

# Using NeTraMet for Production Traffic Measurement

N. Brownlee  
The University of Auckland  
Private Bag 92019  
Auckland  
New Zealand \*

June 21, 2002

## Abstract

When monitoring network traffic, a flow measurement approach can provide the advantage of data reduction in near real-time. RTFM, the IETF's standard, generalised architecture for measuring traffic flows, and NeTraMet, an open-source implementation of that architecture, are introduced. The results of a NeTraMet usage survey are presented, together with a detailed description of the NeTraMet components and the ways they can be used to construct traffic flow measurement systems for any particular network. Nifty, an X Window near-real-time traffic flow analyser, is also described. Experiences with NeTraMet are summarised, highlighting its usefulness in building customised flow measurement systems.

## 1 Introduction

### 1.1 Traffic Monitoring: Packets and Flows

Network operators are very interested in understanding the ways in which traffic flows through their networks. Traffic information is vital for trouble shooting (detecting unusual behaviours), for collecting usage data (logging, security incident detection, billing) and for capacity planning.

One possible approach to traffic measurement is to write copies of packets (or perhaps just their headers) to a 'trace' file on disk, using a tool such as tcpdump [1]. Once a trace file has been collected it can be analysed off-line; analysis can include searching for correlations between traces collected at different measurement points.

Packet traces have two main advantages. First, since they contain data for every packet, they can be analysed in different ways, yielding insight into different aspects

---

\*Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

of traffic behaviour. Second, they can be collected at very high speeds, usually limited only by disk writing speed. A good example of this approach is Feldmann's analysis of HTTP behaviour [2], using trace data collected using the PacketScope tool. Researchers interested in network behaviour can access public archives of header traces, e.g. [3]. The disadvantage of trace files is that they can easily be very large, simply because they record individual packets.

A second approach to traffic measurement is to record 'flows' instead of packets. The notion of traffic flows has been used for many years, but flows can be defined in several different ways. In routing, for example, a flow is a sequence of packets going from one IP host to another, is identified only by its source and destination IP addresses, and is unidirectional. Researchers often identify flows at a finer granularity. In this context flows are identified by a 5-tuple, i.e. (protocol, source and destination addresses, source and destination port numbers) - again they are unidirectional.

A flow 'meter' observes packets and builds tables of flow information, recording each flow's address as a 5-tuple, and counting its packets and bytes. An obvious point at which to measure flows is within routers and switches; Cisco's NetFlow [4] and Cabletron's LFAP [5] are two well-known examples of this. Several open-source tools are available to collect and analyse NetFlow data, such as cflowd [6] and NeTraMet [7].

NetFlow data provides a good view of traffic through a router or switch, but its flows are limited to unidirectional 5-tuples. In contrast, the RTFM architecture (described below) provides a very general way for a user to specify which flows are to be measured. Furthermore, RTFM flows are bi-directional, having byte and packet counters for each direction.

RMON, the IETF's remote network monitoring system, provides a third way to measure network traffic. An RMON agent (usually built into a switch or router) implements the RMON MIB [8], allowing a network manager to determine traffic levels in network segments, total traffic loads to/from busy hosts and traffic loads between host pairs, for different protocols, and so on. RMON, however, does not provide any flow measurement capability.

## 1.2 The IETF, RTFM and NeTraMet

The Internet Engineering Task Force (IETF) [9] is the organisation which produces technical standards for the Internet, published as RFCs (Requests for Comment). Within the IETF there are many active Working Groups, each pursuing standards activity in their area of interest. Working Groups are set up as the need arises; they complete their work, publish their RFCs, then become dormant.

In 1995 the Realtime Traffic Flow Measurement (RTFM) Working Group [10] was chartered to produce an architecture for traffic flow measurement. The architecture defines three network entities:

**Meters** gather data about packets, and consolidate this to provide flow data. Flows are identified by their end-point attributes, and are bi-directional.

**Meter Readers** retrieve flow data from meters using SNMP.

**Managers** co-ordinate the activities of meters and meter readers.

In October 1999 the Working Group published RFCs 2720-2724, documenting the RTFM architecture. RFC 2720 - the Traffic Meter MIB [11] - is a Proposed Internet Standard.

