nifty

a Network Traffic Flow Analyser

Version 4.1

Nevil Brownlee

Information Technology Systems & Services
The University of Auckland
Auckland, New Zealand

November 1997

The information which nifty uses and displays is generated by a NeTraMet traffic meter; readers should consult the NeTraMet documentation for definitions and explanations of the terminology used here.

1. Introduction

nifty is a network traffic flow analyser, i.e. a program which monitors the network segment to which a NeTraMet traffic meter is attached and displays information to help you understand how the traffic is flowing on that network.

1.1. Overview

The traffic flow data which nifty analyses is gathered by a network traffic meter, i.e. by a copy of NeTraMet running on a DOS (PC) or Unix host attached to the network segment of interest. In terms of the Realtime Traffic Flow Measurement (RTFM) architecture, nifty is a combined meter reader, manager and analysis application.

nifty is an X/Motif program. This means that it runs on a Unix system which supports X Windows and Motif, and produces a display on the nifty user's hosts system. For example, I have built and tested the nifty program on Irix (Silicon Graphics) and Solaris (Sun) Unix systems; to access the Unix system I normally use X-Win32, an X Windows client on a PC running Microsoft Windows 95.

nifty displays a log-log plot which is updated after each 'sample,' i.e. each time the NeTraMet meter is read. Flow duration is plotted on the abscissa (X axis). The meter records the times the first and last packets of every flow were seen, so flow duration can be determined with centisecond (SNMP TimeTick) resolution.

The flows to be observed by the meter are specified in a *rule set,* nifty reads this from a *rule file* and downloads its rules to the meter.

A variety of different plots can be generated - the user chooses the required plot using a simple menu interface. For each sample nifty chooses a number of the busiest flows, and plots them using their *FlowKind* attribute value as a plot symbol.

2. Flow Analysis

2.1. Data collection

Nifty collects data from a NeTraMet meter, which makes it similar in function to NeMaC, NeTraMet's manager and collector. In this manual I often refer to 'samples.' A 'sample' is the collection of flow data read from the meter during a data collection. The interval (in seconds) between collections (the 'sample interval') is specified by nifty's -c command-line option.

When nifty starts up it may download a rule set to the meter. This rule set specifies which flows are of interest, and which attributes will be reported on the 'format line.' The 'format line' describes the information to be displayed for a flow being pointed at by the cursor when the left mouse button is pressed.

In principle any rule set could be used by nifty but nifty does impose one restriction, as follows. nifty uses the *FlowKind* attribute for each flow as the symbol to be used when the flow appears on a plot. This can be any ASCII character (enclosed in apostrophes), or one of the following special plot symbols:

0 = dot
1 = diamond
2 = plus
3 = square

A sample rule file, `rules.x_ip,` is provided for use with nifty. A listing of this rule set appears an appendix to this manual. It specifies that the following attributes for IP flows are of interest:

Protocol type (UDP, IP, ..)
SourcePeerAddress and DestPeerAddress
TransType (port number)
SourceTransAddress and DestTransAddress

Non-IP flows are also collected, but they are simply aggregated by protocol type, e.g. all Novell packets produce a single 'Novell' flow.

The format line displays the attributes listed above. Note that the 'count' attributes for the flow are commented out of the format line, as indicated below

```
  # FirstTime LastTime
  # ToPDUs ToOctets "  " FromPDUs FromOctets "   "
  # FlowRuleSet FlowIndex  " | "
```

These attributes are *always* read by nifty for *every* flow which was active during a sample interval. Their values are used to determine which flows are to be plotted.

rules.x_ip also specifies (using *Pushto* rule actions) the plot symbol for each flow. For IP flows rules.x_ip selects a well-known port as the destination, and sets a suitable plot symbol. For example, FTP flows use 'F', nntp flows use 'N', WWW flows use 'W,' etc. Flows which have no well-known ports use a diamond. Clicking on plotted diamonds will display the format line, so you can see what the actual port numbers are. Non-IP flows are plotted as a square. For these the format line will show the Ethernet packet type; see the NeTraMet manual for details of its 'other' protocol feature.

