Network Working Group
Request for Comments: 2722
Obsoletes: 2063
Category: Informational

N. Brownlee
The University of Auckland
C. Mills
GTE Laboratories, Inc
G. Ruth
GTE Internetworking

October 1999

Traffic Flow Measurement: Architecture

Status of this Memo

Copyright Notice

Abstract

This document provides a general framework for describing network traffic flows, presents an architecture for traffic flow measurement and reporting, discusses how this relates to an overall network traffic flow architecture and indicates how it can be used within the Internet.

Table of Contents

1 Statement of Purpose and Scope

1.1 Introduction

This document describes an architecture for traffic flow measurement and reporting for data networks which has the following characteristics:

- The traffic flow model can be consistently applied to any protocol, using address attributes in any combination at the 'adjacent' (see below), network and transport layers of the networking stack.

- Traffic flow attributes are defined in such a way that they are valid for multiple networking protocol stacks, and that traffic flow measurement implementations are useful in multi-protocol environments.

- Users may specify their traffic flow measurement requirements by writing 'rule sets', allowing them to collect the flow data they need while ignoring other traffic.

- The data reduction effort to produce requested traffic flow information is placed as near as possible to the network measurement point. This minimises the volume of data to be obtained (and transmitted across the network for storage), and reduces the amount of processing required in traffic flow analysis applications.

'Adjacent' (as used above) is a layer-neutral term for the next layer down in a particular instantiation of protocol layering. Although 'adjacent' will usually imply the link layer (MAC addresses), it does not implicitly advocate or dismiss any particular form of tunnelling or layering.

The architecture specifies common metrics for measuring traffic flows. By using the same metrics, traffic flow data can be exchanged and compared across multiple platforms. Such data is useful for:

- Understanding the behaviour of existing networks,

- Planning for network development and expansion,

- Quantification of network performance,

- Verifying the quality of network service, and

- Attribution of network usage to users.

The traffic flow measurement architecture is deliberately structured using address attributes which are defined in a consistent way at the Adjacent, Network and Transport layers of the networking stack, allowing specific implementations of the architecture to be used effectively in multi-protocol environments. Within this document the term 'usage data' is used as a generic term for the data obtained using the traffic flow measurement architecture.

In principle one might define address attributes for higher layers, but it would be very difficult to do this in a general way. However, if an RTFM traffic meter were implemented within an application server (where it had direct access to application-specific usage information), it would be possible to use the rest of the RTFM architecture to collect application-specific information. Use of the same model for both network- and application-level measurement in this way could simplify the development of generic analysis applications which process and/or correlate both traffic and usage information. Experimental work in this area is described in the RTFM 'New Attributes' document [RTFM-NEW].

This document is not a protocol specification. It specifies and structures the information that a traffic flow measurement system needs to collect, describes requirements that such a system must meet, and outlines tradeoffs which may be made by an implementor.

For performance reasons, it may be desirable to use traffic information gathered through traffic flow measurement in lieu of network statistics obtained in other ways. Although the quantification of network performance is not the primary purpose of this architecture, the measured traffic flow data may be used as an indication of network performance.

A cost recovery structure decides "who pays for what." The major issue here is how to construct a tariff (who gets billed, how much, for which things, based on what information, etc). Tariff issues include fairness, predictability (how well can subscribers forecast their network charges), practicality (of gathering the data and administering the tariff), incentives (e.g. encouraging off-peak use), and cost recovery goals (100% recovery, subsidisation, profit making). Issues such as these are not covered here.

Background information explaining why this approach was selected is provided by the 'Internet Accounting Background' RFC [ACT-BKG].

2 Traffic Flow Measurement Architecture

A traffic flow measurement system is used by Network Operations personnel to aid in managing and developing a network. It provides a tool for measuring and understanding the network's traffic flows. This information is useful for many purposes, as mentioned in section 1 (above).

The following sections outline a model for traffic flow measurement, which draws from working drafts of the OSI accounting model [OSI-ACT].

2.1 Meters and Traffic Flows

At the heart of the traffic measurement model are network entities called traffic METERS. Meters observe packets as they pass by a single point on their way through the network and classify them into certain groups. For each such group a meter will accumulate certain attributes, for example the numbers of packets and bytes observed for the group. These METERED TRAFFIC GROUPS may correspond to a user, a host system, a network, a group of networks, a particular transport address (e.g. an IP port number), any combination of the above, etc, depending on the meter's configuration.

We assume that routers or traffic monitors throughout a network are instrumented with meters to measure traffic. Issues surrounding the choice of meter placement are discussed in the 'Internet Accounting Background' RFC [ACT-BKG]. An important aspect of meters is that they provide a way of succinctly aggregating traffic information.

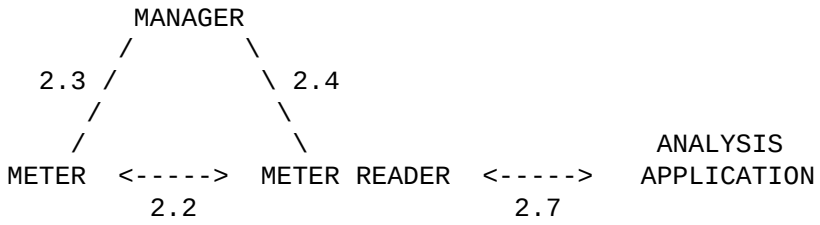
For the purpose of traffic flow measurement we define the concept of a TRAFFIC FLOW, which is like an artificial logical equivalent to a call or connection. A flow is a portion of traffic, delimited by a start and stop time, that belongs to one of the metered traffic groups mentioned above. Attribute values (source/destination addresses, packet counts, byte counts, etc.) associated with a flow are aggregate quantities reflecting events which take place in the DURATION between the start and stop times. The start time of a flow is fixed for a given flow; the stop time may increase with the age of the flow.

For connectionless network protocols such as IP there is by definition no way to tell whether a packet with a particular source/destination combination is part of a stream of packets or not - each packet is completely independent. A traffic meter has, as part of its configuration, a set of 'rules' which specify the flows of interest, in terms of the values of their attributes. It derives attribute values from each observed packet, and uses these to decide

which flow they belong to. Classifying packets into 'flows' in this way provides an economical and practical way to measure network traffic and subdivide it into well-defined groups.