NeTraMet is the first implementation of the RTFM architecture. It is a toolkit for measuring traffic flows, including:

- meters (NeTraMet, NetFlowMet)
- combined manager / meter readers (NeMaC, nifty)
- a compiler to help produce meter configuration files (srl)
- flow data utility programs (fd\_filter, fd\_extract)

NeTraMet has been distributed as open-source software since 1993, under the terms of the Gnu Public License (GPL). At present NeTraMet is the only open-source implementation of RTFM. It is in widespread use, with about 280 people on the NeTraMet mailing list

## 2 What do people use NeTraMet for?

An email survey was sent to the NeTraMet mailing list in August 2000. Twenty-three completed responses were returned within ten days, i.e. about 8%. This is a fairly low response rate, but it did provide an interesting insight into the NeTraMet user community.

Twenty-one of the 23 respondents were currently using NeTraMet. Of these, 80% use it for production traffic measurement (i.e. for collecting network management or billing data), and 50% use it for research purposes. The respondents described their sites as indicated in table 1:

User Type	% of responses
ISP	33%
Research/Education Backbone	14%
Enterprise	10%
University	38%

Table 1: NeTraMet user Groups

Although NeTraMet is most commonly used to meter 10/100 Mbps Ethernet links, there are several sites using it with DS3, OC3 and OC12 links. At the high-speed sites, average traffic rates above 50 Mbps were being metered.

A single RTFM manager can control many meters, and many of the responding sites do this. The number of meters being by each respondent is shown on table 2:

Number of meters	% of responses
1	24%
2	43%
3-10	14%
more than 10	5%

Table 2: Number of Meters

The above percentages don't sum to 100%. This is because four of the sites are also collecting packet data from Cisco routers using NetFlow data - they use NeTraMet to aggregate and reduce NetFlow data from up to ten Cisco routers.

A vital task in using NeTraMet is preparing 'rulesets,' i.e. meter configuration files. 78% of the respondents use SRL, the Simple Ruleset Language [12], to create rulesets, which seems a low percentage. 62% of them, however, create rulesets 'by hand,' i.e. by editing the sample rulesets supplied with NeTraMet. It seems that the responses are biased towards long-time NeTraMet users, who created rulesets before the SRL compiler was written, and who are still using them. Even more interesting, because it was completely unexpected, were the two users who said that they have produced their own table-driven programs to create rulesets!

When it comes to reducing and analysing the flow data collected by meter readers, all respondents have developed their own collection of perl scripts, cron jobs, etc., and nearly 50% of them said they store their flow data in some kind of database (rather than in a collection of flat files).

Since the packet and byte counts in a flow data file come from an SNMP agent - an RTFM meter - they are 64-bit SNMP counters, which means that they are never reset, and can wrap around. NeTraMet provides a utility program called `fd_filter` which will compute differences between successive meter readings, but only 38% of the respondents said they are using `fd_filter`. This implies that the other 62% have written their own programs to deal directly with the flow data files.

To summarise: NeTraMet is used for production traffic measurements by ISPs, Enterprises and University/Research networks. Some of the users have large numbers of meters, on links at speeds from 10/100 Mbps to OC3 and above. The large-scale users have developed their own flow data reduction programs, and often store their reduced flow data in a database.

### 3 Measuring traffic flows with NeTraMet

Traffic flows can be measured at one or many points in a network. In principle, one places NeTraMet meters at the points where traffic is to be measured, and one or more NeMaC manager / meter readers at a convenient location. NeMaC configures the meters by downloading rulesets to them (the creation of rulesets is discussed in section 4), collects data from the meters at regular intervals and writes it to flow data files (data files and the processing of flow data is discussed in section 5). Approaches to storing and analysing flow data are discussed in section 6.

The following sections present three examples of how to place meters in various networks.

### 3.1 Single meter watching traffic to/from a site's intranet

Figure 1 (below) shows a single NeTraMet meter, running on a Unix or Linux host, observing packets passing across a 'gateway' network. Packets reach the gateway from the Internet (right of diagram) via a router. From the gateway router they go through a short network segment, through the firewall, and on into the site's intranet. The essential feature of this example is that all traffic into or out of the site passes through a single point, making that point the obvious place to meter the site's Internet traffic.