2.2. Selecting the samples

Once the rule set has been downloaded and is being used by the meter, nifty can collect samples of flow data. Each sample includes data for every flow which was active during the sample interval, i.e. every flow for which the meter observed one or more packets.

The meter stores its flow data in a 'flow table,' the maximum size of which is specified by a command-line parameter when the meter is started up. nifty reads this maximum size from the meter, and initialises its own table with the same number of flows.

The nifty table is similar to that on the meter, except that it includes extra attributes which hold the number of packets and bytes in each direction for the last sample collected for this flow. These are similar to the 'rate' attributes computed by the *fd_filter* program.

nifty maintains one further attribute for each flow: the number of samples since the flow was last active. A flow will remain in nifty's flow table until the meter re-uses its memory; the time required for this to happen will depend on how active the network is. The 'last active' attribute is used to select the colour used to plot a flow. The colour range contains eight colours:

black, navy, blue, green, orange, red, magenta, purple.

Black is used for the flows active during the most recent sample, navy for the second-to-last sample, and so on down to purple for flows last active eight or more samples ago. This makes it easy to see which are the active flows on a plot, as well as allowing those which have recently been active to be easily discerned.

When selecting which samples should be searched for flows to plot, nifty offers three choices. These are:

Last sample	Points are selected from the last sample only.
Recent samples	Points are selected from the last eight samples
All samples	Points are selected from all samples, i.e. the whole of nifty's flow table

I find the *recent samples* plots most useful, since they allow me time to look at (and think about) flows which appear only briefly, and also allow inactive flows to disappear from the plots after a reasonable time (eight sample intervals).

2.3. Which flows are 'interesting?'

nifty aims to analyse the traffic flows observed by the meter, and to produce plots which will allow a user to discern the ones which are 'interesting.' This begs the question, "which flows are interesting?" So far three classes of 'interesting' flows have been suggested to me:

a. Flows between pairs of non-well-known ports. The most common cause of these are passive FTP transfers, where a remote FTP server specifies the port which a local FTP client should open to retrieve data.

b. Long-term, high-volume flows. These are interesting because they tie up capacity on links, reducing the capacity available to other users. As an initial assumption, any flow which continues for more than about 20 minutes is worth investigating.

c. Short-term, 'bursty' flows. These are interesting because they can flood router input queues, causing packets for other flows to be discarded (and later re-sent). They are hard to detect because they only appear briefly. Bursts of a few thousand packets in one second can be generated by some programs. These are usually just badly configured; if they persist, they should be investigated.

If you have further suggestions for 'interesting' flows, please send them to me, or to the RTFM WG mailing list (addresses at the end of this manual).

2.4. nifty plots

The nifty plots are designed to be interesting in themselves, and to help in spotting your 'interesting' flows. Type a) flows (non-well-known ports) are plotted using a small diamond. You should develop a rule file which specifies suitable ASCII characters for all the well-known ports which are 'uninteresting' for your network.

When specifying a plot via the menu options, you may specify the ordinate (packets or bytes) and the metric (rate, count or percent). This provides six different types of plot:

Packet Rate	Number of packets per second, estimated from the counts and times observed for the last sample. *This plot is most useful for observing type c) flows (short bursts.)*
Packet Count	Total number of packets seen.
Packet Percent	Flow packet rate as a percentage of total observed packet rate for the last sample.
Byte Rate	Number of bytes per second, estimated from the counts and times observed for the last sample.
Byte Count	Total number of bytes seen. *This plot is most useful for observing type a) flows (long term, high volume).*
Byte Percent	Flow byte count as a percentage of total observed bytes for the last sample.

Note that in all the above plots the rates and counts include all the flow's packets, i.e. they are the sum of the 'to' packets (source-to-destination) and the 'from' packets (destination-to-source). It is often interesting to see the 'from' and 'to' counts separately; this can be done for individual flows using mouse buttons 2 and 3 (see below for details).

