DNS administrator and let them worry about it.

The objective of this section is to configure your DNS system to allow off-site SMTP mail to be sent to your central SMTP mailer first, then to your Charon gateway. Additionally, the following DNS configuration allows off-site nodes to determine where to send mail destined for your Novell Servers.

DNS Configuration Overview

A. Create an Internet address and Name for the Charon gateway
B. Create an Internet name for each Novell Server
C. Create an MX entry for each Novell Server

[ ] *Create an IP Address for the Charon gateway*

Using the appropriate local procedure, create an internet address and name for each of your Charon gateways. Typically you will request an internet name, and your DNS administrator will return the assigned internet address to you. Additionally, you will probably also receive a netmask value and an internet gateway address. You will use this information later when configuring Charon.

I encourage you to be creative when choosing the gateway internet name. Perhaps something reflecting the gatewaying/defender/arbiter nature of Charon.'s mythological background.

[ ] *Create an Internet name for each Novell Server*

Each Novell server requires an internet name. You may wish to use the Novell server name with the appropriate local internet sub-domain name added. In any case, create/obtain an appropriate internet name for each Novell server. You will use this information throughout the installation process.

[ ] *Create an MX entry for each Novell Server*

Each Novell server requires a DNS MX entry to equate it’s name with an internet address. Because the Novell servers themselves do not have an internet address, you will use the MX feature of the DNS to equate mail only access to an internet address. The equated address should be the internet address of your central mailer. Optionally you can set it to the internet address of your Charon gateway. This isn’t recommended because off-site TCP/IP traffic will probably fair better going to a large processor (your central SMTP mailer), rather than to a small PC with limited capacity.

At our site, we used both internet addresses, with a preference for our central mailer first, and if that is not available SMTP mail will be sent to our Charon gateway as a second choice.

Example:

Our central mailer is omnigate.clarkson.edu.

Our Charon gateway is romulus.erc.clarkson.edu.

We have two Novell Servers with the following internet names:

- darius.adm.clarkson.edu
- draco.erc.clarkson.edu

We entered the following information into our DNS configuration files.

```
draco.erc.clarkson.edu     IN    MX    0    omnigate.clarkson.edu
draco.erc.clarkson.edu     IN    MX    10   romulus.erc.clarkson.edu
darius.adm.clarkson.edu    IN    MX    0    omnigate.clarkson.edu
darius.adm.clarkson.edu    IN    MX    10   romulus.erc.clarkson.edu
```

Warning! You must **never** send mail to the Charon gateway address itself. In the above example, mail sent to user@romulus.erc.clarkson.edu would cause an infinite loop. This is because Charon knows that romulus is not a Novell server, it then sends the message to the central SMTP agent. The agent knows that romulus receives mail for itself, then sends it back...

2.3 Configure your Central SMTP Mailer

This portion of the installation process is another one that can be passed off to a local ’expert’. Unless you are the local expert!

Your central SMTP mailer may need to be configured to pass off mail destined for your Novell

Servers to your Charon gateway.

If you used the DNS configuration sample given above, you may not need to do anything to your central mailer’s configuration.

However, at our location we had to configure our central SMTP mailer (running MMDF) as follows. (Your mileage will vary).

In our domain table, we added the following:

```
draco.erc.clarkson.edu:        draco.erc.clarkson.edu romulus.erc.clarkson.edu
darius.adm.clarkson.edu:    darius.adm.clarkson.edu romulus.erc.clarkson.edu
```

This causes MMDF to rewrite mail destined for user@draco.erc.clarkson.edu into the form @romulus.erc.clarkson.edu:user@draco.erc.clarkson.edu and then deliver it to our Charon gateway, Romulus.

If you are using Unix sendmail, consult Appendix B for installation suggestions submitted by Mr. John Wobus of Syracuse University.

2.4 Configure Charon

Charon uses two configuration files during its operation.

- **CONFIG.TEL**

  gives Charon information about the networking environment, including Charon.’s internet address (IP address), network mask, subnet information and remote host address information.

- **CHARON.DAT**

  gives Charon information about the Novell Servers that will be accessed, the names of the associated queues, mailing lists and other management information.

Both of these configuration files are completely read and processed before Charon ’logs in’ to the specified Novell Servers as a Print Server. The advantage to this technique is that the configuration files can be stored on a Novell Server and initially accessed by a Novell User account. The disadvantage to this technique is that some of the contents of both files are stored in memory during the operation of the program. Fortunately neither file is particularly large.

Keep in mind that any change to either data file requires a subsequent restart of the gateway before that change will take effect.

2.4.1 Create the CONFIG.TEL file.

The Charon distribution includes a sample **config.tel** that will assist you in creating your own configuration file. You should simply edit the supplied file and replace the appropriate values with those specific to your site. The sample file contains several sections, each section represents different network related information required by the gateway. You should be able to obtain the required information from your site's TCP/IP network administrator.

Use a text editor to alter the supplied **config.tel** file according to the following specifications.

[ ] *Enter the Gateway Network Information*

- In section 2.2, you obtained an IP address for your Charon gateway machine. Enter this IP address in the **myip** variable section of the file.

- Enter the netmask appropriate for the IP subnet that your Charon gateway will be using in the **netmask** variable section of the file.

  Example: A subnet that uses 6 bits of subnetting would have a subnet mask of **255.255.252.0**

[ ] *Enter the Gateway Hardware Information*

- If you are using a packet driver for network communication, set the **hardware** variable to

hardware=packet.

Otherwise select the appropriate interface card from the list presented in the file.

- If you are not using a packet driver for the hardware interface, set the following variables to match your current ethernet hardware settings: **interrupt, address, ioaddr.**

[ ] *Enter the Domain Name Resolution Information*

There is only one variable that should be set in this section of the configuration file. You should set the **domainslist** variable to contain your site's domain name.

Example: My gateway is **Romulus.erc.clarkson.edu** my site's domain name is **clarkson.edu.** I set the domainslist variable to

domainslist="clarkson.edu"

[ ] *Enter the Mailer Agent Information*

In section 2.3 you configured your central SMTP mailer to pass mail files to the Charon gateway. This section of the configuration file contains information about your central SMTP mailer (referred to as an 'agent').

- Enter the 'short name' of your agent in the **name** variable of this section.
- Enter the 'long name' of your agent in the **host** variable of this section.
- Enter the IP Address of your agent in the **hostip** variable of this section.

[ ] *Enter the Domain Name Server Information*

You should enter the IP address of at least two Domain Name Servers. In the sample configuration file included with the Charon package, the mailer agent happens to be the primary domain name server for our site. You do NOT have to make the primary name server and the mailer agent be the same host.

You should enter the 'short name', 'long name' and the IP address of your primary and secondary domain name server in the appropriate variables of this section of the file.

[ ] *Enter the IP Gateway Information*

In this section of the file, you should enter the 'short name', 'long name' and IP address of the IP gateway for the subnet that the Charon gateway machine will be using.

[ ] *Enter Other Host Information*

In this section you may enter additional host information as desired.

It is recommended that you include an entry for the gateway machine itself.

2.4.2 Create the Charon.Dat file

The Charon package includes a sample Charon.dat file which contains all possible options that can be specified. I recommend that you create a new Charon.dat file using a text editor of your choice. Use the sample charon.dat file as a reference only.

The Charon.dat file syntax follows a free form treelike hierarchy. Specifically that means that:

A. Commands are not case sensitive
B. Commands are not positionally dependant
C. Some commands produce lists internally, and therefore may be repeated
D. Only the sequence in which commands are encountered is important
E. Quoted strings are handled specially

The sample Charon.dat file includes comments next to each command explaining its purpose, the arguments it takes, and the sequencing that is required.

For the most part, the sample file represents command levels with indentation.

Example:

```
Server
        mailqueue
                poll
                        10
```

In the previous example, the **poll** command has one argument, the poll time in seconds. This command is only valid after a **mailqueue** command, which in turn is only valid after a **server** command. The indentation is not required, and serves only as a reminder that the command structure is a treelike structure with levels.

Any text following a ';' on a single line represents a comment and is ignored. Quoted text may contain special escape sequences which can be used to insert control characters into the charon.dat file.

Arguments do not need to be quoted unless they contain spaces, even though the sample charon.dat file quotes many arguments to make them more noticeable.

The following Conventions will be used in this section to describe the format of the charon.dat file:

A. Command arguments will be outlined with the ’"’ symbol.

Example: "Argument"

B. Commands which may be repeated will be followed by the symbol **(1+).** Do not include the (1+) in your charon.dat file.

C. Commands which are not required will be followed by the symbol **(opt).**

2.4.3 Top Level Commands

The following are ’top level’ commands. They are listed in the sequence in which they should appear in the Charon.dat file. Each command has zero or more sub commands which are described later in this chapter. Frequently, information entered for a given command will be referenced in following commands. Pay special attention to file names, queue names, etc to ensure that the spelling is consistent through the configuration file.

I. **MYNAME** "gateway internet name"

This required command specifies the internet name of the gateway. In section 2.2A you obtained an internet name for the Charon gateway. Insert that name here.

II. **SCREENSAVE** "screensaver delay time" (opt)

This optional parameter specifies the delay time in seconds before the screen saver activates. The default delay time is 300 seconds (5 minutes). If you do not want the screen saver to activate, enter a value of 0. When the screen saver is activated, the screen is cleared and a crawling worm appears. The screen is restored when any key is pressed.

