Gcom®
XOT Daemon
User Guide

October, 2007

Gcom, Inc.

1800 Woodfield Drive
Savoy, IL 61874

Voice: 217.351.4241
Fax: 217.351.4240

Email: support@gcom.com
http://www.gcom.com

Dave Grothe was the subject matter expert for this manual.

Table of Contents

List of Tables

List of Figures

Introduction

The Gcom XOT daemon is a user process that runs on Gcom's GPA2G. It implements the XOT protocol as described in RFC 1613. The XOT daemon uses Gcom's SyncSockets protocol to interface to the X.25 access lines and uses the system's socket interface to interface to TCP/IP.

Gcom's XOT daemon implements the full RFC 1613 protocol elements including both SVC and PVC channels.

Gcom's XOT daemon incorporates certain additional features such as load balancing of the X.25 lines and call redirection not called for in the RFC. The routing of X.25 calls is performed via address pattern matching with "wildcard" characters allowed much as in shell file name expansion. It also is capable of incorporating X.25 call user data field values in its call routing decisions.

The daemon also includes numerous debugging aids that can be used to trouble shoot problems that may arise in the use of the XOT protocol.

The daemon reads a configuration file at startup time to set the X.25 interface parameters and all the call routing patterns that it will use. The configuration file is an ASCII text file with intuitive syntax and the ability to incorporate comments. Typically, however, the configuration file is generated by the Gcom Management Console, a browser based point and click configuration utility.

In addition Gcom provides an XOT management utility program that can be run to interrogate connection status, statistics and trace information from the XOT daemon. This program communicates with the XOT daemon via a TCP connection to a dedicated port number on which the daemon is listening. The utility program can be run on the same system as the daemon or on any other machine that can establish a TCP connection to the system on which the daemon is running.

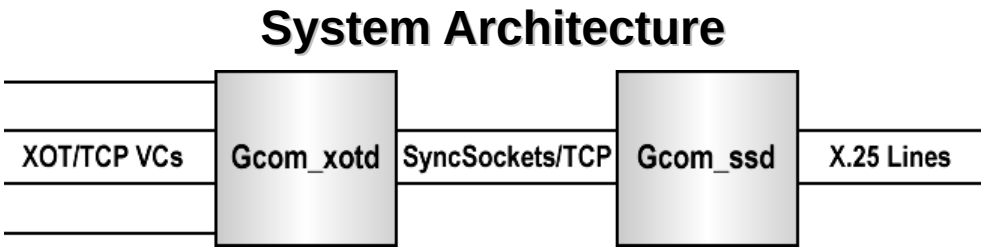

Figure 1 – Gcom SyncSockets API

The Gcom XOT daemon is called "Gcom_xotd". It is a user level process. It uses Gcom's SyncSockets connections to communicate with Gcom's SyncSocket Daemon, Gcom_ssd. Each SyncSocket connection represents one X.25 access line. The X.25 access lines are configured for the usual synchronous serial options and specify a LAPB protocol stack, without the X.25 packet level. That is why there is only one SyncSockets connection per access line rather than one per X.25 virtual circuit.

Gcom_xotd performs the XOT protocol as specified by RFC 1613 and uses TCP connections (left side of illustration) to carry X.25 virtual circuit traffic. Each TCP connection corresponds to a single X.25 virtual circuit.

Gcom_xotd receives X.25 packets as I-frame data from the LAPB lines. It decodes each packet and applies call routing algorithms to X.25 Call Request packets. It remembers the local logical channel number (LCN) of each local virtual circuit and ensures that all packets sent across the local interface contain the proper LCN.

An X.25 Call Request packet causes a TCP connection to be established to a remote XOT host. This TCP connection is associated, on a one to one basis, with the X.25 virtual circuit that caused the connection to be made.

Incoming TCP connections are accepted and XOT Call Request packets received on these TCP connections are routed to one of the LAPB interfaces. In so doing Gcom_xotd ensures that a local non-conflicting LCN is used for the virtual call.

If Permanent Virtual Circuits (PVCs) are configured, Gcom_xotd opens TCP connections to the appropriate remote XOT hosts and negotiates PVC setup parameters.

At initialization time, when the LAPB links are brought up, Gcom_xotd waits for the DTE (or DCE) to send a Restart Request packet. Gcom_xotd will only initiate the restart sequence after the expiration of a timeout. If the local DTE/DCE sends the Restart Request packet first then Gcom_xotd adapts to the role of DCE or DTE, respectively, based upon analysis of the cause field value in the Restart Request packet.

Given that Gcom's LAPB protocol stack can be configured for automatic DTE/DCE role negotiation this means that Gcom's XOT GPA can be connected to an X.25 DTE or DCE without regard to whether the other device is a DTE or a DCE and have the roles established automatically.

Gcom's XOT GPA can be substituted for an X.25 switch with no reconfiguration of the local DTE.

Running Gcom_xotd

Gcom_xotd is run with certain command line parameters, the most important of which is the name of its configuration file. The following table summarizes the command line options.