3. nifty user's guide

3.1. Installation

Since nifty is an X/Motif program it requires the X and Motif libraries for compilation and execution. The NeTraMet distribution file - NeTraMet.tar.gz - provides separate directory trees for various operating systems, e.g. sg for Irix, solaris for Solaris, etc. Each subdirectory contains a make file; nifty is built by the Make files in the manager/ subdirectories. Before building nifty you should inspect the Make file, and make sure that the X and Motif library locations (compile and execute time) are correct for your system.

A copy of nifty is included in the Irix and Solaris object files (Irix.tar.gz and Solaris.tar.gz); these were created using the distribution make files.

With the version 4.1 release, a new directory tree - autoconf/ - is provided, allowing you to build all of the NeTraMet programs using GNU autoconfigure. Instructions for doing this are given in the the autoconf/INSTALL file.

If you are already using X applications on your workstation, nifty should run without further effort. If not, you will need to set up an X environment - you should seek help from your system support staff to do this.

Like NeMaC, nifty is an SNMPv2 client. It opens a UDP port so as to connect to the NeTraMet meter, and it doesn't need any special privileges to do this. It does, however, need access to a copy of the meter MIB; it is simplest to provide this by setting the MIBTXT variable to indicate the location of the MIB file. See the NeTraMet manual for details of this.

3.2. Command line options

nifty's command line options are specified as usual, i.e. each option starts with a hyphen, a letter indicating the options, then any parameters required by the option.

The options most important for nifty are:

-c nnn	Specifies the required collection interval in seconds. When choosing a collection interval, remember that you need time to examine each plot - a minimum of 60s seems sensible.
-r fff	Tells nifty to download the rule set from file fff, and to use that format statement in that file to generate the 'format line' displayed when button 1 is pressed.
-L fff	Specifies that nifty should write its log (including records selected by the 'logging' menu options) to the file named fff.
-n nnn	Tells nifty that nnn flows should be plotted for each sample. The default is 30.

The remaining options have the same meaning that they do for NeMaC; see the NeTraMet/NeMaC manual for further details.

-l	Lists nifty's rule set while it is being downloaded.
-s	Indicates that the rule set file is to be checked for syntax, but not downloaded to the meter. nifty exits after performing the syntax check.
-a sss	Indicates the lag required for samples. Collections occur sss seconds after the 'synchronised' times, e.g. -c60 -a5 would request nifty to collect flow data at 5 seconds after every minute.

-u	Indicates that collections are to be 'unsynchronised,' i.e. they will be performed immediately after the rule set is downloaded, and at intervals specified by the -c option thereafter. The default is 'synchronised.'
-g sss	Specifies the NeTraMet meter's garbage collection interval.
-m pppp	Specifies the UDP port to use for communication with the NeTraMet meter. By default this is port 161 (SNMP).
-h pp	Specifies the NeTraMet meter's high water mark.
-i sss	Specifies the NeTraMet meter's inactivity time.
-o pp	Specifies the NeTraMet meter's flood mark.

Following the options, the name of a meter and its write SNMP community should appear on the command line. nifty must use a meter's write community, because it sets the meter's *LastCollectTime* variable to tell it that a collection has been started. Failure to use the write community will not prevent nifty from getting started, but it will prevent the meter from recovering memory space from inactive flows!

For example

```
    nifty -c120 -r rules.x_ip 130.216.234.237 test
```

would cause nifty to begin analysing flow data from meter 130.216.234.237 with write SNMP community 'test'. The rule file 'rules.x_ip' would be read and downloaded to the meter, and that meter's flow data would be collected every two minutes and used to generate a plot.