III. **RCONSOLE** (opt)

This optional command enables the Remote Console facility within Charon. (See Chapter 4.0). If you do not include this command, the remote console facility will not be enabled.

IV. **SERVER** "Novell Server Name" (1+)

This required command specifies the name of a Novell server that should be serviced by the Charon gateway. This command has several sub commands that specify login userid, password, etc. Specify this command once for each server that requires service by this gateway. See section 2.4.4 for server sub-command configuration information.

V. **MAILER**

This required command demarks the beginning of the mailer section of the configuration file. See section 2.4.5 for mailer sub-command configuration information.

VI. **TIMESYNC** (opt)

This optional command demarks the beginning of the timesync section of the configuration file. If you do not wish to timesync your Novell file servers, you need not include this command (nor its corresponding sub-commands). See section 2.4.6 for timesync sub-command configuration information.

VII. **ALIASES**

This required command demarks the beginning of the aliases section of the configuration file. At least one alias (postmaster) must be defined for Charon to operate correctly. See section 2.4.7 for aliases sub-command configuration information.

VIII. **LOGFILES** (opt)

This optional command demarks the beginning of the logfiles section of the configuration file. If you do not wish to create log files, you need not enter this command (nor its corresponding subcommands) in the configuration file. See section 2.4.8 for logfiles sub-command configuration information.

IX. **LOG** (opt)

This optional command demarks the beginning of the log section of the configuration file. If you do not wish to write log entries, you need not enter this command (nor its corresponding subcommands) in the configuration file. See section 2.4.9 for log sub-command configuration information. If you do include this command, you MUST also include the LOGFILES command.

2.4.4 Server Subcommands

This section outlines the commands which may be used following a server command. You should repeat the server command for each server that will be serviced by Charon.. After each occurrence of the server command, enter the server subcommands that correspond to the previously entered server.

I. **USERID** "Print Server Userid"

This command specifies the userid that will be used by Charon when it logs into the specified Novell server. If you followed the installation instructions outlined in section 2.1, you should enter the name "charon".

Keep in mind that this userid is a Print Server name, and is created using PConsole. You may also have a regular user account by this name (created by Syscon), or a user group of the same name (also created by Syscon). They are all distinct bindery objects!

II. **PASSWORD** "Print Server Password"

This command specifies the password that should be used to log in as the specified user on the specified Novell server. This will typically be the null password (empty quotes ""). Unlike version 3.1 of Charon, this version (3.4) is capable of logging into a NW 386 file server which has encrypted passwords enabled.

The password for the print server is set using PConsole.

III. **MAILQUEUE** "Mail Queue Name"

This required command specifies the name of the mail queue on the indicated file server. If you followed the instructions outlined in section 2.1, you should enter the name "mailqueue".

This queue is used by Pmail (and other mailers) to deliver outgoing messages to Charon. You may use the PConsole utility to control which users have access to the gateway. You must grant Charon full access to this queue (see section 2.1).

This command has one subcommand.

1. **POLL** "Queue Poll Time" (opt)

This optional Mailqueue subcommand specifies the poll frequency in seconds. The default is 10 seconds.

2.4.5 Mailer Subcommands

This section outlines the mailer subcommands which control how mail is handled by Charon. Some of these commands are optional. Some of the commands accept more than one argument.

I. **AGENT** "mailer agent name"

This required command specifies the name of your central SMTP mailer. In section 2.4.1 you entered the short and long name of your central SMTP mailer in the CONFIG.TEL file. You should enter the short name of your SMTP mailer here. Be sure that the spelling is exactly the same.

Charon will initiate SMTP connections to this host to deliver all outgoing SMTP mail.

II. **LISTS** "Master Server Name" "Directory Path" (opt)

This optional parameter specifies where mailing list information can be found. If you will not be using Charon to expand any local mailing lists, you need not specify this command.

The **"Master Server Name"** argument specifies the name of the Novell server where the mailing lists are stored. This name must match one of the values specified after a server command.

The **"Directory Path"** argument specifies the directory on the server where the lists are stored. You should not include a trailing ’\’ in the path statement. Any ’\’ that is entered should be escaped with a leading ’\’ because the ’\’ character has a special meaning in the configuration file. You may store your lists on a local hard disk (within the gateway machine), simply specify a path that references the disk drive letter (eg: C:).

III. **SMTPOUT** "Master Server Name" "Work Queue Name" "Poll Frequency" (opt)

This optional parameter specifies the queue location where Charon should enqueue outgoing SMTP messages. If you do not specify this command, the server and queue specified in the SMTPIN command is used. You are encouraged to make use of a separate outgoing queue to make Charon’s job easier.

Additionally, using this command you may specify a longer poll frequency for the queue, allowing outgoing SMTP messages to be batched up and transmitted to the SMTP agent in a single TCP session.

The **"Master Server Name"** argument specifies the name of the server where the outgoing messages should be queued.

The **"Work Queue Name"** argument specifies the name of the outgoing queue on the master server. If you followed the installation instructions outlined in section 2.1, you should enter the name "workqueue".

The **"Poll Frequency"** argument specifies the poll frequency in seconds for this queue. There is no default value, you must specify one.

IV. **SMTPIN** "Master Server Name" "Mail Queue Name"

This required command specifies the location where incoming SMTP messages should be queued.

The **"Master Server Name"** is the name of your master Novell server.

The **"Mail Queue Name"** is the name of the mail queue on your master Novell server. If you followed the installation instructions outlined in section 2.1, you should enter the name "mailqueue".

V. **LISTCYCLES** "Cycle Count" (opt)

This optional command specifies the number of addressees of a mailing list which should be processed per 'cycle'. This command allows you to tailor the loading characteristics of Charon with respect to mailing list expansion.

The **"Cycle Count"** argument specifies the number of addressees which should be processed in a single 'cycle'. The default is 5. Mail lists with addressees primarily 'off Novell' represent very little load on Charon, and therefore the cycle count could be quite large without severely impacting the operation of the gateway. Local mailing lists, on the other hand, which have a large number of Novell recipients, do represent a noticeable load (no more so than regular mail delivery).

This command works in conjunction with the LISTDELAY command. You must be careful when determining the appropriate values to use for each command.

For example, a mailing list with 100 addressees could take up to four minutes to process if the cycle count is set to 1 and the listdelay is set to 1. A more reasonable value would be a cycle count of 10 or 20 and a listdelay of 1.

VI. **LISTDELAY** "List Delay Time" (opt)

This optional command specifies the number of seconds which Charon should suspend processing of a mailing list between cycles.

The **"List Delay Time"** argument specifies the delay time in seconds. The default is 1 second.

While Charon delays processing a list, no other mail messages are handled. Therefore this argument should be kept small.

Note: Both ListCycles and ListDelay commands may be eliminated in future versions in favor of a more dynamic load balancing scheme.

VII. **DEBUG** (opt)

This optional command instructs Charon to open an SMTP debugging window which may be used to check the progress of SMTP transmissions through the Charon gateway. The default is to NOT open such a window. I suggest that you do include this command in your configuration file, and verify the operation of the gateway for the first few days. Afterwards you may remove this command, thereby freeing up memory for other uses.

VIII. **MAX_SMTPDS** "Number of SMTPDs" (opt)

This optional command specifies the maximum number of SMTPDs that may be operating at any given time. At least one SMTPD must be operating at all times. An SMTPD process is used to receive incoming SMTP mail via TCP/IP. Normally when one incoming SMTP connection is established, a new SMTPD will be created to receive additional messages (if any). However, it is possible to run out of memory if too many SMTPDs are spawned.

The **"Number of SMTPDS"** argument specifies the maximum number of SMTPDs that may be operating at any given time. The default is 1. I recommend that you keep this value less than 4.

IX. **NOBROADCAST** (opt)

This optional command inhibits Charon from sending broadcast messages to users when they receive new mail. If you are using Pmail, you may use the PConfig program to inhibit broadcast messages on a user by user basis instead of turning it off for all users.

X. **RETURNLINES** "Number of Lines to Return" (opt)

This optional command specifies the number of lines of a rejected message that should be returned to the sender (or the postmaster) when Charon is unable to deliver a message.

The **"Number of Lines to Return"** argument is the number of lines to return (integer). The default is 10. If you want all of the original message to be returned, specify a value of 0.

XI. **RETURNTO** "Return Rejected Mail To" (opt)

The **"Return Rejected Mail To"** argument specifies the recipient of rejected mail. It must be one of the following values.

- **sender**
- **postmaster**
- **both**

The default value is **sender.**

If you specify **postmaster** the rejected message will be sent only to the postmaster, and not to the sender of the mail message.

• **TIMEOUT** "Timeout Duration" (opt)

This optional command specifies how long Charon should attempt to deliver or receive a message via SMTP before considering the connection invalid and aborting the transfer.

The **"Timeout Duration"** argument is the maximum duration of an SMTP transfer (per message) in seconds. The default is 600 seconds (10 minutes).

Note that batched outgoing SMTP messages (or incoming messages) reset the timeout counter for each message, not for the total duration of the SMTP session. Therefore it is possible to have an SMTP session last longer than the timeout duration, so long as the session is actively transmitting messages.