Option	Description
`-B`	Run in background mode. Default is foreground.
`-e filename`	Set name of configuration file. Default: xot.cfg
`-P filename`	Set name of PID file. The PID file is used by management software to be able to terminate Gcom_xotd. Default: ./Gcom_xotd.pid
`-v`	Print version information.
`-h or -H`	Print brief summary of command line options.

Table 1 – Gcom_xotd Parameters

Port Number Usage

Gcom_xotd uses the following IP port numbers.

Port	Usage
1998	Standard XOT listening port.
8000	Used to connect to local instance of Gcom_ssd.
8200	Statistics and management port for Gcom_xotd. The utility program Gcom_xot connects to this port to obtain statistics and trace information from Gcom_xotd.
Dynamic	Ports assigned by IP for use with outgoing TCP connections.

Table 2 – Port Number Usage

Configuration File Format

The configuration file that Gcom_xotd reads at initialization time is an ASCII file. Lines in the file that begin with the character '#' are considered comment lines. Individual lines can also be commented by placing a '#' followed by the comment text at the end of the line. Blank lines are ignored.

Generally a sequence of tabs and/or spaces functions syntactically as a single space. Spaces or tabs can be used almost anywhere for formatting purposes.

The only exceptions are:

- A "label" must begin in column 1.
- Labels cannot contain spaces or tabs.
- Numbers cannot contain spaces or tabs.
- If a string contains spaces the spaces are significant.

The file is divided into sections and each section begins with a label. Labels must be of a certain form in order to identify the section to which they pertain (see below).

The parameters that are associated with any section must be indented by at least one space or tab following the label for the section. So the general form of a configuration section is the following.

```
Label
   Param1 = Value1
   Param2 = Value2
   Param2 = Value3
   Etc.
```

In general labels must have a certain prefix to identify the type of section that they label. The remainder of the label is up to the user and does not even have to be unique.

Within a given section of the configuration file parameter names may be repeated. In such cases the value associated with the last reference to the parameter, reading from top to bottom, is the value that the parameter will assume. In the example the line "Param2 = Value3" is an example of a repeated parameter.

Parameters specified as type Number can be any decimal, octal or hexadecimal number using C language format. Parameters specified as type String are enclosed in quotes ("").

Some String valued parameters specify hexadecimal values, for example, "`01ABCD`". These parameters should contain an even number of characters

enclosed in quotes, and no spaces. They are converted pair-wise to binary values that occupy consecutive bytes in memory. The letters A-F can also be written in lower case as a-f. With these types of parameters there is no initial "0x" on the front of the value since hexadecimal is assumed.

The file is processed by finding the section for global parameters and processing them, followed by finding routing parameters and PVC configuration parameters.

When processing one type of section, Gcom_xotd simply goes from label to label for labels that identify sections of that type. Thus there is no requirement for uniqueness of labels.

The following table summarizes the different types of section labels.

Label Prefix	Type of Label
GLOBAL	Parameters that follow set global options for Gcom_xotd, such as log file name and debug options. There should only be one of these labels in the file.
X25.	Parameters that follow specify properties of an X.25 interface including port number and channel ranges. There should be one such entry per X.25 port.
X25_TO_TCP.	Parameters that follow define a route from X.25 to a remote XOT/TCP host. Basically an X.121 address is associated with a remote IP address. There can be multiple entries of this type, one for each distinct route to a remote XOT host.
TCP_TO_X25.	Parameters that follow define a route from an XOT/TCP connection to an X.25 interface. This route applies to Call Request packets that arrive on the XOT/TCP connection. Basically it amounts to associating an X.121 address to an X.25 port. There can be multiple entries of this type. At minimum there should be one per X.25 port.
X25_PVC.	Parameters that follow specify a PVC for a particular X.25 port. The parameters include the remote XOT host's IP address and logical channel number. There needs to be one entry of this type for each defined PVC.

Table 3 – Section Labels

Global Parameters

These are parameters that pertain to Gcom_xotd as a whole.

Example:

```
GLOBAL
   num_connections = 200
   connect_timeout = 10
   log_name        = "/usr/spool/gcom/xot.log"
   log_options     = 0x00C00003    # production setting
   log_size        = 8000          # 8 mega-bytes
```

The following table describes the global parameters.

Parameter	Type	Description
num_connections	Number	Total number of TCP connections. This value is used to size a polling list maintained internally by Gcom_xotd. It should be at least as large as the sum of all PVC and SVC channel ranges for all X.25 ports.
connect_timeout	Number	Number of seconds to wait for a connection to a LAPB port. The connection is retried if it does not complete in this amount of time.
log_name	String	Name of the log file maintained by Gcom_xotd.
log_options	Number	Usually specified in hexadecimal. These are single bit logging options that control which functions of Gcom_xotd write messages to the log. See below.
log_size	Number	The size of the log in kilo-bytes. If set to zero the log will grow without bound. If set to a non-zero value then the log will wrap around when it reaches this size.
errlog_name	String	Name of the error log file maintained by Gcom_xotd. Messages that are classified as error messages are written into this file in addition to the log file.
errlog_size	Number	The size of the error log file in kilo-bytes. If set to zero the log will grow without bound. If set to a non-zero value then the log will wrap around when it reaches this size.
trace_buffer_size	Number	Number of trace buffer entries maintained by Gcom_xotd on a per connection basis. Default is 512.
keepalive_time	Number	Number of seconds before TCP sends the first keep-alive probe on an XOT TCP connection. Set to zero to defeat the keepalive mechanism.
keepalive_intvl	Number	Number of seconds between TCP keep-alive probes on an XOT TCP connection. Set to zero to defeat the keepalive mechanism.
keepalive_probes	Number	Number of retries before an XOT TCP connection is disconnected.