From version 4.1, the NeTraMet meter is able to run more than one rule set at the same time. For example, you can run a 'nifty' rule set while another 'daily logging' rule set continues to run normally. The meter uses 'Owner Names' to help distinguish its rule sets. You can specify an Owner Name for nifty by specifying it on the command line, after the write community name, e.g.

```
    nifty -c120 -r rules.x_ip 130.216.234.237 test Net-Ops
```

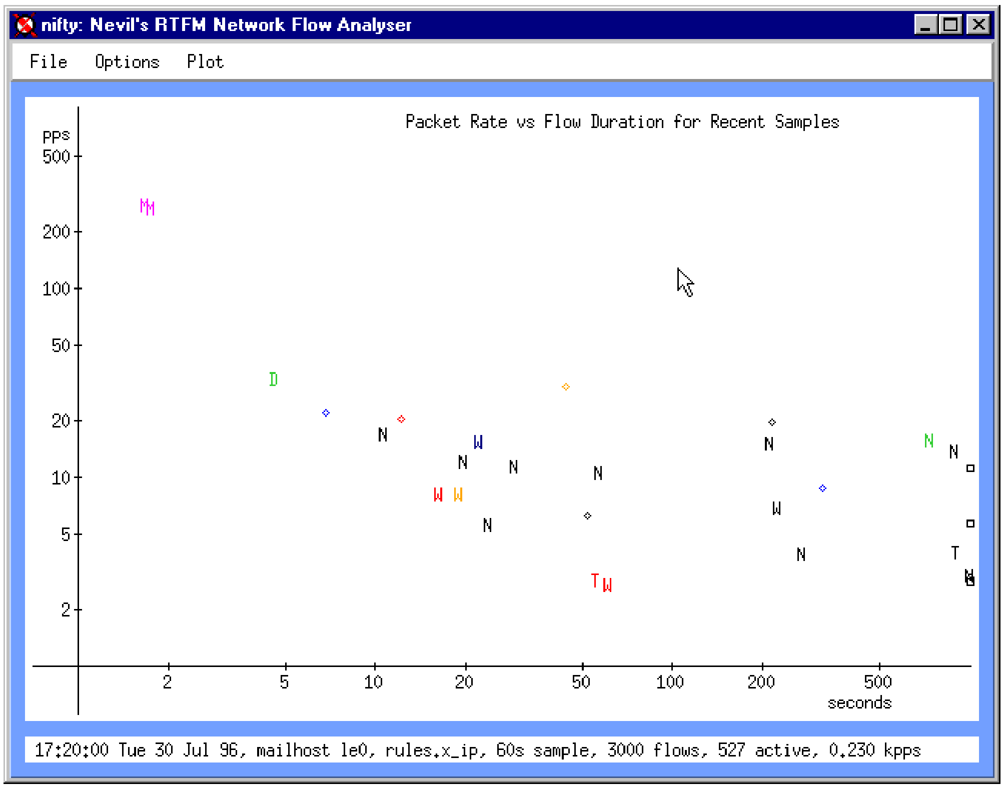
The Owner Name is an alphameric string with a maximum length of 16 characters. It may contain any characters except a blank. In the example above we used `Net-Ops` for nifty's Owner Name. If an Owner Name is not specified, 'nifty' is used.

When nifty execution terminates normally (using the 'exit' menu item, closing nifty's window, or by interrupting nifty using the Control-C key), nifty will stop its rule set from executing, and delete it from the meter. This leaves the meter continuing to run its other rule sets, with no trace remaining of nifty's rule sets or the flows it measured.

If two users wish to run nifty at the same time, they need to agree to use different Owner Names, otherwise both might use 'Nifty' by default, which is bound to cause confusion!

3.3. nifty display

Once nifty is running, it displays a window as shown below. This is a standard X/Motif window (with its name at the top, a go-away box, etc), a menu bar, a plot area and a status line. These are described in the following sections.


nifty: Nevil's RTFM Network Flow Analyser
File
Options
Plot
Packet Rate vs Flow Duration for Recent Samples
pps
500
200
100
50
20
10
5
2
2
5
10
20
50
100
200
500
seconds

17:20:00 Tue 30 Jul 96, mailhost le0, rules.x_ip, 60s sample, 3000 flows, 527 active, 0.230 kpps

3.3.1. Menu bar

The menu bar appears at the top of nifty's display. It provides access to a tree of menus which allow you to specify exactly what you want nifty to do. The menu options are described in detail in the next chapter (nifty options).

nifty's default (startup) settings are:

- no logging
- display packet rate packet rate for recent samples
- axes 900 pps vs 15 minutes
- display IP addresses in numeric (dotted quad) form

Note that when you change the requested display, e.g. from packet rate to total bytes, the axis scales are not changed automatically. Instead you must change them (using the 'plot' menu) to produce a sensible display. Points which lie off the axes are plotted at the top or bottom of the display.