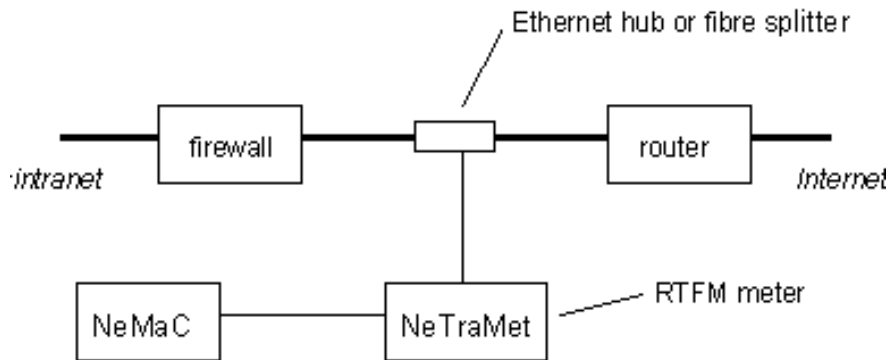


Figure 1: Single meter observing traffic at one point

There are several ways in which packets can be passed to the NeTraMet meter. If the gateway network uses an Ethernet segment, a hub can be used to copy all the packets (both inbound and outbound) to the meter. If the gateway router is a Unix or Linux host running a routing daemon, a NeTraMet meter can be run on the same host, with kernel packet header capture being used to copy packet headers into it. Lastly, if the Internet connection is on fibre, a pair of optical splitters (one for each direction) can be used to tap the fibre, copying the packets to a pair of fibre-optic interface cards on the NeTraMet host.

To the left of the meter is a Unix or Linux host running NeMaC, a combined RTFM manager and meter reader. NeMaC downloads rulesets to NeTraMet (telling it which flows are to be metered, and how much address detail is required for each flow). It also reads, at specified intervals, flow data from the meter, and writes it to flow data files.

In the diagram NeMaC and NeTraMet are shown running on separate hosts. This is only one possibility - if there is only one meter and meter reader it may be more sensible to run both on the same host, so as to minimise the delay in passing SNMP PDUs between meter and meter reader.

### 3.2 Several meters at remote locations, single meter reader

In Figure 2 (below) there are many meters (NeTraMet or NetFlowMet) observing flows at many points throughout a network, perhaps at different Points of Presence (PoPs) within an ISP's network. There is a single manager / meter reader (NeMaC) at the lower left of the diagram; this downloads rulesets to all the meters, and collects flow data from all of them at regular intervals.

This configuration requires good network connectivity between NeMaC and the meters, so that all the flow data can be retrieved from them in a time much shorter than the meter reading interval.

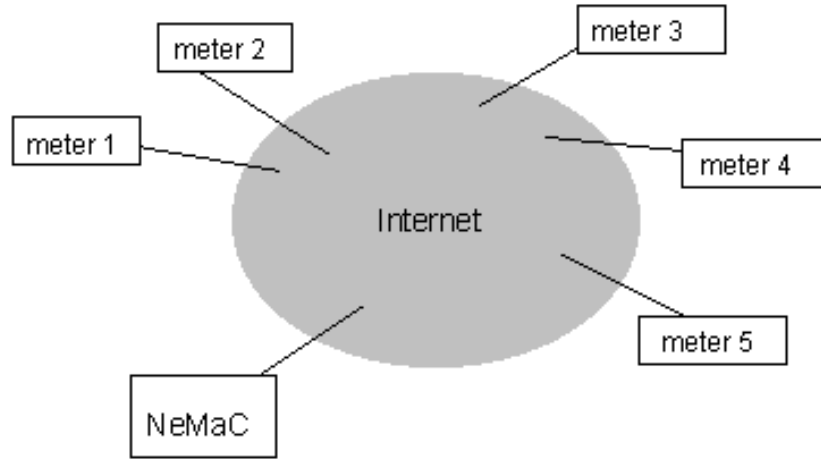


Figure 2: Multiple meters observing traffic at many points

More complicated versions of this configuration can also be useful. For example one could have two instances of NeMaC, both retrieving flow data from all the meters so as to provide redundant data collection. Similarly, one could have multiple meters observing traffic at some of the measurement points.