Table 4 – Global Parameters

Logging Options

The `log_options` parameter specifies a set of single bit logging options that control the types of messages that are written to the log file. These can be used for usage logging or for trouble shooting.

The following table summarizes these option bits.

Value	Description
`0x00000001`	Log error conditions.
`0x00000002`	Log connection setup and disconnect summary.
`0x00000010`	Log I-frames as they are received from LAPB ports.
`0x00000020`	Log I-frames as they are sent to LAPB ports.
`0x00000040`	Log XOT packets as they are received from TCP connections.
`0x00000080`	Log XOT packets as they are sent to TCP connections.
`0x00000100`	Write messages to the log pertaining to choosing a route from X.25 to a remote XOT host.
`0x00000200`	Write messages to the log pertaining to choosing a route from an XOT host to an X.25 port.
`0x00000400`	Log X.25 facilities that are included in each Call Request packet.
`0x00000800`	Log all Call Request packets.
`0x00001000`	Log all Clear Request packets.
`0x00002000`	Log all Reset Request and Confirm packets.
`0x00004000`	Log all Interrupt packets.
`0x00008000`	Log all Restart Request and Confirm packets.
`0x00010000`	Log all Data packets.
`0x00020000`	Log details of Call Request packets.
`0x00040000`	Log details of receiving TCP packets including message fragments.
`0x00080000`	Log TCP connection setup progress.
`0x00100000`	Log messages sent on the SyncSocket connections to LAPBs.
`0x00200000`	Log messages received from the SyncSocket connections to LAPBs.

0x00400000	Log SyncSocket LAPB connection setup messages.
0x00800000	Log SyncSocket LAPB disconnect messages.
0x01000000	Log messages pertaining to the running of the internal polling list (very verbose output).
0x02000000	Log messages pertaining to setting up PVCs.
0x04000000	Log messages pertaining to the statistics interface.
0x10000000	For certain other log options, print the entire data buffer rather than an excerpt of the first few bytes.

Table 5 – Logging Options

The following table provides some example logging option values.

Example Value	Description
0x00C00003	Good setting for production.
0x000000F3	See packets entering and leaving Gcom_xotd.
0x00C00303	See routing algorithms in operation. Good for verifying that route configurations are working as intended.

Table 6 – Logging Option Examples

X.25 Interface Parameters

For each X.25 interface supported by Gcom_xotd there must be a section in the configuration file in which the labels have the prefix “X25.”.

Example:

```
X25.1
   port_number         = 1      # serial port number from 1
   lo_pvc              = 100    # lowest PVC channel number
   hi_pvc              = 103    # highest PVC channel number
   lo_svc              = 1      # lowest SVC channel number
   hi_svc              = 16     # highest SVC channel number
   suppress_facilities = 1  # suppress facilities to X.25 host
   suppress_facilities = 0  # deliver facilities to X.25 host
   window_size         = 4      # packet window size
   packet_size         = 256    # packet level packet size
```

The following table specifies the parameters used to configure an X.25 port.

Parameter	Type	Description
port_number	Number	The port number of the LAPB interface to be configured.
lo_pvc	Number	Lowest PVC channel number. Zero means no PVCs. PVCs must be configured via a special section in the configuration file, see below.
hi_pvc	Number	Highest PVC channel number.
lo_svc	Number	Lowest SVC channel number. Zero means no SVCs.
hi_svc	Number	Highest SVC channel number.
suppress_facilities	Number	If set to 1, suppress all facilities in calling and clearing packets sent to this interface. Set to 0 to allow passing of facilities.
window_size	Number	Default window size for the interface.
packet_size	Number	Default packet size for the interface.

Table 7 – X.25 Interface Parameters

X.25 to TCP Routing Parameters

Each appearance of one of these sections defines a set of routing parameters that are used to choose a remote XOT host to which an X.25 call is to be routed. These sections are headed by a label with the prefix "X25_TO_TCP.".

In its simplest form each route specifies an X.121 address and associated IP address such that if the called address in the X.25 Call Request packet matches the X.121 address then the call is routed to the XOT host at the indicated IP address.

Example:

```
X25_TO_TCP.1
   dte_addr        = "*1"
   host_name       = "host1.mynet.com"
```

The following table summarizes the parameters associated with these routing entries.