From time to time - at the specified collection intervals - nifty pauses to read flow data from the meter. During these collections nifty displays a 'wrist watch' cursor to indicate that it is busy, and will not respond to mouse clicks. *I have found that mouse clicks at such times can often cause nifty to crash on my system. One way to minimise this problem is to use a fairly long collection interval, say 120 seconds; this gives plenty of time to look at the screen, click on any 'interesting' flows, etc.*

3.3.2. Plot area

The central part of nifty's display is the plot area, where the plots specified by the menu options are displayed, and refreshed after each data collection. The title of the plot appears in the upper right area of the plot; it clearly indicates exactly what is being plotted. The ordinate (Y axis) units appear at the top left of the screen, and the abscissa (X axis) units appear at the lower right.

3.3.3. Status line

The lower part of nifty's display is a status line, which is used to display information to the user, as follows:

- When the display is initialised, the status shows version information
- After each data collection, a summary of that sample is shown
- After a mouse click in the plot area, the information about the flow nearest the cursor is shown (details are given in the next section)

3.4. Mouse Buttons

Nifty uses a normal X/Motif window, which responds to the ordinary 'window housekeeping' actions. For example, clicking on the 'go away' box will terminate nifty, and dragging on the bottom right-hand corner of the window will resize it.

Once nifty is running you can use the mouse to move the on-screen (arrow) cursor so that it points to a flow on the display. You may then click one of the three mouse buttons; nifty will determine which flow is closest to the cursor, then display some information about it on the status line. There are three possibilities:

Button 1 (left)	*co-ordinates and format line:.* the flow's co-ordinates use the same units as the axes; the contents of the format line are specified in the rule set which nifty is using.
Button 2 (centre)	*packets:* total number of forward and backward packets seen for the flow, and the number of forward and backward packets seen for the last sample of the flow.
Button 3 (right)	*bytes:* total number of forward and backward bytes seen for the flow, and the number of forward and backward bytes seen for the last sample of the flow.

4. nifty menus

4.1. File

4.1.1. Logging

·***Sample***	Log records are written each time nifty reads flow data from the NeTraMet meter
·***Points***	A log record is written whenever the user clicks Button 1 (the left button) on a displayed data point. This record contains the co-ordinates of the point and the flow data as specified by the rule set's FORMAT statement
·***None***	No log records are written by nifty

4.1.2. Peer Address

·***IP Address***	IP addresses are displayed as numbers, i.e. as four integers separated by dots
·***Domain Name***	IP addresses are displayed as domain names; nifty looks them up in the DNS

4.1.3. Quit

	Shuts nifty down. Equivalent to clicking the 'go-away' box on nifty's Window

4.2. Options

4.2.1. Ordinate

·***Bytes***	Use bytes when selecting flows to plot; the Y axis will be used for byte counts or bit rates (bps)
·***Packets***	Use packets when selecting flows to plot; the Y axis will be used for packet counts or packet rates (pps)

4.2.2. Metric

·***Rate***	Display plots showing packet rate (pps) or bit rate (bps) for flows
·***Count***	Display plots showing total packet (pkt) or byte (MB) counts for flows
·***Percent***	Display plots showing the percentage contribution of selected flows to the total packet or byte counts

4.2.3. Selection

·***Last sample***	No log records are written by nifty
·***Recent samples***	No log records are written by nifty
·***All samples***	No log records are written by nifty

4.3. Plot

.3.1. X axis

·***100 s***	Sets X scale 1 to 100 seconds
·***15 m***	Sets X scale 1 to 900 seconds
·***2 h***	Sets X scale 1 to 120 minutes