### 3.3 NetFlowMet meter collecting NetFlow data from several Cisco routers

This example is similar to the previous one, except that it uses NetFlowMet - the NetFlow version of the RTFM meter - to retrieve NetFlow data from four Cisco routers at various points in a wide-area network.

At the bottom left of Figure 3 (below) is a Unix or Linux host running NeMaC, which downloads rulesets to NetFlowMet and reads flow data from it. From the RTFM point of view, NetFlowMet is indistinguishable from NeTraMet. The only difference is that instead of observing packet headers directly, it reads NetFlow data.

Cisco's NetFlow data [4] summarises routing (unidirectional) flow data, which means that NetFlow data records contain total packet and byte counts for many pack-

ets. The time resolution of NetFlow records can be set when one configures a router to produce (and forward) NetFlow data.

If one is using a Cisco router, NetFlow provides a simple way to gather flow data from high-speed router interfaces. Care should be exercised, however, when turning on NetFlow, since it does have a performance impact on the router. Again, routers with many high-speed interfaces can generate high volumes of NetFlow data packets - this suggests that the best approach is to confine a NetFlowMet meter within the same PoP as the routers it serves.

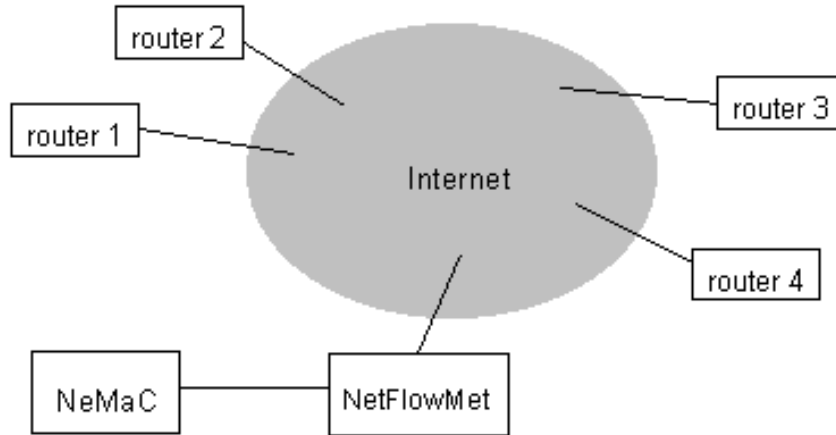


Figure 3: Multiple NetFlow meters observing traffic at many points

When used in this way, NetFlowMet provides a good way to aggregate NetFlow data from many routers. It can be an effective alternative to other tools such as cflowd [6].

### 3.4 NeTraMet hardware requirements

As indicated above, NeTraMet is most commonly used on 10/100 Mbps Ethernets. Two years ago, the reference manual stated 'on a 60 MHz Pentium with a PCI-bus Ethernet card NeTraMet can easily handle a steady load of 6,000 packets per second.'

Since then, network speeds have increased rapidly, but so has processor speed. Now, I routinely use a 500 MHz Pentium with 256 MB of memory and two ATM measurement cards [13], to meter traffic on an OC3 network carrying around 18,000 packets per second in each direction. In this configuration NeTraMet takes about 15% of the available processor time. Furthermore, NeTraMet can run several rulesets simultaneously; each additional running ruleset takes about 3% more of the available processor time.

## 4 Configuring Meters: attributes, flows and rulesets

Within the RTFM architecture [14] flows are identified by their end-point attribute values. The most commonly used attributes specify addresses at various levels of the protocol stack, e.g. SourcePeerAddress, DestPeerAddress (IP addresses) and SourceTransAddress, DestTransAddress (IP port numbers), SourceTransType (IP protocol, e.g. TCP, UDP, ICMP). These five attributes are sufficient to specify a 5-tuple, enough to identify individual TCP streams. If less detail is needed, fewer attributes should be specified, e.g. specifying only SourcePeerAddress and DestPeerAddress produces a smaller number of larger flows.