Parameter	Type	Description
dte_addr	String	A pattern to match against the called address of the X.25 Call Request packet. See discussion.
host_name	String	The host name or IP address of the destination XOT host.
redirect_address	String	If specified, substitute this address for the called address of the X.25 Call Request packet.
additional_facilities	String	If specified, add these X.25 facilities to the facility field of the Call Request packet.
redirect_facility	Number	If set to 1, and if the call is redirected, add a redirection notification facility that carries the original called address of the X.25 Call Request packet. If set to 0, do not add this facility even if the call is redirected.
restore_redirect_address	Number	If set to 1, and if the facility field of the X.25 Call Request packet contains a redirection notification facility then restore the address from this facility as the called address of the XOT Call Request packet. If set to 0, ignore any redirection notification facility that may be present in the Call Request packet.
reroute	Number	If set to 1, reroute the Call Request packet after performing call redirection. If set to 0, do not reroute even if the address is changed.
a_bit	Number	Allow the use of the X.25 A-bit in the Call Request packet sent to the remote XOT host. See discussion.

Table 8 – X.25 to TCP Routing Parameters

Discussion of Routing from X.25 to TCP

When an X.25 Call Request packet is received it is decoded into its constituent parts: Called address, Calling address, facilities and call user data.

The called address is used to search the list of routes from X.25 to TCP to find a match. When a match is found Gcom_xotd makes a TCP connection to the remote XOT host specified by the `host_addr` parameter and then sends the Call Request packet to the remote host over that connection.

In connecting to the remote XOT host the `host_addr` can specify a host name or an IP address. The TCP connection is always directed to the standard XOT port number at the remote host.

Address Pattern Matching

In performing the match the called address is compared to the pattern specified by the `dte_addr` parameter. The `dte_addr` parameter is specified as a quoted string and may contain decimal digits, the character '?' and the character '*'. The `dte_addr` is matched character for character with the called address in a left to right manner. If the character '?' occurs in the `dte_addr` then that character position matches any single digit in the called address. If the character '*' occurs then that character position matches any sequence of digits in the called address. The '*' character can match a zero length sequence of digits, but the '?' can only match a single digit, which must be present.

Thus, the pattern "`1?3?5`" matches "`12345`" and "`11335`", but not "`22335`" or "`123456`". The last example does not match because there is nothing in the pattern to match the final '`6`'.

Also, the pattern "`1*2`" matches "`12345432`" and "`12`" but does not match "`12345`".

If it is meaningful, '?' and '*' may be intermingled and used multiple times in the address pattern.

The pattern "`*`" matches any address and could be used in a route if there is such a thing as a "default host" in the network.

The first route to match is the one that is selected. The routes are evaluated in the same order as they occur in the configuration file. Thus, it is a good idea to place more particular address patterns before more general ones, with complete wildcard addresses of "`*`" being placed last.

Facilities

A route can specify that certain X.25 facilities be added to the Call Request as it is forwarded to the remote XOT host. Note that RFC 1613 requires that the window

size and packet size facilities be included in the facility field of the forwarded Call Request packet. If the Call Request from X.25 contains those facilities they are forwarded as-is. If not then Gcom_xotd adds these facilities based upon the configured `window_size` and `packet_size` parameters of the X.25 interface over which the Call Request was received.

Call Redirection

Call redirection can be used to establish aliases for remote DTEs. Suppose that you have an X.25 host that uses a set of DTE addresses A, B and C to designate three remote X.25 DTEs. But suppose that, for some reason, the DTE addresses of these hosts change to X, Y and Z, respectively. Call redirection allows you to specify XOT routing parameters that will result in calls placed to address A to be sent, instead, to address X, and similarly for B to Y and C to Z.

In order to cause call redirection to occur simply specify a non-empty string for the `redirect_address` parameter. If this route is selected then the called address will be changed to the `redirect_address`.

The `redirect_address` is actually a pattern, with '?' and '*' characters, not simply a fixed address. So, for example, suppose that you have a route with the following parameters:

```
dte_addr         = "217*"
redirect_address = "1415*"
```

In routing a call with called address "`2173514241`" the redirected called address will be "`14153514241`". That is, the '*' in the `redirect_address` receives the characters that matched the '*' in the `dte_addr`.

If a call is redirected, Gcom_xotd must know whether or not to include a facility that indicates that the call had been redirected and which includes the original called address. The parameter `redirect_facility` controls this. If this parameter is set to `1` then the facility is added to the facilities of the Call Request packet; if it is `0` then it is not. The facility that is added is the Call Redirection Notification facility with code `0xC3`. Octet `1` of the facility value field, which gives a reason for the redirection, is always set to `0x0F` which means "systematic call redirection."

When redirecting a call it is possible that the redirected address might exceed the 15 digits allowed in a standard X.25 called address. However, there is a mechanism, first introduced in X.25 1988, called the A-bit that makes provision for addresses with as many as 255 digits. The parameter `a_bit` can be set to `1` to allow Gcom_xotd to use the A-bit, and longer addresses, if necessary. If you are using call redirection and if you know that there is a Gcom XOT on both ends of the connection then you can enable this option on all your routes. However, if

some other vendor's XOT is at the other end of a route you should not set this parameter.