4.3.2. Y axis

·***40***	Sets Y scale 0.1 to 40
·***900***	Sets Y scale 1 to 900
·***9k***	Sets Y scale 100 to 9,000
·***90k***	Sets Y scale 1,000 to 90,000
·***900k***	Sets Y scale 10,000 to 900,000
·***9M***	Sets Y scale 100,000 to 10,000,000
·***90M***	Sets Y scale 1,000,000 to 90,000,000

5. Appendix: *rules.x_ip* rule set

```
#  1200, Sun 26 May 96
#
#  Rules to look at IP packets, pushing all flow attributes
#
#  Nevil Brownlee,  ITSS Technology Development,  The University of Auckland
#
```

```
SET 2
#
RULES
  SourcePeerType & 255 = dummy:  Ignore, 0;
  SourcePeerType & 255 = IP:     PushtoAct, IP_pkt;
  SourcePeerType & 255 = Other:  PushToAct, other_pkt;
#
  Null & 0 = 0:   GotoAct, Next;  # Not IP or Other
  FlowKind & 255 = 3:  PushtoAct, Next;  # Plot as SQUARE
  SourcePeerType  & 255 = 0:  PushPkttoAct, Next;
  SourceInterface & 255 = 0:  CountPkt, 0;
#
other_pkt:  # We want to know ethertype/LSAP (in source/dest Peer)
  FlowKind & 255 = 3:  PushtoAct, Next;  # Plot as SQUARE
  SourcePeerAddress  & 255.255 = 0: PushPktToAct, Next;
  DestPeerAddress    & 255.255 = 0: CountPkt, 0;
#
IP_pkt:
  SourceTransType & 255 = tcp:    Pushto, tcp_udp;
  SourceTransType & 255 = udp:    Pushto, tcp_udp;
  Null & 0 = 0:   GotoAct, Next;  # Not TCP or UDP
  SourceTransType  & 255 = 0:  PushPkttoAct, Next;
  FlowKind & 255 = 3:  PushtoAct, count_IP;  # Plot as SQUARE
#
tcp_udp:
  SourceTransAddress & 255.255 = domain:   Retry, 0;  # Want WKP as dest
  SourceTransAddress & 255.255 = 79:       Retry, 0;
  SourceTransAddress & 255.255 = ftp:      Retry, 0;
  SourceTransAddress & 255.255 = ftpdata:  Retry, 0;
  SourceTransAddress & 255.255 = gopher:   Retry, 0;
  SourceTransAddress & 255.255 = 113:      Retry, 0;
  SourceTransAddress & 255.255 = 513:      Retry, 0;
  SourceTransAddress & 255.255 = 138:      Retry, 0;
  SourceTransAddress & 255.255 = nntp:     Retry, 0;
  SourceTransAddress & 255.255 = 2049:     Retry, 0;
  SourceTransAddress & 255.255 = ntp:      Retry, 0;
  SourceTransAddress & 255.255 = 110:      Retry, 0;
  SourceTransAddress & 255.255 = 515:      Retry, 0;
  SourceTransAddress & 255.255 = smtp:     Retry, 0;
  SourceTransAddress & 255.255 = snmp:     Retry, 0;
  SourceTransAddress & 255.255 = 1080:     Retry, 0;  # UA socks gateway
  SourceTransAddress & 255.255 = telnet:   Retry, 0;
  SourceTransAddress & 255.255 = www:      Retry, 0;
  SourceTransAddress & 255.255 = 8080:     Retry, 0;  # UA WWW proxy
  SourceTransAddress & 255.255 = 6000:     Retry, 0;
#
  DestTransAddress & 255.255 = domain:     GotoAct, c_domain;
  DestTransAddress & 255.255 = 79:         GotoAct, c_finger;
  DestTransAddress & 255.255 = ftp:        GotoAct, c_ftp;
  DestTransAddress & 255.255 = ftpdata:    GotoAct, c_ftpdata;
  DestTransAddress & 255.255 = gopher:     GotoAct, c_gopher;