Usage information which is not derivable from traffic flows may also be of interest. For example, an application may wish to record accesses to various different information resources or a host may wish to record the username (subscriber id) for a particular network session. Provision is made in the traffic flow architecture to do this. In the future the measurement model may be extended to gather such information from applications and hosts so as to provide values for higher-layer flow attributes.

As well as FLOWS and METERS, the traffic flow measurement model includes MANAGERS, METER READERS and ANALYSIS APPLICATIONS, which are explained in following sections. The relationships between them are shown by the diagram below. Numbers on the diagram refer to sections in this document.



- MANAGER: A traffic measurement manager is an application which configures 'meter' entities and controls 'meter reader' entities. It sends configuration commands to the meters, and supervises the proper operation of each meter and meter reader. It may well be convenient to combine the functions of meter reader and manager within a single network entity.

- METER: Meters are placed at measurement points determined by Network Operations personnel. Each meter selectively records network activity as directed by its configuration settings. It can also aggregate, transform and further process the recorded activity before the data is stored. The processed and stored results are called the 'usage data'.

- METER READER: A meter reader transports usage data from meters so that it is available to analysis applications.

- ANALYSIS APPLICATION: An analysis application processes the usage data so as to provide information and reports which are useful for network engineering and management purposes. Examples include:

  - TRAFFIC FLOW MATRICES, showing the total flow rates for many of the possible paths within an internet.

  - FLOW RATE FREQUENCY DISTRIBUTIONS, summarizing flow rates over a period of time.

  - USAGE DATA showing the total traffic volumes sent and received by particular hosts.

The operation of the traffic measurement system as a whole is best understood by considering the interactions between its components. These are described in the following sections.

2.2 Interaction Between METER and METER READER

The information which travels along this path is the usage data itself. A meter holds usage data in an array of flow data records known as the FLOW TABLE. A meter reader may collect the data in any suitable manner. For example it might upload a copy of the whole flow table using a file transfer protocol, or read the records in the current flow set one at a time using a suitable data transfer protocol. Note that the meter reader need not read complete flow data records, a subset of their attribute values may well be sufficient.

A meter reader may collect usage data from one or more meters. Data may be collected from the meters at any time. There is no requirement for collections to be synchronized in any way.

2.3 Interaction Between MANAGER and METER

A manager is responsible for configuring and controlling one or more meters. Each meter's configuration includes information such as:

- Flow specifications, e.g. which traffic flows are to be measured, how they are to be aggregated, and any data the meter is required to compute for each flow being measured.

- Meter control parameters, e.g. the 'inactivity' time for flows (if no packets belonging to a flow are seen for this time the flow is considered to have ended, i.e. to have become idle).

- Sampling behaviour. Normally every packet will be observed. It may sometimes be necessary to use sampling techniques so as to observe only some of the packets (see following note).

A note about sampling: Current experience with the measurement architecture shows that a carefully-designed and implemented meter compresses the data sufficiently well that in normal LANs and WANs of today sampling is seldom, if ever, needed. For this reason sampling algorithms are not prescribed by the architecture. If sampling is needed, e.g. for metering a very-high-speed network with fine-grained flows, the sampling technique should be carefully chosen so as not to bias the results. For a good introduction to this topic see the IPPM Working Group's RFC "Framework for IP Performance Metrics" [IPPM-FRM].

A meter may run several rule sets concurrently on behalf of one or more managers, and any manager may download a set of flow specifications (i.e. a 'rule set') to a meter. Control parameters which apply to an individual rule set should be set by the manager after it downloads that rule set.

One manager should be designated as the 'master' for a meter. Parameters such as sampling behaviour, which affect the overall operation of the meter, should only be set by the master manager.

2.4 Interaction Between MANAGER and METER READER

A manager is responsible for configuring and controlling one or more meter readers. A meter reader may only be controlled by a single manager. A meter reader needs to know at least the following for every meter it is collecting usage data from:

- The meter's unique identity, i.e. its network name or address.

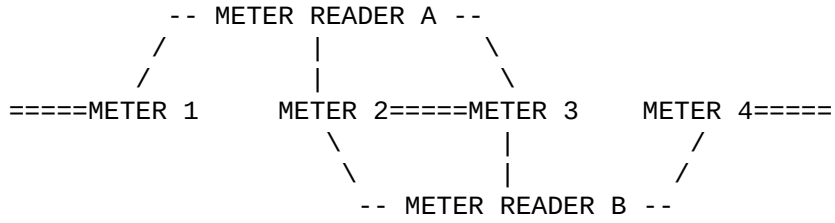
- How often usage data is to be collected from the meter.

- Which flow records are to be collected (e.g. all flows, flows for a particular rule set, flows which have been active since a given time, etc.).

- Which attribute values are to be collected for the required flow records (e.g. all attributes, or a small subset of them)

Since redundant reporting may be used in order to increase the reliability of usage data, exchanges among multiple entities must be considered as well. These are discussed below.

2.5 Multiple METERs or METER READERs



Several uniquely identified meters may report to one or more meter readers. The diagram above gives an example of how multiple meters and meter readers could be used.

In the diagram above meter 1 is read by meter reader A, and meter 4 is read by meter reader B. Meters 1 and 4 have no redundancy; if either meter fails, usage data for their network segments will be lost.

Meters 2 and 3, however, measure traffic on the same network segment. One of them may fail leaving the other collecting the segment's usage data. Meters 2 and 3 are read by meter reader A and by meter reader B. If one meter reader fails, the other will continue collecting usage data from both meters.

The architecture does not require multiple meter readers to be synchronized. In the situation above meter readers A and B could both collect usage data at the same intervals, but not necesarily at the same times. Note that because collections are asynchronous it is unlikely that usage records from two different meter readers will agree exactly.

If identical usage records were required from a single meter, a manager could achieve this using two identical copies of a ruleset in that meter. Let's call them RS1 and RS2, and assume that RS1 is running. When a collection is to be made the manager switches the meter from RS1 to RS2, and directs the meter reader(s) to read flow data for RS1 from the meter. For the next collection the manager switches back to RS1, and so on. Note, however, that it is not possible to get identical usage records from more than one meter, since there is no way for a manager to switch rulesets in more than one meter at the same time.