Finally, once the mechanics of the call redirection are complete, Gcom_xotd either forwards the resulting Call Request packet to the remote XOT host via a TCP connection, or it repeats the routing process using the newly formed called address. The reroute parameter controls this behavior. If the reroute parameter is set to 1 then the call is routed again, otherwise it is forwarded to the remote XOT host.
The following is a sample log listing that shows Gcom_xotd routing an X.25 Call Request using the redirection feature. The route that is being applied in this case looks like the following:

```
dte_addr          = "*1"
redirect_address  = "2222222222*1"
redirect_facility = 1
a_bit             = 1
host_name         = "dave1"

14:24:39.931: route_x25_call:
14:24:39.932: "321" vs "*1" Host=dave1 Match
14:24:39.932: i5:f-1:p-1: replace_remote_address: new 2222222222321
14:24:39.932: i5:f-1:p-1: add_redirect_facil: old 321
```

TCP to X.25 Routing Parameters

Each appearance of one of these sections defines a set of routing parameters that are used to choose a local X.25 port to which an XOT call is to be routed. These sections are headed by a label with the prefix "TCP_TO_X25.".

In its simplest form each route specifies an X.121 address and associated port number such that if the called address in the XOT Call Request packet matches the X.121 address then the call is routed to the indicated X.25 port.

Examples:

```
TCP_TO_X25.X
   dte_addr         = "*2"
   port_number      = 2

TCP_TO_X25.Any
   dte_addr         = "*"
   port_number      = 1
```

This is a pair of simple routes. The first routes any Call Request received from a remote XOT in which the called address ends in the digit '2' to port 2. The second routes any address to port 1.

For a single port XOT GPA this second route is all that is necessary since all incoming calls can only go to port 1 no matter what their called addresses are.

The following table summarizes the parameters that apply to routes going from XOT/TCP to X.25. Note that the parameters are similar to those that parameterize routes for calls going in the opposite direction, except that the routing to X.25 ports has some additional parameterization having to do with load balancing.

Parameter	Type	Description
dte_addr	String	A pattern to match against the called address of the XOT Call Request packet. See discussion.
port_number	String	The port number of the X.25 interface to which the call is to be sent.
redirect_address	String	If specified, substitute this address for the called address of the XOT Call Request packet.
additional_facilities	String	If specified, add these X.25 facilities to the facility field of the Call Request packet.
user_data	String	An ASCII hexadecimal coding of a sequence of bytes to be matched against the call user data field. See discussion.
user_data_mask	String	An ASCII hexadecimal coding of a sequence of bytes to be ANDed with the user_data for matching purposes. See discussion.

`cost`	Number	A number that is multiplied by the number of connections to a given port to calculate the cost of routing an additional call to that port. Set to zero to defeat load balancing. See discussion.
`redirect_facility`	Number	If set to 1, and if the call is redirected, add a redirection notification facility that carries the original called address of the XOT Call Request packet. If set to 0, do not at this facility even if the call is redirected.
`restore_redirect_address`	Number	If set to 1, and if the facility field of the XOT Call Request packet contains a redirection notification facility then restore the address from this facility as the called address of the X.25 Call Request packet. If set to 0, ignore any redirection notification facility that may be present in the Call Request packet.
`reroute`	Number	If set to 1, reroute the Call Request packet after performing call redirection. If set to 0, do not reroute even if the address is changed.
`a_bit`	Number	Allow the use of the X.25 A-bit in the Call Request packet sent to the local X.25 host. See discussion.

Table 9 – TCP to X.25 Routing Parameters

Discussion of Routing from TCP to X.25

When an XOT Call Request packet is received on a TCP connection it is decoded into its constituent parts: Called address, Calling address, facilities and call user data.

The called address is used to search the list of routes from TCP to X.25 to find a match. When a match is found Gcom_xotd sends the Call Request packet to the local X.25 over the indicated port number.

Address Matching

The called address from the XOT Call Request packet is matched against all the `dte_addr` parameters for routes from TCP to X.25. The manner of the address matching is the same as for routing in the opposite direction, from X.25 to TCP and is explained in a previous section.

However, in routing XOT calls to X.25 ports there are two additional pieces of routing criteria that can come into play: the call user data field and load balancing. Because of these additional criteria Gcom_xotd does not select the first route that matches the address pattern. It continues to search the entire route table to evaluate these additional criteria. Judged by these criteria it selects the best match. The definition of “best” is

The first match is considered the best route, initially.

If a route matching all other criteria is found and has a cost that is less than the cost of the best route then the new route becomes the best route.

Thus, if all routes evaluate to the same cost then the best route is the first route that matches the address and user data criteria.

If the `user_data` parameter is set to a non-empty string then an attempt is made to match the call user data field of the XOT packet against the specified value of the `user_data` parameter. The `user_data` parameter consists of an even number of ASCII coded hexadecimal characters, with no blanks, that specifies a sequence of binary byte values. For example, `user_data` = "`210C`" specifies two binary bytes with values 0x21 and 0x0C.

In addition, the `user_data_mask` parameter can be set to specify a sequence of bytes that is logically "anded" with the `user_data` bytes and the call user data bytes prior to comparing them. The form of the `user_data_mask` is the same as for the `user_data` parameter. If the `user_data_mask` parameter specifies fewer bytes than the `user_data` parameter it is logically extended with FF values. If it is longer than the `user_data` parameter then it is truncated to the length of the `user_data` parameter.

The matching consists of comparing the following expressions for each byte 'i' of the call user data field.

```
call-user-data[i] AND user_data_mask[i] vs user_data[i] AND
user_data_mask[i]
```

If the call user data field has fewer bytes than the length of the `user_data` parameter then only those bytes are compared. In the degenerate case, if there are no bytes of call user data, then the user data field is deemed to match the `user_data` parameter.