2.4.6 TimeSync Commands

This section describes the configuration commands that relate to the timesync feature of Charon. Charon has the capability of synchronizing the server time on attached Novell servers. The source of the ’official’ time may be either another Novell server or a remote Unix system (or other system) that supports the RDate command.

In a typical mixed TCP/IP Novell environment, one or more Unix systems will be synchronized to universal coordinated time (UTC) via the Network Time Protocol (NTP). Any one of these Unix systems may in turn become the time source for Charon. via the RDate protocol.

Charon must be a file server console operator to be able to set the time on your file servers. If you followed the installation instructions outlined in section 2.1, use **Syscon** to add the Group **Charon** as a file server console operator on each of your Novell Servers. (Supervisor Options/File Server Console Operators).

If you do not want Charon to synchronize the time on your Novell file servers, skip this section.

The following commands are allowed under the timesync section of the configuration file.

I. **MASTER** "Master Time Server Name"

This required command designates the name of the master time server.

The **"Master Time Server Name"** designates the name of either a Novell server (specified in the Server section), or a Unix system (specified in the config.tel file). If you specify a Unix system, the RDATE protocol will be used to obtain the current time from the master time server. Additionally, when using a Unix system as the master time server, enter the short name of the Unix system for this command. Be sure to include the short and long name of the Unix system along with its Internet Address in the config.tel file under the ’Other Hosts’ Section.

II. **SLAVE** "Slave Novell Server Name" (1+)

This required command specifies the name(s) of your slave Novell Servers.

The **"Slave Novell Server Name"** argument specifies the name of the slave server (previously specified in the Server section) which will be synchronized to the master time server.

A slave may NOT also be a master. Repeat this command for each Novell server which should be synchronized.

III. **POLL** "Synchronization Poll Frequency" (opt)

This optional command specifies the frequency with which Charon should attempt to synchronize the slave Novell servers.

The **"Synchronization Poll Frequency"** specifies the frequency in seconds. The default is 43200 seconds (12 hours).

You should avoid making this value too small.

2.4.7 Alias Commands

This section describes the commands that may be used in the alias section of the configuration file.

There are three types of alias subcommands that may be used:

- **node**
- **user**
- **list**

I. **NODE** "Novell Server Internet Name" "Novell Server Name" (1+)

This required command equates a Novell server name with its Internet Name. You must specify this alias once for each Novell server defined in the servers section.

The **"Novell Server Internet Name"** argument specifies the Internet Name of the Novell server. In section 2.2 of this manual you obtained an Internet Name for each of your Novell servers. You should enter the fully qualified Internet Name as the first argument of this command.

The **"Novell Server Name"** argument specifies the Novell server name (as entered in the Servers section) that corresponds to the specified Internet Name.

There must be a one-to-one mapping of Internet Names and Novell server names. Do not enter a Novell server name more than once.

II. **USER** "username" "user@node" (1+)

III. **USER** "username@Server" "user@node" (1+)

This command has two forms as shown above. The first form takes a simple username and equates it to a specified user at a given node. The second form takes a username at a given Novell server, and equates it to a specified user at a second node.

The **"Username"** or **"Username@Server"** argument specifies the name that should be aliased.

The **"user@node"** argument specifies the result of the alias. A result must include both the user name and the node name.

Charon resolves aliases from the specific to the general. That means that if a user name is aliased twice, once as "fred", and once as "fred@draco", the second form will be used.

You must enter at least one alias for "postmaster". You may, if you so desire, create an alias for postmaster at each node, possibly all being sent to the same user, or to the supervisors at each node. Your choice.

IV. **LIST** "ListName" "List File" (opt) (1+)

V. **LIST** "ListName@Server" "List File" (opt) (1+)
VI. **LIST** "ListName" ":SERVER/GROUP" (opt) (1+)
VII. **LIST** "ListName@Server" ":SERVER/GROUP" (opt) (1+)

This optional command specifies a mailing list alias.

The **"ListName"** or **"ListName@Server"** argument specifies the name of the mailing list, optionally at the specified Novell server.

The **"List File"** argument specifies the name of the list file. This file must be stored in the LISTS directory specified in the Mailer section of the configuration file (see section 2.4.5).

The format of a list file is quite simple. One address per line is allowed. Both Charon and PMail can share the same list file.

The **":SERVER/GROUP"** provides a method of mailing to a Novell user group. The ':' is required, followed by the name of the Novell server and the group name.

Example:

list "everyone@draco" ":draco/everyone"

Future versions of Charon are expected to have a much more advanced list processing capability, akin to a Bitnet ListServe.

When a user alias and a list alias have the same name, the user alias has precedence.

2.4.8 Logfiles Subcommands

This section outlines the subcommands available under the Logfiles section of the configuration file. If you will not be logging any information, you may skip this section and the log section.

Logfiles are used to record information about message transfers, errors encountered, warnings etc. Internally, Charon as several 'processes' which actually carry out the work of handling mail and file transfer. Each process can record information about its work in a log file. The logfiles concept allows for a lot of flexibility. For example, you could have all processes write to the same physical log file. You could have each process write to a different physical log file. You could even have the same process write different information to several log files.

To accomplish this, Charon breaks down the logfile information into two parts:

○ **logfiles** Where information should be stored

○ **log** What information should be stored

This section describes logfiles, where log information will be stored.

I. **FILE** "Internal LogFile Name"

This required command specifies the internal name of a log file.

The **"Internal LogFile Name"** argument is used to demark the beginning of a log file specification. This name is used in the log section to refer to 'this' log file.

This command actually begins a new level of subcommands. Therefore it must be specified before any of the following commands in the logfiles section.

II. **SERVER** "Novell Server Name" (opt)

This optional subcommand specifies the name of the Master Novell server where this log file is to be stored. If you want your logfile stored on a local disk, do not specify the SERVER command. If you do specify this command, the logfile is opened in a deferred mode. This means that it is not actually opened until Charon is logged in and a process sends log information to the file.

The **"Novell Server Name"** is the name of the Novell server (specified in the Servers section) which will contain the log file.

III. **NAME** "Physical File Name"

This required command specifies the physical name of the log file. If you did not specify a SERVER (you intend to store the log on a local disk), be sure to include the drive letter of the disk.

Charon requires full read and write access to this file. If you are storing your log files on a Novell server, you may set up the file flags to enable viewing of the log files while Charon is in operation.

A. Create an empty file that matches the Physical File Name
B. Use FLAG to mark it S R W (sharable Read Write)

C. Use Grant (as required) to enable Charon to write to the file.

You may optionally use a common sub-directory to store all the log files. Then grant Charon access to this sub-directory.

I. **MAXSIZE** "Maximum Log File Size" (opt)

This optional command specifies the maximum log file size in bytes. If this command is not specified, the log file will grow until disk space is exhausted.

II. **RECYCLE** (opt)

This optional command specifies that Charon should recycle the log file to the beginning when the maximum size has been reached. If MAXSIZE is specified, and RECYCLE has not been specified, then the log file is simply truncated when the maximum size is reached. Otherwise when Charon reaches the maximum size it begins writing log information at the beginning of the file, overwriting previous information.

III. **SEPARATOR** "Separator data" (opt)

This optional command specifies the separator string that should be written between log file data items. (see the log section for more information about data items). The default is a single space character. This string may contain any characters, including "\n" (newline). Experimentation will enable you to determine how best to store log information.

For example, if you will want to incorporate log information into a spreadsheet, you could set the separator to ",", which would produce a comma-separated list of data items which could easily be handled by a spreadsheet program.

IV. **SHOWTAGS** (opt)

This optional command causes Charon to include the data tag name before each data item. The default is to NOT include the data tag name. See the Log section for a listing of data tag names.

V. **COLUMNS** "Column Width" (opt)

This optional command causes output log data to be right justified in columns of the specified width.

The **"Column Width"** argument specifies the width of the column in characters. There is no default value.

Log data values wider than the specified width are truncated.

2.4.9 Log Subcommands

This section outlines the subcommands available under the Log section of the configuration file. If you will not be logging any information, you may skip this section and the Logfiles section. In this section of the configuration file you will bind a process name to a log file name, and specify the log data items that you want recorded to the log file.

In this version of Charon, there are four processes that may be logged.

A.	**system**	Informational and Warning Messages
B.	**smtpout**	Outgoing SMTP Mail
C.	**smtpin**	Incoming SMTP Mail
D.	**mailer**	'Local' Mail and Lists

It is important to realize that the mailer process handles ALL mail that is transferred by Charon, including incoming and outgoing SMTP messages. If you wish to account for incoming and outgoing SMTP, you should record smtpout and smtpin data in a file separate from the mailer data. Otherwise you will count SMTP mail twice.

Each process generates log data items that you may select for logging purposes. Different processes generate different log data items, however all processes generate the following log data items:

- **date** The current date
- **time** The current time
- **process** The name of the process

The 'system' process generates the following additional log data items:

- **info** Information Messages
- **warn** Warning Messages

The other three mail related processes generate the following additional log data items:

- ○ **sender_name** The sender's username
- ○ **sender_node** The sender's node name
- ○ **filesize** The size of the message in bytes
- ○ **destination_name** The recipient's username
- ○ **destination_node** The recipient's node name
- ○ **message_id** The internal message id

I. **PROCESS** "Process Name" (1+)

This required command specifies the name of a process to be logged. This name must match one of the names listed above. You may specify the same process name more than once, if you want the process data to be logged in different ways to different files.