If there is only one meter reader and it fails, the meters continue to run. When the meter reader is restarted it can collect all of the accumulated flow data. Should this happen, time resolution will be lost (because of the missed collections) but overall traffic flow information will not. The only exception to this would occur if the

traffic volume was sufficient to 'roll over' counters for some flows during the failure; this is addressed in the section on 'Rolling Counters'.

2.6 Interaction Between MANAGERs (MANAGER - MANAGER)

Synchronization between multiple management systems is the province of network management protocols. This traffic flow measurement architecture specifies only the network management controls necessary to perform the traffic flow measurement function and does not address the more global issues of simultaneous or interleaved (possibly conflicting) commands from multiple network management stations or the process of transferring control from one network management station to another.

2.7 METER READERs and APPLICATIONs

Once a collection of usage data has been assembled by a meter reader it can be processed by an analysis application. Details of analysis applications - such as the reports they produce and the data they require - are outside the scope of this architecture.

It should be noted, however, that analysis applications will often require considerable amounts of input data. An important part of running a traffic flow measurement system is the storage and regular reduction of flow data so as to produce daily, weekly or monthly summary files for further analysis. Again, details of such data handling are outside the scope of this architecture.

3 Traffic Flows and Reporting Granularity

A flow was defined in section 2.1 above in abstract terms as follows:

> "A TRAFFIC FLOW is an artifical logical equivalent to a call or connection, belonging to a (user-specieied) METERED TRAFFIC GROUP."

In practical terms, a flow is a stream of packets observed by the meter as they pass across a network between two end points (or from a single end point), which have been summarized by a traffic meter for analysis purposes.

3.1 Flows and their Attributes

Every traffic meter maintains a table of 'flow records' for flows seen by the meter. A flow record holds the values of the ATTRIBUTES of interest for its flow. These attributes might include:

- ADDRESSES for the flow's source and destination. These comprise the protocol type, the source and destination addresses at various network layers (extracted from the packet header), and the number of the interface on which the packet was observed.

- First and last TIMES when packets were seen for this flow, i.e. the 'creation' and 'last activity' times for the flow.

- COUNTS for 'forward' (source to destination) and 'backward' (destination to source) components (e.g. packets and bytes) of the flow's traffic. The specifying of 'source' and 'destination' for flows is discussed in the section on packet matching below.

- OTHER attributes, e.g. the index of the flow's record in the flow table and the rule set number for the rules which the meter was running while the flow was observed. The values of these attributes provide a way of distinguishing flows observed by a meter at different times.

The attributes listed in this document (Appendix C) provide a basic (i.e. useful minimum) set; IANA considerations for allocating new attributes are set out in section 8 below.

A flow's METERED TRAFFIC GROUP is specified by the values of its ADDRESS attributes. For example, if a flow's address attributes were specified as "source address = IP address 10.1.0.1, destination address = IP address 26.1.0.1" then only IP packets from 10.1.0.1 to 26.1.0.1 and back would be counted in that flow. If a flow's address attributes specified only that "source address = IP address 10.1.0.1," then all IP packets from and to 10.1.0.1 would be counted in that flow.

The addresses specifying a flow's address attributes may include one or more of the following types:

- The INTERFACE NUMBER for the flow, i.e. the interface on which the meter measured the traffic. Together with a unique address for the meter this uniquely identifies a particular physical-level port.

- The ADJACENT ADDRESS, i.e. the address in the the next layer down from the peer address in a particular instantiation of protocol layering. Although 'adjacent' will usually imply the link layer, it does not implicitly advocate or dismiss any particular form of tunnelling or layering.

For example, if flow measurement is being performed using IP as the network layer on an Ethernet LAN [802-3], an adjacent address will normally be a six-octet Media Access Control (MAC) address. For a host connected to the same LAN segment as the meter the adjacent address will be the MAC address of that host. For hosts on other LAN segments it will be the MAC address of the adjacent (upstream or downstream) router carrying the traffic flow.

- The PEER ADDRESS, which identifies the source or destination of the packet for the network layer (n) at which traffic measurement is being performed. The form of a peer address will depend on the network-layer protocol in use, and the measurement network layer (n).

- The TRANSPORT ADDRESS, which identifies the source or destination port for the packet, i.e. its (n+1) layer address. For example, if flow measurement is being performed at the IP layer a transport address is a two-octet UDP or TCP port number.

The four definitions above specify addresses for each of the four lowest layers of the OSI reference model, i.e. Physical layer, Link layer, Network layer and Transport layer. A FLOW RECORD stores both the VALUE for each of its addresses (as described above) and a MASK specifying which bits of the address value are being used and which are ignored. Note that if address bits are being ignored the meter will set them to zero, however their actual values are undefined.

One of the key features of the traffic measurement architecture is that attributes have essentially the same meaning for different protocols, so that analysis applications can use the same reporting formats for all protocols. This is straightforward for peer addresses; although the form of addresses differs for the various protocols, the meaning of a 'peer address' remains the same. It becomes harder to maintain this correspondence at higher layers - for example, at the Network layer IP, Novell IPX and AppleTalk all use port numbers as a 'transport address', but CLNP and DECnet have no notion of ports.

Reporting by adjacent intermediate sources and destinations or simply by meter interface (most useful when the meter is embedded in a router) supports hierarchical Internet reporting schemes as described in the 'Internet Accounting Background' RFC [ACT-BKG]. That is, it allows backbone and regional networks to measure usage to just the next lower level of granularity (i.e. to the regional and stub/enterprise levels, respectively), with the final breakdown according to end user (e.g. to source IP address) performed by the stub/enterprise networks.

In cases where network addresses are dynamically allocated (e.g. dial-in subscribers), further subscriber identification will be necessary if flows are to ascribed to individual users. Provision is made to further specify the metered traffic group through the use of an optional SUBSCRIBER ID as part of the flow id. A subscriber ID may be associated with a particular flow either through the current rule set or by unspecified means within a meter. At this time a subscriber ID is an arbitrary text string; later versions of the architecture may specify details of its contents.