In addition to matching the DTE address and the user data field, routing from XOT to X.25 is also capable of load balancing between (or among) X.25 ports. When selecting a route to use Gcom_xotd first finds routes that match the address and user data fields. It then evaluates the "cost" of the route and selects the route with the minimum cost.

The "cost" is evaluated in terms of the number of connections that are open to the X.25 port under consideration, plus one on the supposition that this connection is routed to that port. The cost parameter of the route specification is multiplied by the number of connections plus one and the route with the lowest cost is the route that is selected.

This allows for load balancing, on a per connection basis, among multiple X.25 ports.

Call Redirection

Once a route is selected the call redirection parameters of the route are evaluated and acted upon in a manner similar to that which was described above for routes from X.25 to TCP.

The following is an example of routing an XOT packet to an X.25 port. In this case an XOT call arrived addressed to "2222222222321". You can see the comparisons to the various router table entries and the three that matched. Because the cost of all routes in this example was zero, the first route to match is the one that was selected.

```
14:24:39.934: route_xot_call:
14:24:39.934: "2222222222321" vs "445566" Port=1 No-match
14:24:39.934: "2222222222321" vs "445566" Port=2 No-match
14:24:39.935: "2222222222321" vs "445566" Port=1 No-match
14:24:39.935: "2222222222321" vs "889900" Port=2 No-match
14:24:39.935: "2222222222321" vs "7890" Port=1 No-match
14:24:39.935: "2222222222321" vs "1111111111*" Port=1 No-match
14:24:39.935: "2222222222321" vs "2222222222*" Port=2 Match
14:24:39.935: "2222222222321" vs "*1" Port=1 Match
14:24:39.935: "2222222222321" vs "*2" Port=2 No-match
14:24:39.936: "2222222222321" vs "*" Port=1 Match
14:24:39.936: Match: "2222222222*" Port=2
```

PVC Setup Parameters

Each appearance of one of these sections defines a set of setup parameters that are used to establish a Permanent Virtual Circuit (PVC) with a remote XOT host. These sections are headed by a label with the prefix "X25_PVC.".

The gist of these parameters is to specify local and remote LCNs for the PVC as well as the IP address of the remote XOT host. Here is an example that configures a PVC from local port 2 to a port named "lapb_1" on a remote XOT host. The LCN is 100 on both ends of the PVC.

```
X25_PVC.2-100
   port_number           = 2
   lcn                   = 100
   host_name             = "dave1"
   remote_interface_name = "lapb_1"
   remote_lcn            = 100
```

The following table summarizes the PVC setup parameters.

Parameter	Type	Description
port_number	Number	The local port number.
lcn	Number	The local LCN.
remote_lcn	Number	The LCN on the remote XOT host.
in_window	Number	Window size for packets coming in from the local X.25 port.
out_window	Number	Window size for packets going out to the local X.25 port.
in_packet_size	Number	Packet size for packets coming in from the local X.25 port. Should be a power of 2.
out_packet_size	Number	Packet size for packets going out to the local X.25 port. Should be a power of 2.
timeout	Number	Setup timeout in seconds. RFC 1613 recommends 300 (5 minutes).
host_name	String	Host name or IP address of remote XOT host.
remote_interface_name	String	Name of X.25 interface port on remote XOT host.

Table 10 – PVC Setup Parameters

Discussion of PVC Setup

RFC 1613 makes provision for two XOT hosts to negotiate PVC setups. There is a special negotiation packet that is exchanged between the two hosts for each PVC

that needs to be set up. In order for a PVC to be established one needs to know the following items of information.

The local and remote Logical Channel Numbers (`lcn` and `remote_lcn`).

The local X.25 port number (`port_number`).

The IP address of the remote XOT host (`host_name`).

The remote X.25 port number. This is specified not with a number but with an ASCII name for the port. Each type of XOT host will have its own naming convention for its ports, so it is necessary to know what the port names are on the remote XOT host. The Gcom XOT GPA uses the names "`lapb_1`", "`lapb_2`", etc as the port names. The `remote_interface_name` parameter sets the port name for the remote XOT host.

The window sizes for the virtual circuit (`in_window_size` and `out_window_size`).

The packet sizes for the virtual circuit (`in_packet_size` and `out_packet_size`).

When Gcom_xotd initializes itself it opens TCP connections to all remote XOT hosts involved in PVC setups. It opens one TCP connection for each defined PVC. If the TCP connection fails to complete it waits an amount of time given by the timeout parameter for the PVC in question and then tries again.

Once the TCP connection completes the negotiation proceeds. If the remote XOT host agrees to the PVC parameters then Gcom_xotd performs an X.25 reset operation on the local and remote LCNs. When the reset is complete the PVC is ready for data transfer.

If the negotiation fails the TCP connection is closed and is then retried after the configured timeout expires. Of course, if the configuration remains the same on both ends the negotiation is likely never to succeed.