The following commands are really subcommands of PROCESS, and therefore must follow a PROCESS statement.

A. **FILE** "Internal Log File Name"

This subcommand binds the process to a logfile.

The **"Internal Log File Name"** argument specifies the name of the log file to which data should be logged. This is the same internal log file name that you specified in the file statement under the logfiles section.

B. **ITEM** "Log Data Tag Name" (1+)

This subcommand specifies the log data which is to be recorded in the log file for this process.

The **"Log Data Tag Name"** must match one of the Log Data Tag names listed above (date, info, etc). The order in which you specify the tag values is the order in which the data is written to the log file.

Experimentation with the (logfiles/file/separator) command, the (logfiles/file/showtags) command and the (logfiles/file/columns) command will allow you to determine the most suitable format for your log files.

The sample charon.dat file uses only two physical log files. The first log file, called "system", records both information and warning messages. Normally, warning messages should be logged to a separate file from information messages to ensure that they stand out more. This could be easily accomplished by adding another (log/process "system") entry which contains only the date and warn tag names.

The "system" process is a direct copy of the "System Messages Window" data. Therefore, the text of the warning and informational messages already contain the current time. That is why the sample charon.dat file does not include the "time" tag under the "system" process.

2.5 Prepare A Boot Disk

The final step in the installation process is the creation of a suitable boot disk. This is extremely site dependent, so I will describe the minimum requirements and what we've done here.

When Charon begins operation, it needs to access two files:

- **charon.dat**
- **config.tel**

The **charon.dat** file must be in the 'current directory' when Charon begins operation. Normally the **config.tel** file is also in the current directory. However you may set a DOS environment variable to point to the location of this file.

Example:

DOS SET CONFIGTEL="C:\net\config.tel"

You must also set the DOS environment variable, **TZ** to the local time zone setting. The TZ variable has the following format:

SET TZ=ZZZ[+/-]d[d][lll]

Where **ZZZ** is the three character string that represents the current time zone, such as 'EST' or 'PST'. **[+/-]d[d]** is a required field containing an optionally signed number with one or more digits. This number is the local time zone's difference from GMT in hours. Positive numbers are West of GMT. **[lll]** is an optional three character string that represents the local time zone daylight saving time (PDT, EDT, etc).

Examples:

SET TZ=EST5EDT

SET TZ=MET-1

At Clarkson University we use a remote boot Prom to boot many of our PCs, including our gateway machine. The boot disk image causes the gateway PC to log in as a user called 'Charon'. The personal login script of this account maps the first network drive to the directory which contains the two configuration files, the program and the log files. It then performs an

exit "charon"

to begin operation of the gateway.

Warning! You must NOT use a #charon command in your login script. This will cause the login.exe program to remain resident while Charon runs. Login.exe uses too much memory and Charon will not function properly when it runs out of memory.

Your boot disk should work along these lines. You will also have to include appropriate copies of NET, IPX and a packet driver (if that's how your setup works).

Chapter 3. Running Charon

Begin the operation of Charon by 'booting' a PC using your previously prepared boot disk. Charon should clear the screen and begin writing diagnostic information in the System Messages Window. You may see some rapid screen flashing during the first few seconds of operation as various windows are opened.

At this point you will see a screen similar to that shown in figure 1 of Appendix A. You may switch between windows to view various diagnostic information. The function of each window is described below.

If you do not see the expected results on the screen, press the <ESCAPE> key twice. This should return you to the DOS prompt. In this case, Charon is unable to automatically determine where video memory is on the gateway machine.

If the gateway machine has a monochrome monitor, you may need to execute the DOS mode command before running Charon.

mode bw80

If, after entering the above command and restarting Charon, you still do not get the expected results, you may need to manually specify the video buffer address using the -v command line option.

Charon -v b000 (for monochrome displays)

Charon -v b800 (for color displays)

3.1 Handyman Notes - Troubleshooting

First and foremost, this version of Charon can not recover from a crashed Novell server. If your network temporarily becomes unusable (for a few minutes or less) Charon can usually recover. However, if you 'down' a server, use fconsole to force off the Charon connection, or one of your servers crashes, Charon will die. This is not a bug. Charon was not designed for this type of abuse (our servers stay up for months at a shot). However, I am looking into workarounds for this *limitation*. In the meantime, be aware that Charon won't start if you specify a server that's down. I should say, it starts, but the 're-attach' code doesn't seem to work. Please don't report this as a bug....

Charon is written in the C++ language, it makes heavy use of system memory, and will operate erraticly when memory is depleted. Although your gateway machine may have 640K of ram, not all of that will be usable due to various loaded drivers. If Charon operates strangely, crashes, etc, the first thing you should check is the amount of available ram. See the section on the **Task Display** for information on how to determine the available amount of ram. Version 3.4 of Charon has a low-memory detection routine that will write a diagnostic message on the system message window when free memory falls below a pre-determined value (currently 15Kbytes). Charon may fail before free memory falls below this point! Read the system.act file carefully for indications of the cause of failure.

If jobs get stuck in the MAILQUEUE or WORKQUEUE, they can be deleted using the Pconsole command. Pconsole can be very helpful in diagnosing problems with outgoing SMTP mail. The queue can be placed on hold, individual jobs can be fed through as desired, etc. One caveat, don't delete a job that is currently being serviced by Charon.

Networking problems often plague Charon users. The first test is to send an ICMP Echo Request (ping) to the gateway address from another system. Often administrators set the myip= value in the config.tel file to something other than they wanted. This results in comments of the type "mail leaves my novell system, but nothing gets delivered". If a ping works, but mail still isn't delivered to your Novell users, you should telnet to port 25 on the gateway and look for a response. The typical Unix command for this operation is:

telnet *gateway_ip_address* 25

If the gateway responds, this indicates a possible configuration problem in your SMTP agent.

If mail being sent from Novell to 'the outside world' never gets delivered, but stays in the

WORKQUEUE, enable the **debug** command under the **mailer** section of charon.dat. Watch to see if Charon is able to contact your SMTP agent. Often the mailer agent settings don't match the values in the config.tel file. If you see the message

domain name resolution failed

you **know** you've done something wrong because this version of Charon does NOT support domain name lookups.

As a last resort, you may press the **<?>** key on the gateway console and receive a dump of the current TCP control blocks. This information may be helpful to you or to the cutcp-bugs recipient.

3.2 Operating the Charon Console

Charon has several console windows which present important information of a diagnostic nature. The windows are arranged in a circular order. The following keys are recognized:

- **<+>** Show Next Window
- **<->** Show Previous Window
- **<UPARROW>** Scroll Up
- **<DOWNARROW>** Scroll Down
- **<PgUp>** Page Up
- **<PgDown>** Page Down
- **<LeftArrow>** Scroll Left
- **<RightArrow>** Scroll Right
- **<HOME>** Jump to Top of Window
- **<END>** Jump to Bottom of Window
- **<ESC>** Exit Program
- **<?>** Dump TCP socket Info

Each window displays a 'scroll indicator' in the upper left corner of the display. Arrows in the indicator section point to data off screen that may be viewed by pressing the appropriate arrow key.

Pressing the <ESC> key will cause Charon to exit. Charon will then switch back to the System Message Window and request that the operator press <ESC> a second time. This allows the operator to view closing error messages (if any).

3.2 The System Message Window

Appendix A, figure 1 and figure 2 show a typical System Message Window. This window displays information and warning messages.

All error messages will be reported in this window. You may set up a log file to record these messages (see section 2.4.9). This window is approximately 50 lines in size. This allows the operator to page back to previous messages if desired.

3.3 The Stream Status Display Window

Appendix A, figure 3 shows a typical Stream Status Display Window. This window is used to present diagnostic information about the internal SYS V streams emulation code.

Internally, Charon passes data between processes using Messages and Data Blocks. There are a fixed number of Data Blocks of various sizes. The top portion of this window displays information about the Data Blocks.

Although this information is not required for normal daily operation, if you experience problems with your gateway an indication of the number of block allocation refusals could be useful to the program designer in determining the cause of the problem.

The middle portion of this window displays information about Messages.

The bottom portion of this window displays information about Stream Queues. A symbolic representation of each queue and its link to other queues is shown on the left side of the window. The **Count** field indicates the number of Messages enqueued for service by the queue. The **Flags** field indicates if a queue **H**as a service procedure, **N**eeds service, if the queue is **B**locked, if the queue is **E**nabled for service, if the queue is **F**ull, or if the queue is **I**nhibited from being enabled for service. Each queue has a high and low water mark that is used to control the flow of data through the queue.

3.4 The Task Display Window

Appendix A, figure 4 shows a typical Task Display Window. Other than the System Message Window, this window is the most useful. This window is normally updated once every three seconds.

Charon uses a non-preemptive tasking scheme with round-robin scheduling. Each task is either runnable, or sleeping. Tasks may be signaled by receiving data, waking up from an alarm, or by being killed.

The top line of the window displays the number of tasks, the free memory that has **never** been allocated for use by the program, the up time, and the current date and time.