3.2 Granularity of Flow Measurements

GRANULARITY is the 'control knob' by which an application and/or the meter can trade off the overhead associated with performing usage reporting against its level of detail. A coarser granularity means a greater level of aggregation; finer granularity means a greater level of detail. Thus, the number of flows measured (and stored) at a meter can be regulated by changing the granularity of their attributes. Flows are like an adjustable pipe - many fine-granularity streams can carry the data with each stream measured individually, or data can be bundled in one coarse-granularity pipe. Time granularity may be controlled by varying the reporting interval, i.e. the time between meter readings.

Flow granularity is controlled by adjusting the level of detail for the following:

- The metered traffic group (address attributes, discussed above).
- The categorisation of packets (other attributes, discussed below).
- The lifetime/duration of flows (the reporting interval needs to be short enough to measure them with sufficient precision).

The set of rules controlling the determination of each packet's metered traffic group is known as the meter's CURRENT RULE SET. As will be shown, the meter's current rule set forms an integral part of the reported information, i.e. the recorded usage information cannot be properly interpreted without a definition of the rules used to collect that information.

Settings for these granularity factors may vary from meter to meter. They are determined by the meter's current rule set, so they will change if network Operations personnel reconfigure the meter to use a new rule set. It is expected that the collection rules will change rather infrequently; nonetheless, the rule set in effect at any time

must be identifiable via a RULE SET NUMBER. Granularity of metered traffic groups is further specified by additional ATTRIBUTES. These attributes include:

- Attributes which record information derived from other attribute values. Six of these are defined (SourceClass, DestClass, FlowClass, SourceKind, DestKind, FlowKind), and their meaning is determined by the meter's rule set. For example, one could have a subroutine in the rule set which determined whether a source or destination peer address was a member of an arbitrary list of networks, and set SourceClass/DestClass to one if the source/dest peer address was in the list or to zero otherwise.

- Administratively specified attributes such as Quality of Service and Priority, etc. These are not defined at this time.

Settings for these granularity factors may vary from meter to meter. They are determined by the meter's current rule set, so they will change if Network Operations personnel reconfigure the meter to use a new rule set.

A rule set can aggregate groups of addresses in two ways. The simplest is to use a mask in a single rule to test for an address within a masked group. The other way is to use a sequence of rules to test for an arbitrary group of (masked) address values, then use a PushRuleTo rule to set a derived attribute (e.g. FlowKind) to indicate the flow's group.

The LIFETIME of a flow is the time interval which began when the meter observed the first packet belonging to the flow and ended when it saw the last packet. Flow lifetimes are very variable, but many - if not most - are rather short. A meter cannot measure lifetimes directly; instead a meter reader collects usage data for flows which have been active since the last collection, and an analysis application may compare the data from each collection so as to determine when each flow actually stopped.

The meter does, however, need to reclaim memory (i.e. records in the flow table) being held by idle flows. The meter configuration includes a variable called InactivityTimeout, which specifies the minimum time a meter must wait before recovering the flow's record. In addition, before recovering a flow record the meter should be sure that the flow's data has been collected by all meter readers which registered to collect it. These two wait conditions are desired goals for the meter; they are not difficult to achieve in normal usage, however the meter cannot guarantee to fulfil them absolutely.

These 'lifetime' issues are considered further in the section on meter readers (below). A complete list of the attributes currently defined is given in Appendix C later in this document.

3.3 Rolling Counters, Timestamps, Report-in-One-Bucket-Only

Once a usage record is sent, the decision needs to be made whether to clear any existing flow records or to maintain them and add to their counts when recording subsequent traffic on the same flow. The second method, called rolling counters, is recommended and has several advantages. Its primary advantage is that it provides greater reliability - the system can now often survive the loss of some usage records, such as might occur if a meter reader failed and later restarted. The next usage record will very often contain yet another reading of many of the same flow buckets which were in the lost usage record. The 'continuity' of data provided by rolling counters can also supply information used for "sanity" checks on the data itself, to guard against errors in calculations.

The use of rolling counters does introduce a new problem: how to distinguish a follow-on flow record from a new flow record. Consider the following example.

	CONTINUING FLOW	OLD FLOW, then NEW FLOW
Usage record N:	start time = 1 flow count = 2000	start time = 1 flow count = 2000 (done)
Usage record N+1:	start time = 1 flow count = 3000	start time = 5 new flow count = 1000
Total count:	3000	3000

In the continuing flow case, the same flow was reported when its count was 2000, and again at 3000: the total count to date is 3000. In the OLD/NEW case, the old flow had a count of 2000. Its record was then stopped (perhaps because of temporary idleness), but then more traffic with the same characteristics arrived so a new flow record was started and it quickly reached a count of 1000. The total flow count from both the old and new records is 3000.

The flow START TIMESTAMP attribute is sufficient to resolve this. In the example above, the CONTINUING FLOW flow record in the second usage record has an old FLOW START timestamp, while the NEW FLOW contains a recent FLOW START timestamp. A flow which has sporadic bursts of activity interspersed with long periods of inactivity will produce a sequence of flow activity records, each with the same set of address attributes, but with increasing FLOW START times.

Each packet is counted in at most one flow for each running ruleset, so as to avoid multiple counting of a single packet. The record of a single flow is informally called a "bucket." If multiple, sometimes overlapping, records of usage information are required (aggregate, individual, etc), the network manager should collect the counts in sufficiently detailed granularity so that aggregate and combination counts can be reconstructed in post-processing of the raw usage data. Alternatively, multiple rulesets could be used to collect data at different granularities.

For example, consider a meter from which it is required to record both 'total packets coming in interface #1' and 'total packets arriving from any interface sourced by IP address = a.b.c.d', using a single rule set. Although a bucket can be declared for each case, it is not clear how to handle a packet which satisfies both criteria. It must only be counted once. By default it will be counted in the first bucket for which it qualifies, and not in the other bucket. Further, it is not possible to reconstruct this information by post-processing. The solution in this case is to define not two, but THREE buckets, each one collecting a unique combination of the two criteria:

```
        Bucket 1:  Packets which came in interface 1,
                   AND were sourced by IP address a.b.c.d

        Bucket 2:  Packets which came in interface 1,
                   AND were NOT sourced by IP address a.b.c.d

        Bucket 3:  Packets which did NOT come in interface 1,
                   AND were sourced by IP address a.b.c.d

       (Bucket 4:  Packets which did NOT come in interface 1,
                   AND were NOT sourced by IP address a.b.c.d)
```

The desired information can now be reconstructed by post-processing. "Total packets coming in interface 1" can be found by adding buckets 1 & 2, and "Total packets sourced by IP address a.b.c.d" can be found by adding buckets 1 & 3. Note that in this case bucket 4 is not explicitly required since its information is not of interest, but it is supplied here in parentheses for completeness.