Gcom_xotd can also accept PVC setup negotiations from remote XOT hosts. It receives the negotiation packet and evaluates its parameters. If the parameters in the packet match up with a configured PVC then the negotiation succeeds and PVC is made available for use. If a parameter mismatch occurs the negotiation fails and the PVC will remain unusable.

Gcom_xotd uses the timeout parameter to determine whether the PVC setup procedure is to be active or passive for a given PVC. If the parameter is zero then the passive mode is used; if it is non-zero then the active mode is used.

In the passive mode Gcom_xotd does not initiate a TCP connection and send a PVC setup packet for the PVC. Instead it passively waits for some remote XOT

host to send a PVC setup packet of its own, to which Gcom_xotd then responds.

In active mode Gcom_xotd initiates a TCP connection followed by a PVC setup packet for the PVC. In this mode collision situations are possible when both ends initiate the PVC setup procedure simultaneously using separate TCP connections. In order to resolve these collision situations, Gcom_xotd adds a random amount of time (less than one thousand milli-seconds) to the timeout value when retrying a failed PVC setup procedure.

Statistics Interface

Gcom_xotd provides an interface through which an external program can obtain status, statistics and trace information. The interface also provides for certain control functions as well such as causing Gcom_xotd to terminate and the aborting of selected connections.

The interface is implemented by using the Gcom SyncSocket API to listen on IP port number 8200. An external program that uses this interface uses the SyncSocket API to connect to port 8200 and then send an SS_OP_CONN_REQ. Gcom_xotd responds with an SS_OP_CONN_CONF to establish the statistics session. The external program then sends an SS_OP_DATA with the request, encoded in ASCII, in the data field. Gcom_xotd responds with one or more SS_OP_DATA messages, each of which contains the response information encoded in ASCII. Gcom_xotd then sends an SS_OP_DISC_REQ to terminate the session, and closes the underlying TCP connection. Gcom provides a utility program, Gcom_xot, to perform these functions.

The following ladder diagram illustrates this process.

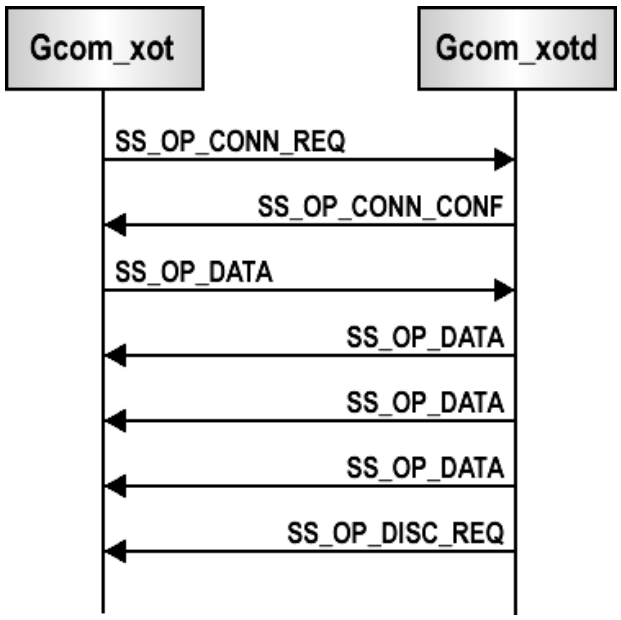

Figure 2 – Gcom Statistics Process

The program Gcom_xot passes command line parameters to the statistics interface of Gcom_xotd and prints the returned ASCII text on the user's terminal screen.

Gcom_xot accepts the following command line parameters.

Parameter	Description
-H*host*	Selects the host name, or IP address, of the machine that is running the Gcom_xotd that is to be interrogated for information. The default is `localhost`.
-h	Print out help text for available options.
-A*n*	Abort connection number *n*.
-d*n*	Set Gcom_xotd's debug mask to *n*.
-E	Causes Gcom_xotd to exit gracefully.
-rt[v]	Display routes from X.25 to TCP. Add the v flag to include a legend explaining what the column headings mean.
-rx[v]	Display routes from TCP to X.25. Add the v flag to include a legend explaining what the column headings mean.
-t*n*	Print out the message trace table for connection number *n*.
-tn*n*	Print out the message trace table for connection number *n* with numeric data instead of decoded data.
-U	Print out connection summary.
-u*n*	Print out connection details for connection number *n*.
-X	Causes Gcom_xotd to exit via abort. If enabled, this will produce a core file.

Table 11 – XOT Statistics Parameters

If you are running Gcom_xot on the same machine that is running Gcom_xotd then you do not have to use the '-H' parameter. Use '-H' to access the Gom_xotd on a distant system.

Parameters -A, -t and -u have "wildcard" forms in which the connection number, n, can be the character '*', usually escaped as '*' so as to avoid shell interpretation. Using this form causes the operation to be performed on all connections instead of a single one. Thus, '-A*' causes all connections to be aborted and '-u*' prints detailed status for each connection.

The action that occurs when a connection is aborted depends upon the connection type. For LAPB connections, these are the X.25 access lines. Aborting a LAPB connection means exchanging X.25 Restart packets with the X.25 host, clearing all SVCs routed to that access line and re-establishing all PVC connections.

Aborting a PVC causes the PVC to be re-established. The X.25 user will perceive an X.25 Reset exchange when this occurs.