The second line of the window shows the load average (a very rough estimate of how hard the gateway is working), the amount of memory that has been allocated to processes (in bytes), the total Heap memory that has been allocated (including process memory), and the total heap memory that was once used and is now free for allocation. The total free memory is the sum of the heap free memory and the free memory that has never been allocated (displayed on the top line). This sum is not shown in this window.

Subsequent lines of this window display information about each task, one per line.

The **ID** field gives the paragraph offset of the task, and is useful in making a distinction between multiple instances of the same task.

The **Description** field gives the name of the task.

The **Cycles** field gives the total number of times that the task has had a chance to run.

The **Usage** field gives an approximate percentage of processor time that has been used by the task. This value is a weighted smoothed sum that is continually updated. It is meant to give a relative indication of which tasks are using the most processor time.

The **Size** field gives the amount of memory (in paragraphs, which is 16 bytes) that is being used by the task. The total of all the Sizes is the process memory displayed on the second line (displayed in bytes).

The **Signal** field shows the current pending signal (if any) for the task.

The **State** field shows the current state of the task. It will be either Sleeping or Runnable.

Finally, the **Wake** field shows the next wakeup time for a sleeping task. If the task is sleeping but has no wakeup time, it will not wake up until it receives a signal.

Following is a quick description of each of the tasks.

3.4.1 The RCONSOLE Task

The RCONSOLE (remote console) task interfaces the window management code with a telnet session. If Rconsole is not enabled, an Rconsole task will still be created, but it will be only a stub function.

When the Rconsole is enabled and activated, this task is responsible for verifying that the user has access to the remote console facility. Once verified, the remote users screen is updated once per second. During a remote console session you will see the cycles and usage increase for this task. When the remote console session is inactive, this task will be sleeping.

3.4.2 The Network Processor Task

This task is always running and will never sleep. It is responsible for ensuring that the underlying TCP/IP connections are handled properly.

3.4.3 The Keyboard Handler Task

This task accepts keystrokes from the keyboard and from the remote console. It instructs the window manager to switch windows, scroll the screens, etc.

This task also runs the screen saver, when it is enabled. This task will never sleep.

3.4.4 The Queue Manager Task

This task is the main queue manager task for the entire program. It is responsible for polling all Novell queues. It also updates the information displayed in the Queue Manager window. This task determines when it must next be awoken by examining the poll times of each queue and scheduling a wakeup of itself at the appropriate time.

Additionally, various other tasks (mailer, smtp_deliver) will send wakeup signals to this task when they have completed their work.

3.4.5 The FingerD Task

This task spawns a TCP_DStream task to listen on the finger daemon TCP port. When it receives a connection it polls the attached Novell servers and returns the requested finger information to the TCP connection.

When a TCP connection is opened, the FingerD task will spawn a clone of itself. The clone will reopen a listening TCP port to handle subsequent connections. The original FingerD task will die when its job is complete.

See section 5. for more information about the FingerD task.

3.4.6 The TCP_DStream Task

This task interfaces the CUTCP TCP/IP libraries to the internal SYS V streams emulation. Whenever a TCP/IP connection is listening or open, one of these tasks will exist to handle it.

When a TCP_DStream task is first created, it spawns a TCP Worker task to assist it. Depending on the type of open, the TCP Worker may complete its task quickly (outgoing TCP connections) or may linger waiting for incoming connections (listening connections).

3.4.7 The TCP Worker Task

This task assists the TCP_DStream task in handling TCP connections. For TCP connections opened in listen mode, this task is awoken by the Network Processor task when a connection is established. It then signals the TCP_DStream task and dies.

For outgoing TCP connections, this task handles any domain name resolution, opens the connection to the remote host, then wakes up the TCP_DStream task and dies.

3.4.8 The SMTPD Task

This task listens on the SMTP TCP port (25). It waits for incoming SMTP mail messages and processes them accordingly.

This task will sleep until awoken by the TCP_DStream task. When a new connection is established, the SMTPD task may spawn a clone, depending on the settings of max_smtpds in the charon configuration file. (See section 2.4.5).

3.4.9 The SMTP_Deliver Task

This task delivers mail messages to the SMTP agent. It spawns a TCP_DStream task to obtain a TCP/IP connection. Only one of these tasks should ever be running. You will only see this task when mail is actually being transmitted to the SMTP agent.

3.4.10 The TelnetD Task

This task processes incoming Telnet connections for the Remote Console by translating the Telnet protocol to a data stream which the Remote Console can understand. It spawns a listening TCP_DStream to listen on the telnet port (23).

This task will only exist if the RConsole feature is enabled.

3.4.11 The Mailer Task

This task moves mail messages from the mailqueues to user mail directories or to the outgoing workqueue. It handles mail list expansion (in version 3.1), and broadcast message deliver. This task will only exist during the actual delivery of mail.

3.5 The Queue Manager Window

Appendix A, figure 5 shows a typical Queue Manager window.

This window displays information about attached Novell servers and the queues that are being processed by Charon.

The top portion of the window displays information about each attached server. This information is updated once every five minutes, or when an incoming FingerD connection is established.

The bottom portion of the window displays information about each queue.

The **Server** field displays the name of the attached Novell server. The **Queue** field displays the name of the associated queue. The **Mode** field displays the mode of the queue. The queue mode will be one of:

- **Incoming** Incoming Print Queue
- **Outgoing** Outgoing Print Queue
- **MailQueue** MailQueue or WorkQueue

The **Dest Host** field displays the destination host name for outgoing queues. It is not useful for incoming or mail queues. The **Dest Printer** field displays the destination printer name for outgoing queues. It is not useful for incoming or mail queues. The **Init** field displays the number of bytes contained in an optional initialization string for incoming and outgoing queues. The **Reset** field displays the number of bytes contained in an optional reset string for incoming and outgoing queues. The **Poll** field displays the poll frequency in seconds for outgoing/mail queues. The **Nxt Poll** field displays the next poll time for outgoing/mail queues. The **Last Host** field displays the host name of the last user of the queue. For mail queues this will display the host name of the sender. The **Last User** field displays the username of the last user of the queue. For mail queues this will display the user name of the sender. The **Files** field displays the total number of files transferred through the queue. The **Bytes** field displays the total number of bytes transferred through the queue. When the key mode is not idle, the bytes field displays the number of bytes remaining to be transferred through the queue (outgoing) or the number of bytes transferred (incoming). The **Avg Size** field displays the average size of each file/message transferred through the queue. The **Errors** field gives a count of the total number of errors experienced by the queue. The **TrnsTime** field displays the total time spent transferring data through the queue. When the queue is not idle, this field shows the total time spent transferring the current file/message. The **Thrpt** field displays the average throughput in Kbytes/second of the queue. The **Status** field displays the current status of the queue. Values displayed in this field might be:

- **Ok**
- **TimeOut** The last operation timed out
- **Held** The queue is currently held
- **Error** General Error Condition

The **Last File** field displays the name of the last field transferred through the queue. For mailqueues, this field shows the name of the last mail process which accessed the queue. Possible processes are:

- **Mailer** Mailer Deposited a File into this queue
- **SMTP-In** An incoming SMTP message was last put into this queue
- **SMTP-Out** An outgoing SMTP message was last transferred by this queue

The **Queued** field displays the number of files/messages currently waiting in the queue. The **Servers** field displays the number of servers currently attached to the queue. The **Map Name** field displays the name of an associated Map file (if any). The **Code** field displays the hexidecimal error code of the last error that occurred on the queue (if any). The **LastXfer** field displays the time of the last transfer through the queue. The **Queue Mode** field displays the current mode of the queue. Possible modes are:

- **Idle**
- **Opening** Opening a Connection to a remote host
- **Working** Internally transferring data
- **Transmitting** Actively transmitting data to a remote host
- **Receiving** Actively receiving data from a remote host
- **Closing** Closing a connection

The **Form Name** field displays the name of the last form type transmitted through the queue. This field is blank for mailqueues.

3.6 The SMTP Debug Window

Appendix A, figure 6 shows a typical SMTP Debug Window. This window displays information about incoming and outgoing SMTP sessions. An understanding of RFC821 will assist you in determining the cause of any SMTP transfer problems that you may encounter.

Chapter 4. RConsole - The Remote Console

The RConsole (Remote Console) facility enables system managers to remotely view Charon console information via the telnet protocol.

The RConsole process assumes that a remote user will 'telnet' to the gateway using a VT100 compatible terminal. The remote user will be required to 'log in' to the gateway by entering a userid and password. After a successful login, the remote user may switch screens and view them.

4.1 RConsole Installation

Two installation steps are required to enable the RConsole facility.

A. Add the RCONSOLE command to the charon.dat configuration file.

B. Create a group on one (or all) of your Novell Servers called **RCONSOLE.** Add remote console users to this group. (Only users may be in this group, not other groups).

4.2 Logging in using RConsole

When you telnet to the Charon gateway, you will be presented with a list of attached Novell file servers. The 'default' file server is the first one listed.

You will be presented with a login prompt. At this prompt you must enter the Novell userid of a user who is a member of the RCONSOLE group. If that user (or group) is not on the default file server, you may select another server from the displayed list and enter that before the user name in the standard Novell login fashion:

Login: **DRACO/BKC**

You will then be prompted for a password. Enter the password for the previously entered userid. The password will echo with '*' characters.

During the login process you may press <DELETE> or <BACKSPACE> to erase the previous character, <CTRL-U> to erase the entire line, or <ESCAPE> to abandon the login process.