Alternatively, the above could be achieved by running two rule sets (A and B), as follows:

```
        Bucket 1:  Packets which came in interface 1;
                   counted by rule set A.
```

Bucket 2: Packets which were sourced by IP address a.b.c.d; counted by rule set B.

4 Meters

A traffic flow meter is a device for collecting data about traffic flows at a given point within a network; we will call this the METERING POINT. The header of every packet passing the network metering point is offered to the traffic meter program.

A meter could be implemented in various ways, including:

- A dedicated small host, connected to a broadcast LAN (so that it can see all packets as they pass by) and running a traffic meter program. The metering point is the LAN segment to which the meter is attached.

- A multiprocessing system with one or more network interfaces, with drivers enabling a traffic meter program to see packets. In this case the system provides multiple metering points - traffic flows on any subset of its network interfaces can be measured.

- A packet-forwarding device such as a router or switch. This is similar to (b) except that every received packet should also be forwarded, usually on a different interface.

4.1 Meter Structure

An outline of the meter's structure is given in the following diagram:

Briefly, the meter works as follows:

- Incoming packet headers arrive at the top left of the diagram and are passed to the PACKET PROCESSOR.

- The packet processor passes them to the Packet Matching Engine (PME) where they are classified.

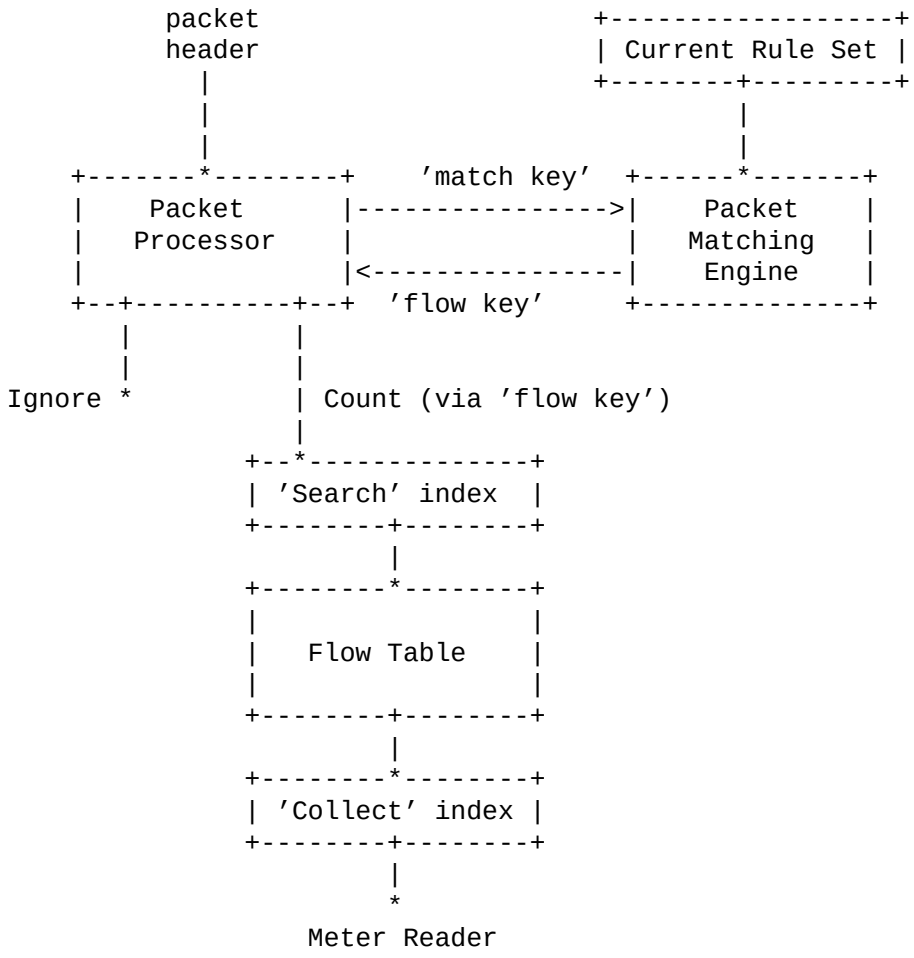
- The PME is a Virtual Machine running a pattern matching program contained in the CURRENT RULE SET. It is invoked by the Packet Processor, executes the rules in the current rule set as described in section 4.3 below, and returns instructions on what to do with the packet.

- Some packets are classified as 'to be ignored'. They are discarded by the Packet Processor.

- Other packets are matched by the PME, which returns a FLOW KEY describing the flow to which the packet belongs.

- The flow key is used to locate the flow's entry in the FLOW TABLE; a new entry is created when a flow is first seen. The entry's data fields (e.g. packet and byte counters) are updated.

- A meter reader may collect data from the flow table at any time. It may use the 'collect' index to locate the flows to be collected within the flow table.



The discussion above assumes that a meter will only be running a single rule set. A meter may, however, run several rule sets concurrently. To do this the meter maintains a table of current rulesets. The packet processor matches each packet against every

current ruleset, producing a single flow table containing flows from all the rule sets. One way to implement this is to use the Rule Set Number attribute in each flow as part of the flow key.

A packet may only be counted once in a rule set (as explained in section 3.3 above), but it may be counted in any of the current rulesets. The overall effect of doing this is somewhat similar to running several independent meters, one for each rule set.

4.2 Flow Table

Every traffic meter maintains 'flow table', i.e. a table of TRAFFIC FLOW RECORDS for flows seen by the meter. Details of how the flow table is maintained are given in section 4.5 below. A flow record contains attribute values for its flow, including:

- Addresses for the flow's source and destination. These include addresses and masks for various network layers (extracted from the packet header), and the identity of the interface on which the packet was observed.