An RTFM ruleset describes the flows which a meter is required to measure. Rulesets are most easily written in SRL, the Simple Ruleset Language [12]. The next few sections give simple examples of rulesets.

### 4.1 Count all peer-to-peer (routing-style) flows

```
save SourcePeerAddress;
save DestPeerAddress;
count;
```

An RTFM meter takes each packet header and attempts to match it using the rules in its ruleset. The rules are executed in sequence, and save statements specify attribute values (from the packet) to be saved. The count statement causes the meter to stop processing, construct a 'flow key' from the saved attribute values, locate the appropriate flow in the meter's flow table using the flow key as an index, and increment the flow's packet and byte counters.

### 4.2 Count (stream-style) flows from a specified network

Within RTFM, flows are bi-directional, and a flow's data in the meter's flow table includes two sets of counters, one for each direction. If the packet matches the ruleset, and already has an entry in the meter's flow table, its forward counters are incremented. Otherwise the meter interchanges the packet's source and destination attributes and tries again; if this second match succeeds, the flow's reverse counters are incremented. From the user's point of view, one specifies the required direction for flows by testing values of source and/or destination values - this gives the meter enough information to maintain the counters for each direction.

```
if SourcePeerAddress == (130.216.7/24, 130.216.34/24) {
    save SourcePeerAddress;  save DestPeerAddress;
    save SourceTransAddress; save DestTransAddress;
    count;
}
```

In this example we test for the flow's source peer (IP) network number being one of those specified in the list. Note that addresses are specified as CIDR prefixes, and that any mixture of addresses may appear in such a list. If the packet is from one of the

listed networks we save its peer (IP) and transport (port number) addresses and count it.

The above examples give a very minimal introduction to RTFM rulesets. There are many more RTFM attributes which can be used when specifying flows, and SRL has many more features, including subroutines and simple textual defines. Furthermore, the NeTraMet distribution includes a small collection of example rulesets written in SRL, which can serve as a starting point when creating new rulesets. Overall, it is not particularly difficult to write new rulesets with which to make one's own (site-specific) flow measurements.

## 5 Gathering the data: NeMaC, flow data files

Once traffic meters (NeTraMet or NetFlowMet) and manager/meter readers (NeMaC) have been installed, and rulesets written and tested, one can begin collecting flow data. NeMaC reads a configuration file, which specifies the rulesets to be run, the meters on which to run them, and the intervals at which the resulting flow data should be read.

As well as specifying the collection interval one must also specify which attribute values are to be read for each flow. This is done using a format statement in the SRL program which creates one's rulesets. A format statement describes the record layout for flow records in the flow data files written by NeMaC. It is simply a list of RTFM attribute names, but it may also include literal strings like those in C's printf calls. Such literal strings can make it easier to discern particular attributes when examining flow data files by hand.

NeMaC's flow data files are simple text files, in which control records begin with a # character. The most important of the control records is the #Format one, which appears near the beginning of the file, and contains the format statement used when reading flow data from the meter.

A sample ruleset like the one described in section 4.2 above was run, collecting data at 60-second intervals. The following is an extract from the resulting flow data file:

```
#Format: flowruleset flowindex firsttime \  
        sourcepeeraddress destpeeraddress tooctets fromoctets  
#Time: 15:16:00 Fri 6 Oct 2000 k-meter.itss \  
        Flows from 336375550 to 336379060  
20 12890 336374321 130.216.7.14 195.29.214.2 6249 5898  
20 12891 336374340 130.216.34.238 213.0.2.198 110632 7680  
20 12892 336374370 130.216.7.14 205.219.255.33 7242 27243  
20 12895 336374613 130.216.34.10 140.200.128.13 769 3452  
  
...  
  
20 12914 336377526 130.216.34.10 202.96.128.68 87 183  
20 12915 336377566 130.216.34.10 202.96.134.133 85 146  
#EndData: k-meter.itss
```

The #Format record begins with three attributes which uniquely identify the flow, making it possible to compute differences (e.g. in packet counts) between successive meter readings. These attributes are FlowRuleset (an integer identifying this flow's ruleset), FlowIndex (this flow's index in the meter's flow table) and StartTime (time in SNMP Timeticks when this flow's first packet was observed). They are followed by SourcePeerAddress and DestPeerAddress, which will have the values set when the meter executed save statements in the ruleset. The last two attributes are ToOctets and From Octets, which are the total number of bytes seen in each direction for this flow.