  DestTransAddress & 255.255 = 113:        GotoAct, c_imap;
  DestTransAddress & 255.255 = 513:        GotoAct, c_login;
  DestTransAddress & 255.255 = 138:        GotoAct, c_netbios;
  DestTransAddress & 255.255 = nntp:       GotoAct, c_news;
  DestTransAddress & 255.255 = 2049:       GotoAct, c_nfs;
  DestTransAddress & 255.255 = ntp:        GotoAct, c_ntp;
  DestTransAddress & 255.255 = 110:        GotoAct, c_pop;
  DestTransAddress & 255.255 = 515:        GotoAct, c_printer;
  DestTransAddress & 255.255 = smtp:       GotoAct, c_smtp;
  DestTransAddress & 255.255 = snmp:       GotoAct, c_snmp;
  DestTransAddress & 255.255 = 1080:       GotoAct, c_socks;  # UA socks
  DestTransAddress & 255.255 = telnet:     GotoAct, c_telnet;
  DestTransAddress & 255.255 = www:        GotoAct, c_www;
  DestTransAddress & 255.255 = 8080:       GotoAct, c_www;  # UA WWW proxy
  DestTransAddress & 255.255 = 6000:       GotoAct, c_xwin;
#
  Null & 0 = 0:  GotoAct, c_tcp_udp;  #  'Unusual' port
#
c_domain:
  FlowKind & 255 = 'D':  PushtoAct, count_IP;
```

```
c_ftp:
c_ftpdata:
  FlowKind & 255 = 'F':  PushtoAct, count_IP;
c_imap:
  FlowKind & 255 = 'I':  PushtoAct, count_IP;
c_news:
  FlowKind & 255 = 'N':  PushtoAct, count_IP;
c_pop:
  FlowKind & 255 = 'P':  PushtoAct, count_IP;
c_smtp:
  FlowKind & 255 = 'M':  PushtoAct, count_IP;
c_socks:
  FlowKind & 255 = 'S':  PushtoAct, count_IP;
c_telnet:
  FlowKind & 255 = 'T':  PushtoAct, count_IP;
 c_www:
  FlowKind & 255 = 'W':  PushtoAct, count_IP;
c_xwin
  FlowKind & 255 = 'X':  PushtoAct, count_IP;
#
c_finger:
c_gopher:
c_login:
c_netbios:
c_nfs
c_ntp:
c_printer:
c_snmp:
#
c_tcp_udp:
  Null & 0 = 0:   Goto, Next;  # Not a well-known TCP or UDP port
  SourceTransType & 255 = tcp:  GotoAct, c_tcp;
  Null & 0 = 0:   GotoAct, c_udp;
c_udp:
  FlowKind & 255 = 2:  PushtoAct, count_IP;  # Plot as PLUS
c_tcp:
  FlowKind & 255 = 1:  PushtoAct, count_IP;  # Plot as DIAMOND
#
count_IP:
  SourceInterface & 255 = 0:  PushPkttoAct, Next;
  SourcePeerAddress  & 255.255.255.255 = 0:  PushPkttoAct, Next;
  DestPeerAddress    & 255.255.255.255 = 0:  PushPkttoAct, Next;
  SourceTransAddress & 255.255         = 0:  PushPkttoAct, Next;
  DestTransAddress   & 255.255         = 0:  CountPkt, 0;
#
#
FORMAT
# FirstTime LastTime
# ToPDUs ToOctets "  " FromPDUs FromOctets "   "
# FlowRuleSet FlowIndex  " | "
  SourcePeerType SourcePeerAddress " -> " DestPeerAddress "  "
  SourceTransType SourceTransAddress " -> " DestTransAddress;
#
#
STATISTICS
#
# end of file
```

6. Author's Address

Please send any comments, suggestions, bug reports to me, Nevil Brownlee, i.e.

n.brownlee@auckland.ac.nz

Ideas for 'interesting' flows, new kinds of plot, etc, etc. should be posted to the RTFM Working Group's mailing list,

rtfm@auckland.ac.nz