- First and last times when packets were seen for this flow.

- Counts for 'forward' (source to destination) and 'backward' (destination to source) components of the flow's traffic.

- Other attributes, e.g. state of the flow record (discussed below).

The state of a flow record may be:

- INACTIVE: The flow record is not being used by the meter.

- CURRENT: The record is in use and describes a flow which belongs to the 'current flow set', i.e. the set of flows recently seen by the meter.

- IDLE: The record is in use and the flow which it describes is part of the current flow set. In addition, no packets belonging to this flow have been seen for a period specified by the meter's InactivityTime variable.

4.3 Packet Handling, Packet Matching

Each packet header received by the traffic meter program is processed as follows:

- Extract attribute values from the packet header and use them to create a MATCH KEY for the packet.

- Match the packet's key against the current rule set, as explained in detail below.

The rule set specifies whether the packet is to be counted or ignored. If it is to be counted the matching process produces a FLOW KEY for the flow to which the packet belongs. This flow key is used to find the flow's record in the flow table; if a record does not yet exist for this flow, a new flow record may be created. The data for the matching flow record can then be updated.

For example, the rule set could specify that packets to or from any host in IP network 130.216 are to be counted. It could also specify that flow records are to be created for every pair of 24-bit (Class C) subnets within network 130.216.

Each packet's match key is passed to the meter's PATTERN MATCHING ENGINE (PME) for matching. The PME is a Virtual Machine which uses a set of instructions called RULES, i.e. a RULE SET is a program for the PME. A packet's match key contains source (S) and destination (D) interface identities, address values and masks.

If measured flows were unidirectional, i.e. only counted packets travelling in one direction, the matching process would be simple. The PME would be called once to match the packet. Any flow key produced by a successful match would be used to find the flow's record in the flow table, and that flow's counters would be updated.

Flows are, however, bidirectional, reflecting the forward and reverse packets of a protocol interchange or 'session'. Maintaining two sets of counters in the meter's flow record makes the resulting flow data much simpler to handle, since analysis programs do not have to gather together the 'forward' and 'reverse' components of sessions. Implementing bi-directional flows is, of course, more difficult for the meter, since it must decide whether a packet is a 'forward' packet or a 'reverse' one. To make this decision the meter will often need to invoke the PME twice, once for each possible packet direction.

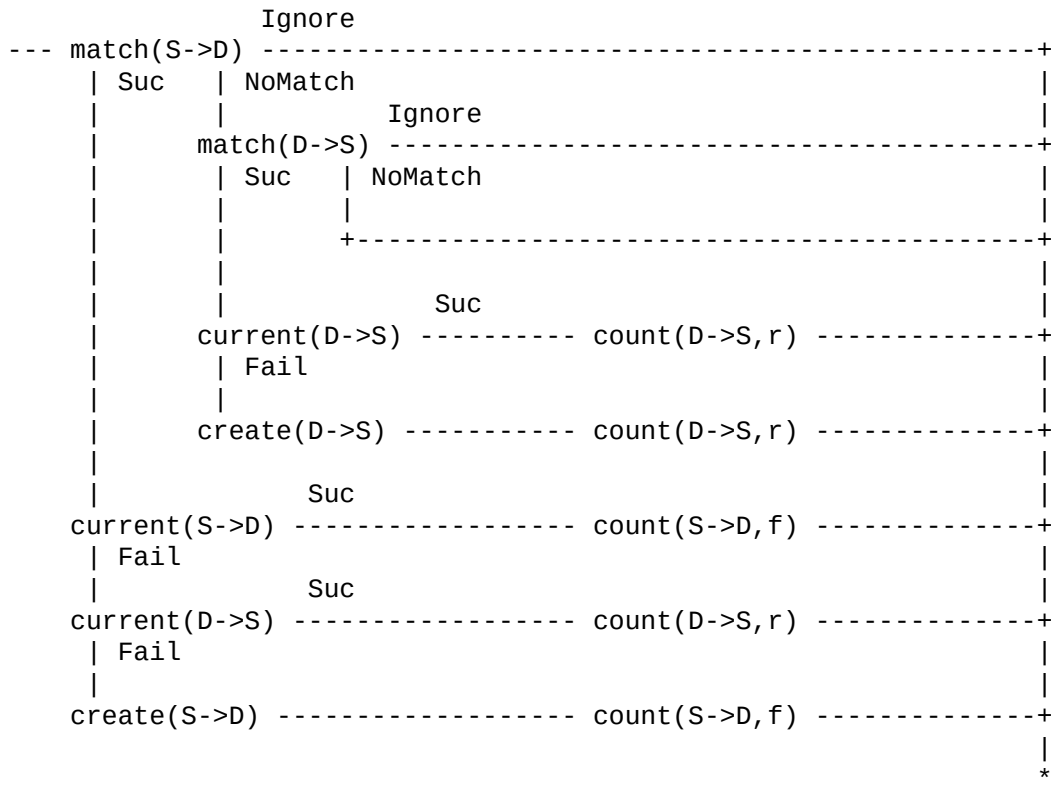
The diagram below describes the algorithm used by the traffic meter to process each packet. Flow through the diagram is from left to right and top to bottom, i.e. from the top left corner to the bottom right corner. S indicates the flow's source address (i.e. its set of source address attribute values) from the packet header, and D indicates its destination address.

There are several cases to consider. These are:

- The packet is recognised as one which is TO BE IGNORED.

- The packet would MATCH IN EITHER DIRECTION. One situation in which this could happen would be a rule set which matches flows within network X (Source = X, Dest = X) but specifies that flows are to be created for each subnet within network X, say subnets y and z. If, for example a packet is seen for y->z, the meter must check that flow z->y is not already current before creating y->z.

- The packet MATCHES IN ONE DIRECTION ONLY. If its flow is already current, its forward or reverse counters are incremented. Otherwise it is added to the flow table and then counted.



The algorithm uses four functions, as follows:

match(A->B) implements the PME. It uses the meter's current rule set to match the attribute values in the packet's match key. A->B means that the assumed source address is A and destination address B, i.e. that the packet was travelling from A to B. match() returns one of three results:

'Ignore' means that the packet was matched but this flow is not to be counted.