Actual flow data records are contained between a #Time record, which gives the time of day (local time) when a meter reading commenced, and a #EndData record, which is written when this meter reading finished. Note that the SourcePeerAddresses in the flow data records are all within the networks specified in the ruleset (two 24-bit subnets of 130.216.0.0)

The packet and byte counts appear within the meter as 64-bit SNMP counters. They are incremented for each packet of the flow as it passes the meter, and they are never reset. Each time the meter is read, their values increase. To demonstrate this, here are the flow data records for a single flow on successive meter readings, taken at 60-second intervals:

```

20 12895 336374613 130.216.34.10 140.200.128.13 769 3452
20 12895 336374613 130.216.34.10 140.200.128.13 1956 9474
20 12895 336374613 130.216.34.10 140.200.128.13 2264 10469
20 12895 336374613 130.216.34.10 140.200.128.13 2666 11471
20 12895 336374613 130.216.34.10 140.200.128.13 2982 12461
20 12895 336374613 130.216.34.10 140.200.128.13 3690 15633

```

Observe that the last flow data record contains the total byte counts for all the packets the meter has observed for the flow. For long-running flows, one would like to know the number of bytes seen during the interval between meter readings. This can be obtained by computing differences between successive readings - but care is required in doing this, since the flow must be uniquely identified, and the difference should be computed as an unsigned 64-bit precision subtraction. The NeTraMet distribution includes fd\_filter, a utility program which computes attribute-value differences between readings in a flow data file.

## 6 Building a usage data collection system

Meters (NeTraMet, NetFlowMet) and Manager/Meter Readers (NeMaC) provide the basic tools needed to collect traffic flow data from one's network. Using them as part of a larger system is not difficult, but it demands thorough analysis (what problem is one trying to solve, what reports are required?) and careful implementation.

As an example, consider a simple cost recovery system, which will measure the total traffic volume for each IP address, and produce daily and monthly usage summaries. One could begin by setting up traffic meters, and developing a suitable ruleset, specifying the amount of detail to be collected for each flow. The next step would be to start NeMaC, which would begin writing flow data files to disk. This, however, would

produce a single ever-growing file, which would quickly become unworkable. Clearly it would be better to produce a series of shorter files, say one for each calendar day. This could easily be done by running a daily cron job which renames the flow data file - the next time NeMaC reads the meter it sees that the flow data file has disappeared, and automatically starts a new one with the original name. Now that we have daily flow data files, a script can be written to produce the required daily reports. Similarly, at the end of the month, another script can produce the monthly summaries.

Daily scripts such as the one discussed above can also run `fd_filter` so as to produce daily 'differences' files. Summary files can use the differences to show usage by time of day, with the total usage for the day being the sum of the day's differences.

Another approach is to compute differences, then write them directly into a relational database. This makes the flow data easily accessible to managers, and can simplify the programming task of generating 'special' reports when the need arises.