The Telnet protocol is not overly secure and the password you enter will be transmitted in the clear (the same as any regular unix login). For security considerations, you may wish to create a Novell account which has a station restriction that does not exist. Enter this account into the RCONSOLE group. If a network snooper captures the password, the worst that could happen is that the snooper could remotely control the gateway. He would be unable to login into your novell server due to the station restriction. The Charon gateway uses the VerifyBinderObjectPassword call to determine if the password is valid, it does not actually login the user.

4.3 Remotely Controlling the Console

After successfully logging in as a remote console operator, you may use the following keystrokes to control the gateway:

- **<+>** Show Next Window
- **<->** Show Previous Window
- **<U>** Scroll Up

- **<D>** Scroll Down
- **<P>** Page Up
- **<N>** Page Down
- **<L>** Scroll Left
- **<R>** Scroll Right
- **<H>** Jump to Top of Window
- **<E>** Jump to Bottom of Window
- **<Q>** Exit Remote Console
- **<^X><^X><^X><^X>** Reboot the Gateway

Pressing the <ESC> key will not cause Charon to quit.

Pressing <^X> four times in a row will cause the gateway to reboot. All files are properly closed before the reboot takes place.

Chapter 5. The Finger Daemon

The FingerD task allows a remote 'finger' of the attached Novell servers. Users must finger the internet address/name of the gateway, not the attached servers. The username specified in the finger command may be:

- **blank**
- **A Novell Server Name**
- **A User Name**

Optionally, the string '/a' may be appended to the username to display the network address of users instead of their login time.

> Example: My gateway is called Romulus.erc, it is attached to Novell servers Draco and Darius. On a Unix system, I can enter the following commands:
>
> **finger @romulus.erc**
>
> (This will display all users on all attached Novell Servers)
>
> **finger draco@romulus.erc**
>
> (This will display only the users on the Novell server Draco)
>
> **finger bkc@romulus.erc**

(This will display only information about user bkc on any attached Novell server)

finger bkc/a@romulus.erc

(This will display information about user bkc on any attached server, and show the station addresses instead of the login time).

Chapter 6. Configuring Pegasus Mail for use with Charon[1]

Version 2 may be configured to use Charon as a mail gateway. The **PCONFIG** program is used to configure Pmail for use with the Charon gateway.

The following installation instructions must be followed for each Novell server that will be running Pmail and Charon.

[ ] *Log into your Novell Server as Supervisor*

[ ] *Execute the program PConfig*

[ ] *Select Define Clarkson Interface*

[ ] *Enter the Queue Name*

If you followed the instructions outlined in section 2.1, you should enter the name **mailqueue** in the queue name field.

[ ] *Enable the Queue*

Enter the value 'Y' in the Enabled field.

[ ] *Set the Preferred Field*

If you are not using an MHS gateway, enter 'Y' in the Preferred field. Otherwise, you must decide which gateway will have preference.

[ ] *Set the Use Always Field*

You must decide if you want Charon to handle all mail delivery on your Novell Servers. If you want Charon to only handle SMTP mail, enter the value 'N' in the Use Always

---

Field. Otherwise enter the value 'Y' to cause Charon to be used for all mail transactions.

[ ] *Set the Server Name Field*

In section 2.2 you obtained an Internet Name for this Novell server. You should enter that Internet Name in the Server Name Field.

[ ] *Set the Time Zone Field*

Set this field to the name of the local, current timezone.

Appendix A.

```
┌>─V──────────────────────────System Message Window──────────────────────────┐
21:36:19 Info: Found Server DRACO
21:36:19 Info: Found Mail Queue MAILQUEUE
21:36:19 Info: Found Server DARIUS
21:36:19 Info: Found Incoming Queue FOLAS
21:36:20 Info: Found Mail Queue MAILQUEUE
21:36:20 Info: TimeSync: added slave server draco
21:36:20 Info: TimeSync: added slave server darius
21:36:20 Info: Found SMTP_Out Queue WORKQUEUE Poll 10
21:36:20 Info: Found SMTP Incoming Queue draco/MAILQUEUE
21:36:20 Info: Log File Open Deferred for file 'vol1:usr\staff\bkc\sys_warn.ac

21:36:20 Info: Opened Log File 'vol1:usr\staff\bkc\sys_warn.act'
21:36:20 Info: Log File Open Deferred for file 'vol1:usr\staff\bkc\sys_info.ac

21:36:21 Info: Opened Log File 'vol1:usr\staff\bkc\sys_info.act'
21:36:21 Info: Log File Open Deferred for file 'vol1:usr\staff\bkc\smtp.act'
21:36:21 Info: Opened Log File 'vol1:usr\staff\bkc\smtp.act'
21:36:21 Info: Created Logger Process system
21:36:21 Info: Created Logger Process system
21:36:21 Info: Created Logger Process smtpout
21:36:21 Info: Created Logger Process mailer
21:36:21 Info: Created Logger Process smtpin
└────────────────────────────────────────────────────────────────────────────┘
```

Figure 1.

```
┌>^V──────────────────────────System Message Window──────────────────────────┐
21:36:21 Info: TimeSync: Activated
21:36:21 Info: Deferred Open on Log File 'vol1:usr\staff\bkc\sys_warn.act' ok
21:36:21 Info: Deferred Open on Log File 'vol1:usr\staff\bkc\sys_info.act' ok
21:36:26 Info: RConsole:: Listening on Telnet Port
21:36:26 Info: Video regs ax=1a00 bx=f1 vmode=3
21:36:26 Info: VideoSubsystem 1 Display 3
21:36:26 Info: Started Charon 3.1
21:36:32 Info: TimeSync: Checking times
21:36:32 Info: TimeSync:Sent RDate request to omnigate
21:36:32 Info: TimeSync: Gateway time changed by 1 seconds.
21:36:33 Info: Time on Server DARIUS changed by 1 seconds.
21:36:41 Info: RConsole Open From 128.153.28.65:7420
21:37:04 Warn: RConsole::Attempted Login by Non_Console Operator DARIUS/BKC
21:37:05 Info: RConsole Closed
21:37:05 Info: RConsole:: Listening on Telnet Port
21:37:08 Info: RConsole Open From 128.153.28.65:7931
21:37:34 Info: RConsole: User DRACO/BKC Logged In
21:38:20 Info: DRACO/MAILQUEUE Opened MailJob 496
21:38:20 Info: Mailer: DRACO/MAILQUEUE:496 Deliver to user bkc at omnigate.cla
son.edu Class 3 Result 1
21:38:21 Info: DRACO/WORKQUEUE Created Job 256

└────────────────────────────────────────────────────────────────────────────┘
```

Figure 2.

```
┌>─V──────────────────────────Stream Status Display──────────────────────────┐
        Data Block Buffers
Allocations: 102 Deallocations: 102 Refusals: 0 Oversize: 0
──────────────────────────────────────────────────────────────────────────────
     Size Count  (Bytes) Free   Perc   Allocations  Perc  Refused (Bytes)

       64    64    4096    64   100%            52    50%       0    3328
      128    32    4096    32   100%             4     3%       0     512
      256    16    4096    16   100%             1     0%       0     256
      512     8    4096     8   100%            45    44%       0   23040
     1024     8    8192     8   100%             0     0%       0       0

 Total      128   24576   128                  102                  17289
──────────────────────────────────────────────────────────────────────────────
      Message Block Information
```

```
Messages: 120 Free: 120 Allocations: 102 Refusals: 0

        Queue Information
   Name             Count Flags   High  Low   Msgs DataBytes
┌─ TCP_Down         0     H--E-I  4192  1024  0    0
└─ SMTP_Down        0     ------  4192  1024  0    0

┌─ SMTP_Up          0     H--E--  4192  1024  0    0
```

Figure 3.

```
─>─V───────────────────────────Task Display Window──────────────────────────
Tasks: 15  Free Mem: 129792 Up 0 Days 0 Hrs 2 Mins    Sat Feb 09 21:39:23 1991
Load : 37% Proc Mem: 28272 Heap Mem Used: 128832 Free: 4880
ID   Description                Cycles Usage Size  Signal   State    Wake

7af8 TCP_DStream                   21     0%   12  None   Sleeping
7b05 TelnetD                        0     0%   21  None   Sleeping
785b Stream Status Display        120     9%  583  None   Sleeping 21:39:28
76a3 Task Display                  59    22%  380  None   Sleeping 21:39:25
7640 TimeSync                      11     0%    3  None   Sleeping 00:36:33
74c4 TCP_DStream                    0     0%   12  None   Sleeping
763a TCP Worker                     1     0%   58  None   Sleeping
746a SMTPD                          0     0%   89  None   Sleeping
72f0 TCP_DStream                    0     0%   12  None   Sleeping
7466 TCP Worker                     1     0%   58  None   Sleeping
72ce FingerD                        0     0%   33  None   Sleeping
6b5c Queue Manager                 17     0%  345  None   Sleeping
6b6e Keyboard Handler               2K    1%    6  None   Runnable
6b4a Network Processor              2K    5%    3  None   Runnable
603f RConsole                     119     0%  152  None   Sleeping 21:39:24
```