'NoMatch' means that the packet did not match. It might, however match with its direction reversed, i.e. from B to A.

'Suc' means that the packet did match, i.e. it belongs to a flow which is to be counted.

current(A->B) succeeds if the flow A-to-B is current - i.e. has a record in the flow table whose state is Current - and fails otherwise.

create(A->B) adds the flow A-to-B to the flow table, setting the value for attributes - such as addresses - which remain constant, and zeroing the flow's counters.

count(A->B,f) increments the 'forward' counters for flow A-to-B.
count(A->B,r) increments the 'reverse' counters for flow A-to-B.
'Forward' here means the counters for packets travelling from A to B. Note that count(A->B,f) is identical to count(B->A,r).

When writing rule sets one must remember that the meter will normally try to match each packet in the reverse direction if the forward match does not succeed. It is particularly important that the rule set does not contain inconsistencies which will upset this process.

Consider, for example, a rule set which counts packets from source network A to destination network B, but which ignores packets from source network B. This is an obvious example of an inconsistent rule set, since packets from network B should be counted as reverse packets for the A-to-B flow.

This problem could be avoided by devising a language for specifying rule files and writing a compiler for it, thus making it much easier to produce correct rule sets. An example of such a language is described in the 'SRL' document [RTFM-SRL]. Another approach would be to write a 'rule set consistency checker' program, which could detect problems in hand-written rule sets.

Normally, the best way to avoid these problems is to write rule sets which only classify flows in the forward direction, and rely on the meter to handle reverse-travelling packets.

Occasionally there can be situations when a rule set needs to know the direction in which a packet is being matched. Consider, for example, a rule set which wants to save some attribute values (source and destination addresses perhaps) for any 'unusual' packets. The rule set will contain a sequence of tests for all the 'usual' source addresses, follwed by a rule which will execute a 'NoMatch' action. If the match fails in the S->D direction, the NoMatch action will cause it to be retried. If it fails in the D->S direction, the packet can be counted as an 'unusual' packet.

To count such an 'unusual' packet we need to know the matching direction: the MatchingStoD attribute provides this. To use it, one follows the source address tests with a rule which tests whether the matching direction is S->D (MatchingStoD value is 1). If so, a 'NoMatch' action is executed. Otherwise, the packet has failed to match in both directions; we can save whatever attribute values are of interest and count the 'unusual' packet.

4.4 Rules and Rule Sets

A rule set is an array of rules. Rule sets are held within a meter as entries in an array of rule sets.

Rule set 1 (the first entry in the rule set table) is built-in to the meter and cannot be changed. It is run when the meter is started up, and provides a very coarse reporting granularity; it is mainly useful for verifying that the meter is running, before a 'useful' rule set is downloaded to it.

A meter also maintains an array of 'tasks', which specify what rule sets the meter is running. Each task has a 'current' rule set (the one which it normally uses), and a 'standby' rule set (which will be used when the overall traffic level is unusually high). If a task is instructed to use rule set 0, it will cease measuring; all packets will be ignored until another (non-zero) rule set is made current.

Each rule in a rule set is an instruction for the Packet Matching Engine, i.e. it is an instruction for a Virtual Machine. PME instructions have five component fields, forming two logical groups as follows:

```
+-------- test ---------+    +---- action -----+
attribute & mask = value:    opcode,  parameter;
```

The test group allows PME to test the value of an attribute. This is done by ANDing the attribute value with the mask and comparing the result with the value field. Note that there is no explicit provision to test a range, although this can be done where the range can be covered by a mask, e.g. attribute value less than 2048.

The PME maintains a Boolean indicator called the 'test indicator', which determines whether or not a rule's test is performed. The test indicator is initially set (true).

The action group specifies what action may be performed when the rule is executed. Opcodes contain two flags: 'goto' and 'test', as detailed in the table below. Execution begins with rule 1, the first in the rule set. It proceeds as follows:

```
If the test indicator is true:
   Perform the test, i.e. AND the attribute value with the
      mask and compare it with the value.
   If these are equal the test has succeeded; perform the
      rule's action (below).
   If the test fails execute the next rule in the rule set.
   If there are no more rules in the rule set, return from the
      match() function indicating NoMatch.

If the test indicator is false, or the test (above) succeeded:
   Set the test indicator to this opcode's test flag value.
   Determine the next rule to execute.
      If the opcode has its goto flag set, its parameter value
         specifies the number of the next rule.
      Opcodes which don't have their goto flags set either
         determine the next rule in special ways (Return),
         or they terminate execution (Ignore, NoMatch, Count,
         CountPkt).
   Perform the action.
```

The PME maintains two 'history' data structures. The first, the 'return' stack, simply records the index (i.e. 1-origin rule number) of each Gosub rule as it is executed; Return rules pop their Gosub rule index. Note that when the Ignore, NoMatch, Count and CountPkt actions are performed, PME execution is terminated regardless of whether the PME is executing a subroutine ('return' stack is non-empty) or not.

The second data structure, the 'pattern' queue, is used to save information for later use in building a flow key. A flow key is built by zeroing all its attribute values, then copying attribute number, mask and value information from the pattern queue in the order it was enqueued.

An attribute number identifies the attribute actually used in a test. It will usually be the rule's attribute field, unless the attribute is a 'meter variable'. Details of meter variables are given after the table of opcode actions below.

The opcodes are:

	opcode	goto	test
1	Ignore	0	-
2	NoMatch	0	-
3	Count	0	-
4	CountPkt	0	-
5	Return	0	0
6	Gosub	1	1
7	GosubAct	1	0
8	Assign	1	1
9	AssignAct	1	0
10	Goto	1	1
11	GotoAct	1	0
12	PushRuleTo	1	1
13	PushRuleToAct	1	0
14	PushPktTo	1	1
15	PushPktToAct	1	0
16	PopTo	1	1
17	PopToAct	1	0

The actions they perform are:

Ignore: Stop matching, return from the match() function indicating that the packet is to be ignored.

NoMatch: Stop matching, return from the match() function indicating failure.

Count: Stop matching. Save this rule's attribute number, mask and value in the PME's pattern queue, then construct a flow key for the flow to which this packet belongs. Return from the match() function indicating success. The meter will use the flow key to search for the flow record for this packet's flow.