## 7 Real-time flow analysis: nifty

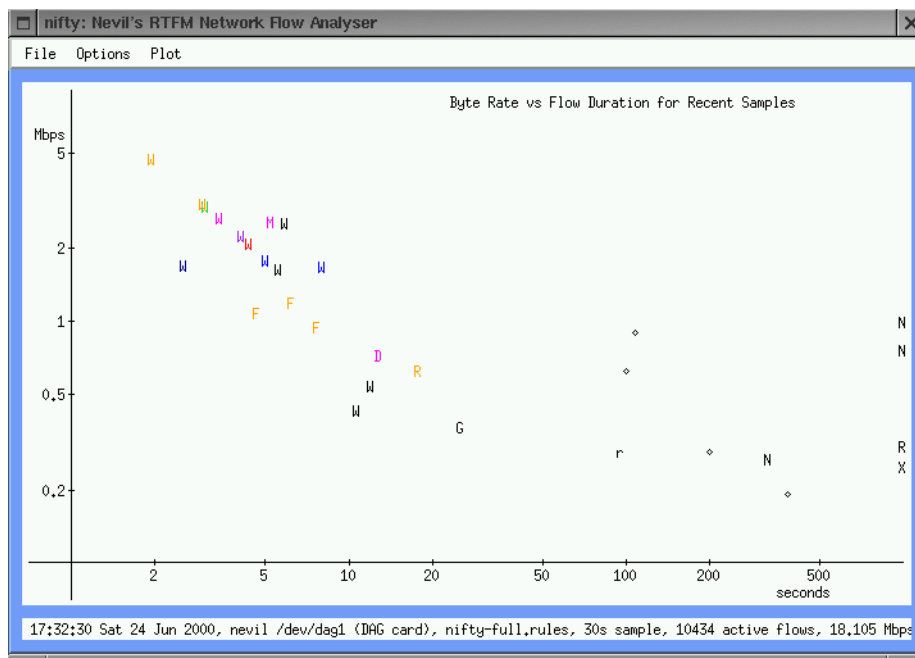


Figure 4: Example plot using nifty's ruleset from the NeTraMet distribution

Earlier sections have treated flow data collection as a long-term background activity. Of course it is also possible to analyse the flow data in near-real-time. The NeTraMet distribution includes an X Windows program called nifty, which is a real-time traffic flow analyser. That is, nifty is an RTFM Manager / Meter Reader, which

downloads a ruleset to a meter, then reads flow data and uses it to produce a regularly-updated plot showing the busiest traffic flows.

The NeTraMet distribution provides a ruleset which nifty can use to determine applications types (using IP port numbers) to specify the plot symbol, e.g. W for http flows, D for DNS flows, F for FTP, and so on.

On Figure 4 (above) the horizontal axis shows the flow's duration, i.e. the time in seconds since its first packet was observed. Long-running flows appear on the left of the picture, then drift to the right. When they terminate (i.e. the meter sees no further packets), they remain stationary of the plot for a few meter readings, changing colour from blue through to red, before finally disappearing. The vertical axis shows the flow data rates, i.e. the average rate over a flow's lifetime.

In this plot, a web page has been downloaded over a fairly fast link, producing a 'diagonal' pattern of flows (plotted as W). At the right of the plot there are two very long-running network news flows (plotted as N), one X Window flow (X) and a RealAudio flow (R). Usually the most interesting flows are the long-term ones which move very large amounts of data, and very short-duration ones which can flood router queues.

## 8 Experience with NeTraMet

Early implementation experiences with NeTraMet are described in [15]. Bear in mind that [15] was written before SRL [12] had been developed; using SRL greatly simplifies the task of writing rulesets. Once one understands how the meter matches packets so as to determine a packet's direction within a flow, SRL is not a difficult language to use. Simple rulesets, e.g. one which measures flows between two lists of IP networks, are straightforward to write and not difficult to maintain. Although maintaining rulesets is still a non-trivial task, rulesets are no more difficult to work with than the configuration of a large multiport router.

Very little has been published about NeTraMet's use for production network management, probably because it is commonly used to gather usage information which is often used for billing. However, [16] documents the Kawaihiko (New Zealand Universities') network, where NeTraMet was used routinely for many years.

The RTFM architecture, with its separation into meters, meter readers and managers, has proved very stable and reliable in use. The architecture provides considerable redundancy. Many sites use more than one meter observing traffic on the same segment to guard against meter failure. One meter can also be read by several meter readers, but this feature has not been widely used. Cases have been reported, however, where a meter reader failed, was restarted several weeks later, and was able to read all the flow data accumulated by a meter on a remote network segment.

Several sites have performed tests for accuracy of NeTraMet compared with other measurement tools, and have reported (by email) very good agreement. Unfortunately, as far as I am aware, no written reports on this have been published.

NeTraMet is sometimes used for security incident detection (also referred to as 'IP Auditing'), by collecting flow data with endpoint IP addresses and port numbers. Incidents such as port scans stand out clearly when one examines the flow data files. This

approach can be satisfactory if NeTraMet is being run for other management purposes, but if only IP Auditing is required one should use a more specific tool such as Argus [17].