Figure 4.

```
─>─V──────────────────────────────Queue Manager─────────────────────────────────
Server          Userid          Version  Current/Peak/Max Connections
DARIUS          LPR              2.15           7/  44/100
DRACO           LPR              3.1           24/  59/250
────────────────────────────────────────────────────────────────────────────────
Server          Queue           Mode      Dest Host       Dest Printer
Init      Reset Timeout    Poll Nxt Poll  Last Host       Last User
Files     Bytes Avg Size Errors TrnsTime  Thrpt    Status Last File
Queued  Servers Map Name   Code LastXfer  Queue Mode      Form Name
────────────────────────────────────────────────────────────────────────────────
DARIUS          FOLAS            Incoming  DARIUS          FOLAS
0             0 10            60 --:--:--
0             0 0              0 --:--:--  0.0          Ok
0             0 folas            --:--:--  Idle
────────────────────────────────────────────────────────────────────────────────
DRACO           WORKQUEUE        MailQueue DRACO           WORKQUEUE
0             0 10            10 21:39:33  draco.erc.clark BKC
1           423 423            0 00:00:12  0.3          Ok SMTP-Out
0             1 None             21:38:36  Idle
────────────────────────────────────────────────────────────────────────────────
DARIUS          MAILQUEUE        MailQueue DARIUS          MAILQUEUE
0             0 10            10 21:39:33
0             0 0              0 --:--:--  0.0          Ok
────────────────────────────────────────────────────────────────────────────────
```

Figure 5.

```
─>^V────────────────────────────SMTP Debug Window───────────────────────────────
21:36:21 SMTPD: Created
21:38:34 SMTP_Deliver: Created
21:38:34 SMTP_Deliver: Entering Parsing State from state 2
21:38:36 SMTP_Deliver: RESP: 220 omnigate.clarkson.edu Server SMTP (Complaints
ugs to:  postmaster@omnigate.clarkson.edu)
21:38:36 SMTP_Deliver: Sending: HELO romulus.erc.clarkson.edu
21:38:37 SMTP_Deliver: Sent Hello. Waiting
21:38:38 SMTP_Deliver: RESP: 250 omnigate.clarkson.edu
21:38:38 SMTP_Deliver: Sending: MAIL FROM: <BKC@draco.erc.clarkson.edu>
21:38:40 SMTP_Deliver: RESP: 250 OK
21:38:40 SMTP_Deliver: Sending RCPT's
21:38:40 SMTP_Deliver: Sending: RCPT TO: <bkc@omnigate.clarkson.edu>
21:38:41 SMTP_Deliver: RESP: 250 Recipient OK.
21:38:41 SMTP_Deliver: Sending DATA
21:38:44 SMTP_Deliver: RESP: 354 Enter Mail, end by a line with only '.'
21:38:44 SMTP_Deliver: Data:..........:Done
21:38:48 SMTP_Deliver: RESP: 250 Submitted & immediates started (msg.aa16271)
21:38:48 SMTP_Deliver: Message Accepted
21:38:48 SMTP_Deliver: RESP: 221 omnigate.clarkson.edu says goodbye to romulus
rc.clarkson.edu at Sat Feb  9 21:38:49.
21:38:48 SMTP_Deliver: Connection Closed
21:38:48 SMTP_Deliver: Destroyed
────────────────────────────────────────────────────────────────────────────────
```

Figure 6.

Appendix B.
Sendmail Installation Suggestions

Date: Thu, 25 Apr 1991 10:06:37 EDT
From: "John M. Wobus" <JMWOBUS%SUVM@clvm.clarkson.edu>
To: bkc@omnigate.clarkson.edu
Subject: Configuring sendmail for use with Charon
cc: Donald E Hanley <sysdeh@cns.cns.syr.edu>

The Charon manual suggests using a Unix system to store and forward mail from the outside world to a Charon gateway and gives a hint as to how to configure MMDF to forward the mail.

We run sendmail, and I was faced with the task of making sendmail do the same thing. I discovered there is no quick, easy solution, but I wrote up what I found out anyway, which might be helpful to anyone else in the same situation. Below is the writeup. You are welcome to insert it in the Charon distribution directory if you think it would be useful.

John Wobus
Syracuse University

CONFIGURING UNIX SENDMAIL FOR USE WITH CHARON GATEWAYS.

This is an application-note on how to configure the Unix sendmail program so that a Unix system can store and forward mail directed to Novell servers. Some Unix systems use sendmail to handle incoming mail while others use MMDF. The Charon manual gives hints for proper configuration of MMDF, but not for Sendmail.

You need to do this if you use MX records (in the Internet Domain Name System) to route mail destined for the Novell servers through some Unix system that runs Sendmail. The Unix system's job is to receive mail addressed to the Novell server and forward it to the Charon gateway. Without special configuration, the Unix system would reject the mail.

You do not need to do this if you choose to have mail delivered directly to the Charon gateway rather than through a Unix system (that runs sendmail) first. Interposing a Unix system in the path makes some sense if the Unix system is up more of the time and/or has more space for storing queued mail. You do a favor to the systems sending mail to the Charon gateway if you take the mail off their hands sooner rather than later. If the sending system is on the other side of a wide expanse of Internet, then receiving the mail as soon as possible saves some load on the Internet since it doesn't have to deal with repeated retries.

Sendmail configuration files are complex, thus the necessity for this application note. A problem is that different Sendmail configuration files can be quite different from each other, so there is no sure-fire place you can put something and know that it will work. You have to learn just a little about Sendmail and use some common sense to change a typical Sendmail configuration file to forward mail in this fashion.

Disclaimer: I'm no sendmail expert. I'm writing this because I had to learn enough about sendmail to make this work and I thought I'd pass along the knowledge. Thus, this is pretty much all I know: questions to me won't be productive--ask on Usenet. The one thing that I do know is that, depending on your present sendmail.cf file, all this might not work. However, if any sendmail experts see outright errors, I'd appreciate hearing about them.

John Wobus
Syracuse University
jmwobus@syr.edu
August 14, 1991
=-=

BASICS:

Sendmail is a Unix program which (among other things) takes all incoming mail and dispatches it based upon what it is configured to do. Its configuration resides in a file, typically called "/etc/sendmail.cf". Sendmail is typically called in the process of sending a message. Also, a daemon, started when the Unix system starts, processes incoming mail as well as wakes up every once in a while (typically 30 minutes) to retry a queue of any messages that failed to get through.

The changes outlined below are in the sendmail.cf file. To change it, edit a copy of the sendmail.cf file, put it under the name /etc/sendmail.cf (taking whatever precautions you want to back up the existing file) and stop & restart the daemon.

QUICKEST AND DIRTIEST METHOD:

Assuming the Novell server's internet name is

    nimbus.state.edu

and the Charon gateway's internet name is

    charlie.state.edu

and the sendmail has a mailer configuration called "tcp", then insert the following rule:

```
R$*<@nimbus.state.edu>$*    $#tcp$@charlie.state.edu$:$1<@nimbus.state.edu>$2
```

(note: lines beginning with "R" are called rules; they have two or
three parts separated by blank space; the third part is a comment)
You can add a succession of these new rules, one for each Novell server.

The place you insert it is important. You insert this under Rule-set 0, i.e., after the line marked:

```
S0
```

and before any other lines S1, S2, S25, etc, which mark the beginning of other rule-sets.

You must place the rule just before any other rule that handles the sending of mail to your own domain (if there is such a rule). If there is no such rule, then put it before the rule that sends to the Internet in general. Example of rule that sends to your own domain:

```
R$*<@$*.$D>$*       $#tcp$@$2.$D$:$1<@$2.$D>$3   user@host.state.edu
```

(note: this rule assumes that earlier in the sendmail.cf file, there is
a macro called D which is defined to be state.edu. This is done with
the macro definition:

```
DDstate.edu
```

)

Example rule that sends to the Internet in general:

```
R$*<@$+>$*          $#tcp$@$2$:$1<@$2>$3         user@some.where
```

MAILERS:

All the above rules use the string "#tcp" on the assumption that the sendmail.cf file has a mailer configured under the name "tcp". If not, you will have to figure out the name used to configure the mailer for mail sent to Internet sites and use it in place of the string "tcp" in the above rules. Two other sendmail.cf files I inspected used the names "ddn" and "arpa-mailer" respectively. Another one defines one, "localsmtp" for mail to local Internet hosts and "nonlocalsmtp" for other Internet mail.

Mailers are associated with their names & otherwise configured with M statements in the sendmail.cf file. Following is a typical M statement to define a mailer to send & receive Internet mail:

```
Mtcp,    P=[IPC], F=mDFMueXLCE, S=14, R=24, A=IPC $h, E=\r\n
```

The thing that determines that the mail is delivered through SMTP is the parameter P=[IPC]. However, the sendmail.cf file probably has more than one mailer name that uses SMTP. The name of the correct mailer is probably something that suggests the Internet. A name involving uucp (or some other non-Internet network) which specifies P=[IPC] is probably a second mailer specification for Internet mail used only for mail routed through some specific set of gateways (e.g. gateways to uucp networks).