Aborting an SVC causes the SVC to be cleared on the X.25 link and the XOT/TCP connection to be closed.

Examples:

Gcom_xot –U

```
Index Type       State          Recv-Cnt Xmit-Cnt Info
    0 XOT_LISTEN TCP-LISTENING     17601    17601 1998
    1 LAPB       READY           5522145  5516037 C-cnt=6
    2 LAPB       READY           5516630  5522022 C-cnt=6
    3 STATS_LSTN TCP-LISTENING         7        0 8200
    5 SVC        DATA_XFER           253      250 192.168.1.186:58003 1/16
    6 PVC        DATA_XFER             1        1 192.168.1.186:40412 1/100
    7 SVC        DATA_XFER           249      253 192.168.1.186:58003 2/1
    8 SVC        DATA_XFER           246      244 192.168.1.186:58004 1/15
    9 SVC        DATA_XFER           244      247 192.168.1.186:58004 2/2
   10 SVC        DATA_XFER           247      245 192.168.1.186:58005 1/14
   11 SVC        DATA_XFER           245      247 192.168.1.186:58005 2/3
   12 SVC        DATA_XFER           245      249 192.168.1.186:58006 1/13
   13 SVC        DATA_XFER           249      245 192.168.1.186:58006 2/4
   14 SVC        DATA_XFER           219      221 192.168.1.186:58007 1/12
   15 SVC        DATA_XFER           221      220 192.168.1.186:58007 2/5
   16 STATS      TCP-CONNECTED         2       48 192.168.1.88:42869
   17 PVC        DATA_XFER             1        1 192.168.1.186:40412 2/100
```

Most of the information shown should be self-explanatory. The “Info” column varies depending on the type of connection. For listening connections it shows the port number being listened on. For LAPB connections it shows the connection count for that X.25 access line. This is the total number of PVCs and SVCs assigned to that line.

For PVCs and SVCs the Info column shows the remote IP address and port number of the other end of the XOT connection. The notation port/lcn at the end of the line shows the assigned X.25 port number and logical channel number.

Gcom_xot –t15

```
Trace for index 15 Type=SVC State=DATA_XFER
Time-Stamp    Type     Cnt Pkt-Type   LCN
19:11:51.913: XOT_IN    26 Call Req   12    10 0c 0b 0d 22 22 22 22 22 32 10 0e
19:11:52.000: XOT_OUT   13 Call Acpt  5     10 05 0f 00 08 02 aa 42 08 08 43 02
19:11:53.151: XOT_IN    43 Data(0,0)  12    10 0c 00 00 00 00 09 00 00 00 00 01
19:11:53.192: XOT_OUT   43 Data(0,0)  5     10 05 00 00 00 00 09 00 00 00 00 01
19:11:53.215: XOT_IN    43 Data(0,1)  12    10 0c 02 00 00 00 09 00 00 00 01 01
19:11:53.224: XOT_OUT    3 RR(2)      5     10 05 41
19:11:53.238: XOT_OUT   43 Data(2,1)  5     10 05 42 00 00 00 09 00 00 00 01 01
19:11:53.283: XOT_IN     3 RR(2)      12    10 0c 41
```

In the trace table printout, as with the log files, all time stamps are in GMT. Packets that contain P(R) and P(S) fields are decoded so that you can easily track the values of these fields. For Data Packets the notation is (Pr,Ps).

Note that when tracing an XOT type of connection (SVC in this case) the LCN field can vary. This is an artifact of the XOT protocol in which the LCN field is not meaningful in XOT packets that flow between XOT hosts. Each end of an

XOT connection fills in the proper LCN prior to delivering the packet to its assigned interface.

Gcom_xot –t1

```
Trace for index 1 Type=LAPB State=READY
Time-Stamp    Type     Cnt Pkt-Type   LCN
19:22:38.720: LAPB_OUT  43 Data(0,7)  12    10 0c 0e 00 00 00 09 00 00 00 7f 01
19:22:38.720: LAPB_IN    3 RR(1)      14    10 0e 21
19:22:38.724: LAPB_OUT   3 RR(1)      13    10 0d 21
19:22:38.727: LAPB_IN   43 Data(1,1)  14    10 0e 22 00 00 00 07 00 00 00 89 01
19:22:38.746: LAPB_OUT  43 Data(3,2)  15    10 0f 64 00 00 00 06 00 00 00 8a 01
19:22:38.746: LAPB_IN    3 RR(0)      12    10 0c 01
19:22:38.746: LAPB_IN   43 Data(0,0)  12    10 0c 00 00 00 00 09 00 00 00 80 01
19:22:38.747: LAPB_OUT   3 RR(2)      16    10 10 41
19:22:38.755: LAPB_IN    3 RR(3)      15    10 0f 61
19:22:38.763: LAPB_OUT  43 Data(1,0)  13    10 0d 20 00 00 00 08 00 00 00 88 01
19:22:38.768: LAPB_IN   43 Data(3,3)  15    10 0f 66 00 00 00 06 00 00 00 8b 01
```

When tracing a LAPB connection the LCN numbers are the true logical channel numbers of the packets that flow across the X.25 access line.