## 9 What's special about RTFM and NeTraMet?

RTFM is unique in that it provides a generalised, asynchronous, distributed system for measuring network traffic flows. Although the examples presented here have only used IP Version 4, RTFM works well with other protocols such as IP Version 6, Novell IPX, and AppleTalk. RTFM's definition of flows as bi-directional logical entities defined only by their end-point attributes is very powerful, and this power is readily available to users via the SRL language for creating rulesets. In short, RTFM is an effective, purpose-designed system for measuring traffic flows. More information on RTFM can be found on [10].

NeTraMet is available from [7] - the web page lists several distributions sites around the world. As mentioned above, NeTraMet is an open-source implementation of RTFM, providing a good toolkit for measuring traffic flows. It can be daunting in its complexity - after all, most new users would really like something which 'works right out of the box' - but every network has different flow measurement needs, so one can't avoid having to put some effort into understanding the RTFM system, and configuring the NeTraMet components to produce the required measurements. On the other hand, there are only a few components to understand - meters, meter readers, managers, rule-sets and SRL - and these can readily be used to make the measurements and produce whatever reports are needed.

The RTFM Meter MIB [11] is a Proposed Internet Standard. If it were implemented in switches and routers, traffic flow measurements would become much easier, since there would be little need to deploy software-based meters such as NeTraMet. At this stage, alas, no hardware vendor has implemented the Meter MIB. If this is to change, RTFM and NeTraMet users need to keep asking their hardware vendors "have you implemented the RFC 2720 Meter MIB?" It's over to you ..

## 10 Acknowledgement

I wish to acknowledge The University of Auckland for their continuing support of both RTFM and NeTraMet. NeTraMet was written in Auckland in 1992, and has been in continuous use ever since. Like all open-source software, NeTraMet is very much shaped by feedback from its users. Special thanks to all those who have contributed to the NeTraMet mailing list, especially those who have sent me bug reports, or who have implemented new features. Thanks are also due to CAIDA, where I developed the CoralReef version of the NeTraMet meter while I was on sabbatical leave last year.

## References

- [1] V. Jacobson, C. Leres, and S. McCanne, *tcpdump*, available at <ftp://ftp.ee.lbl.gov>, June 1989
- [2] A. Feldmann, *BLT: Bi-Layer Tracing of HTTP and TCP/IP*, WWW9, Amsterdam, May 2000, <http://www9.org/w9cdrom/367/367.html>
- [3] *NLANR network traffic packet header traces*, <http://moat.nlanr.net/Traces/>
- [4] *NetFlow Services and Applications (an introduction and overview)*, [http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps\\_wp.htm](http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm)
- [5] P. Amsden, J. Amweg, P. Calato, S. Bensley and G. Lyons, *Cabletron's Light-weight Flow Admission Protocol, Version 1.0*, RFC 2124, March 1997
- [6] *cflowd*, <http://www.caida.org/tools/measurement/cflowd/>
- [7] Netramet website, <http://www.auckland.ac.nz/net/NeTraMet/>
- [8] Waldbusser, *Remote Network Monitoring Management Information Base*, RFC 2819, May 2000
- [9] IETF home page, <http://www.ietf.org>
- [10] RTFM website, <http://www.auckland.ac.nz/net/Internet/rtfm/>
- [11] Nevil Brownlee, *Traffic Flow Measurement: Meter MIB*, RFC 2720, Oct 99
- [12] Nevil Brownlee, *SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups*, RFC 2723, Oct 99
- [13] Dag home page, <http://dag.cs.waikato.ac.nz/>
- [14] Nevil Brownlee, Cyndi Mills and Greg Ruth, *Traffic Flow Measurement: Architecture*, RFC 2722, Oct 1999
- [15] N. Brownlee, *Traffic Flow Measurement: Experiences with NeTraMet*, RFC 2123, March 1997
- [16] N. Brownlee and R. Fulton, *Kawaihiko and the 3rd Quartile Day*, IEEE Communications, July 2000
- [17] Argus home page, <ftp://www.qosient.com/>