A SLIGHTLY BETTER METHOD:

I said this is quick and dirty because it ignores sendmail's ability to keep definitions in one place, i.e., to define a macro to be the string "state.edu" and avoid scattering that string around rules all over the place. The following rule makes use of a macro called D defined to be "state.edu":

```
R$*<@nimbus.$D>$*       $#tcp$@charlie.$D$:$1<@nimbus.$D>$2
```

THE METHOD I USED:

I used a method that localizes the list of servers and gateways using Sendmail's macro and class mechanisms. To do so, I defined a macro and a class for each gateway. The macro is the name of the gateway and the class is the list of servers it serves. I chose the letter "O" as the macro name and the letter "N" as the class name. Unfortunately, you have to choose single upper-case letters for these names and have to make sure you get unused letters by searching through your sendmail.cf file for other definitions.

Assume charlie.state.edu is the server for nimbus.state.edu,
nullo.state.edu, and natural.state.edu

In the definitions section of the sendmail.cf file (near the top):

```
DOcharlie
CNnimbus nullo natural
```

We assume the following appears somewhere:

```
DDstate.edu
```

In rule-set 0 section (just before the rules that deliver other Internet mail within state.edu):

```
R$*<@$=N.$D>$*        $#tcp$@$O.$D$:$1<@$2.$D>$3     user@novell-server
```

SUBDOMAINS:

My site is now avoiding subdomains so I got to avoid figuring out how to make Sendmail handle them. My guess is that you can take any of the rules above and substitute charlie.cc for charlie, nimbus.cc for nimbus, etc. as long as you do it for all the rules and definitions.

A VERY LITTLE ABOUT SENDMAIL CONFIGURATION:

A rule-set is scanned in order.
The first part of each rule is an expression which can match addresses.
The second part of each rule is an expression which formats how it is
  rewritten.
Rule-set zero's job is to take a weird form of the envelope's recipient
  address (with angle brackets around the "@" sign and recipient host,
  e.g. jjdoe<@host.state.edu>) and decide where to send it and what
  recipient address to put on the envelope as it sends it.
Statements:
 Da... defines a macro so $a represents a particular token.
 Ca... ... ... defines a class so $=a represents any token in the class
 Sn starts rule-set n.
 R... ... is a rule.
 Mxxx ... defines mailer "xxx".
Matching (first part of rule):
 $* matches any number of tokens.
 $+ matches one or more tokens.
 $=a matches any token in class a (defined earlier by Ca...).
 $a (a:alpha) matches the token represented by macro a (defined earlier
    by Da...).
 other characters (e.g. ><@abcde...) match themselves.
Formatting (second part of rule):
 $1 inserts the first match from first part ($*,$+, or $=x)
 $2 inserts the second match
 $3 inserts the third match, etc
 $a (a:alpha) inserts the text of a macro a (defined earlier by Da...)
 other characters are simply inserted
 $#xxx$@aaaaa$:bbbbb is a special kind of 2nd part which tells sendmail
    to send the mail through mailer "xxx" to aaaaa using bbbbb as the
    recipient address on the envelope. Note that bbbbb is still in
    the weird syntax, e.g. jjdoe<@state.edu>

MORE ON LEARNING SENDMAIL:

If you must learn sendmail, you probably want to be familiar with RFC822 header format and with SMTP protocol. References for sendmail are the sendmail man page, the sendmail operations guide (in the Unix system manager's manual for Unix's that have sendmail) and the book called "Unix System Administration Handbook" by Evi Nemeth, Garth Snyder, and Scott Seebass; Prentice Hall, 1989.

In trying to figure out how to write these rules, I got nowhere until I started using sendmail with the -bt option to try things.

Appendix C - Available Packet Drivers

The Clarkson packet driver collection

Availability

The Clarkson collection of packet drivers is available by FTP, by archive-server, and by modem. They come in two flavors -- executables only (drivers.zip), and source+executables (driverss.zip). All of the following instructions apply to both drivers.zip and driverss.zip.

Mail:

I distribute the packet drivers on two 360K 5.25" disks, or a 720K 3.5" disk. I charge a fee for the service of copying and mailing these disks. You can send me a check for $20, or you can send me a purchse order and I will bill you for $22. NY residents add 7% sales tax, overseas orders add $3 for shipping. If you send a check, please be sure it is in US dollars -- the bank charges me $15 to convert checks drawn in foreign currencies.

Russell Nelson
11 Grant St.
Potsdam, NY 13676

FTP:

sun.soe.clarkson.edu:/pub/ka9q/drivers.zip
grape.ecs.clarkson.edu:/pub/msdos/tcpip/drivers.zip

E-mail:

Send mail to archive-server@sun.soe.clarkson.edu and put the following command as the body of your message:

help

This will send you a help message. Reading this help message will tell you how to fetch the packet drivers.

Modem:

Call the Clarkson Heath User's Group's BBS: (315)268-6667, 8N1, 1200/2400 Baud, 24 hours. You may need to press break, or simulate it using several nulls. Download pub/msdos/tcpip/drivers.zip.

The Driver List Itself

There are now two packet driver distributions. The smaller one, drivers.zip, contains only the executable programs on the list below, this file, DRIVERS.DOC, and COPYING.DOC.

3C501.COM
3C503.COM
3C505.COM
3C507.COM
3C523.COM
AR450.COM
ARCNET.COM
AT&T.COM
DE600.COM
DEPCA.COM
EXPRESS.COM
HPPCLAN.COM
IBMTOKEN.COM
IPXPKT.COM
IPXSTAT.EXE
ISOLAN.COM
LOCALTLK.COM (requires a modified version of Phil Karn (KA9Q)'s NOS.
NB.COM
NCRET105.COM
NE1000.COM
NE2.COM
NE2000.COM
NI5010.COM
NI5210.COM
NI6510.COM
NI9210.COM
NTI16.COM
PKTADDR.COM
PKTALL.COM
PKTCHK.COM
PKTMODE.COM
PKTMULTI.COM
PKTRING.COM
PKTSEND.COM
PKTSTAT.COM
ROMREL.COM Release the Novell boot rom's control of an Ethernet card.
SLIP8250.COM
TIARA.COM
TRACE.COM
UBNICPC.COM
UBNICPS2.COM
WD8003E.COM

Appendix D. -- How to Interface with Charon

This section details how users can submit and retrieve jobs from Charon. You may wish to read it to understand how Charon works. For the most part, its targeted towards developers of user interfaces/agents (such as David Harris and Pmail).

Mail Delivery

Charon delivers mail to users via the old Novell SYS:MAIL directory system. Most users have a directory in the SYS:MAIL directory of their server with a directory name equal to the hex representation of their object id. For example, the Supervisor's mail directory is

SYS:MAIL\1

Charon deposits incoming messages, one per file, in the target mail directory. File names have the form

BC######.CNM

where ###### represents a hexadecimal number (equal to the number of ticks since midnight on the day of delivery). This format may change in the future. However to retain compatibility with Pegasus Mail, only the file name (not the extension of .cnm) will change.

There is no easy way to control the delivery location of mail messages. A person could create an alias entry for every user on their system, and direct the alias towards a particular user. This would result in all messages being delivered to one mail directory.

Incoming messages are stored in 'Mail Normal Form'. (I just made that name up). Basically that means RFC822 format messages created by SMTP based mail agents (including Pmail). Every line of text ends with a single **<CR><LF>** pair. There is no trailing **^Z** on the file.

Mail Submission

All mail that is to be delivered by Charon must be deposited into the MAILQUEUE in a special form. Submission is accomplished using Queue Services API calls.

The format of deposited messages is:

```
Header String
-------------
Destination Address List
------------------------
End of List Marker
```

Mail Normal Form Message (RFC822 compliant)

The **Header String** is

$$**<CR><LF>**

The **Destination Address List** is a list of recipients, one per line. The form of each address is

User@node.domain**<CR><LF>**

where node can be a local Novell file server name, or an internet name.

The **End of List Marker** is a single line consisting of only:

<CR><LF>

Only one message may be submitted per queue job. The number of recipients is limited only by the operating memory of the gateway itself. At least 200 addresses can be accomindated given 20K of free RAM at the time of message processing.

Important Charon only recognizes Job Type 101 as a valid mail message. Any other Job Type will be discarded. This inhibits users from using Nprint or Capture to submit messages to the Mailqueue. Imagine what a binary graphics file would do to Charon’s address parser!

Sample Message

```
$$
recipient1@node.node.domain
recipient2@novell_server1

Date:    15 Aug 91 06:44:19 GMT
From:            "Marcus Welby" <mw@novell_server2.domain>
To:      recipient1@node.node.domain,
         recipient2@novell_server1.domain
Subject: Hi there test message

And this is the body of the message
```

Caveats

Charon does not look at the text of the message. If your program improperly formats the From: field, or the To: field, Charon won't fix it.

Charon trys to be smart about handling 'chatty' addresses in the recipient list. A chatty address is in the form:

"Some long user garbage" actual_user@realhost.domain

However, you'ld do Charon a favor if you sent only the actual user address in the recipient list.

The From: address and To: (and CC:, Bcc:) fields in the header of outgoing messages SHOULD be completely normalized internet addresses. You have no way of knowing if one or more recipients is forwarding his/her mail to an internet site. If this happens, and the user attempts to reply to an address thats in 'short form' (such as the second address in the recipient list of the example message), the reply won't get delivered. Its ok to have short form addresses in the recipient list, Charon matches the host name to both the Novell name list, as well as the internet name list.

It probably isn't overly worthwhile to work on an MHS gateway for Charon, as I expect to implement MHS support within Charon this year.

Index