CountPkt: As for Count, except that the masked value from the packet header (as it would have been used in the rule's test) is saved in the PME's pattern queue instead of the rule's value.

Gosub: Call a rule-matching subroutine. Push the current rule number on the PME's return stack, set the test indicator then goto the specified rule.

GosubAct: Same as Gosub, except that the test indicator is cleared before going to the specified rule.

Return: Return from a rule-matching subroutine. Pop the number of the calling gosub rule from the PME's 'return' stack and add this rule's parameter value to it to determine the 'target' rule. Clear the test indicator then goto the target rule.

A subroutine call appears in a rule set as a Gosub rule followed by a small group of following rules. Since a Return action clears the test flag, the action of one of these 'following' rules will be executed; this allows the subroutine to return a result (in addition to any information it may save in the PME's pattern queue).

Assign: Set the attribute specified in this rule to the parameter value specified for this rule. Set the test indicator then goto the specified rule.

AssignAct: Same as Assign, except that the test indicator is cleared before going to the specified rule.

Goto: Set the test indicator then goto the specified rule.

GotoAct: Clear the test indicator then goto the specified rule.

PushRuleTo: Save this rule's attribute number, mask and value in the PME's pattern queue. Set the test indicator then goto the specified rule.

PushRuleToAct: Same as PushRuleTo, except that the test indicator is cleared before going to the specified rule.

PushRuleTo actions may be used to save the value and mask used in a test, or (if the test is not performed) to save an arbitrary value and mask.

PushPktTo: Save this rule's attribute number, mask, and the masked value from the packet header (as it would have been used in the rule's test), in the PME's pattern queue. Set the test indicator then goto the specified rule.

PushPktToAct: Same as PushPktTo, except that the test indicator is cleared before going to the specified rule.

PushPktTo actions may be used to save a value from the packet header using a specified mask. The simplest way to program this is to use a zero value for the PushPktTo rule's value field, and to GoToAct to the PushPktTo rule (so that it's test is not executed).

PopTo: Delete the most recent item from the pattern queue, so as to remove the information saved by an earlier 'push' action. Set the test indicator then goto the specified rule.

PopToAct: Same as PopTo, except that the test indicator is cleared before going to the specified rule.

As well as the attributes applying directly to packets (such as SourcePeerAddress, DestTransAddress, etc.) the PME implements several further attribtes. These are:

Null: Tests performed on the Null attribute always succeed.

MatchingStoD: Indicates whether the PME is matching the packet with its addresses in 'wire order' or with its addresses reversed. MatchingStoD's value is 1 if the addresses are in wire order (StoD), and zero otherwise.

v1 .. v5: v1, v2, v3, v4 and v5 are 'meter variables'. They provide a way to pass parameters into rule-matching subroutines. Each may hold the number of a normal attribute; its value is set by an Assign action. When a meter variable appears as the attribute of a rule, its value specifies the actual attribute to be tested. For example, if v1 had been assigned SourcePeerAddress as its value, a rule with v1 as its attribute would actually test SourcePeerAddress.

SourceClass, DestClass, FlowClass,
SourceKind, DestKind, FlowKind:
These six attributes may be set by executing PushRuleTo actions. They allow the PME to save (in flow records) information which has been built up during matching. Their values may be tested in rules; this allows one to set them early in a rule set, and test them later.

The opcodes detailed above (with their above 'goto' and 'test' values) form a minimum set, but one which has proved very effective in current meter implementations. From time to time it may be useful to add further opcodes; IANA considerations for allocating these are set out in section 8 below.

4.5 Maintaining the Flow Table

The flow table may be thought of as a 1-origin array of flow records. (A particular implementation may, of course, use whatever data structure is most suitable). When the meter starts up there are no known flows; all the flow records are in the 'inactive' state.

Each time a packet is matched for a flow which is not in a current flow set a flow record is created for it; the state of such a record is
'current'. When selecting a record for the new flow the meter searches the flow table for an 'inactive' record. If no inactive records are available it will search for an 'idle' one instead. Note that there is no particular significance in the ordering of records within the flow table.

A meter's memory management routines should aim to minimise the time spent finding flow records for new flows, so as to minimise the setup overhead associated with each new flow.

Flow data may be collected by a 'meter reader' at any time. There is no requirement for collections to be synchronized. The reader may collect the data in any suitable manner, for example it could upload a copy of the whole flow table using a file transfer protocol, or it could read the records in the current flow set row by row using a suitable data transfer protocol.

The meter keeps information about collections, in particular it maintains ReaderLastTime variables which remember the time the last collection was made by each reader. A second variable, InactivityTime, specifies the minimum time the meter will wait before considering that a flow is idle.

The meter must recover records used for idle flows, if only to prevent it running out of flow records. Recovered flow records are returned to the 'inactive' state. A variety of recovery strategies are possible, including the following:

One possible recovery strategy is to recover idle flow records as soon as possible after their data has been collected by all readers which have registered to do so. To implement this the meter could run a background process which scans the flow table looking for 'current' flows whose 'last packet' time is earlier than the meter's LastCollectTime.

Another recovery strategy is to leave idle flows alone as long as possible, which would be acceptable if one was only interested in measuring total traffic volumes. It could be implemented by having the meter search for collected idle flows only when it ran low on 'inactive' flow records.

One further factor a meter should consider before recovering a flow is the number of meter readers which have collected the flow's data. If there are multiple meter readers operating, each reader should collect a flow's data before its memory is recovered.

Of course a meter reader may fail, so the meter cannot wait forever for it. Instead the meter must keep a table of active meter readers, with a timeout specified for each. If a meter reader fails to collect flow data within its timeout interval, the meter should delete that reader from the meter's active meter reader table.

4.6 Handling Increasing Traffic Levels

Under normal conditions the meter reader specifies which set of usage records it wants to collect, and the meter provides them. If, however, memory usage rises above the high-water mark the meter should switch to a STANDBY RULE SET so as to decrease the rate at which new flows are created.

When the manager, usually as part of a regular poll, becomes aware that the meter is using its standby rule set, it could decrease the interval between collections. This would shorten the time that flows sit in memory waiting to be collected, allowing the meter to free flow memory faster.

The meter could also increase its efforts to recover flow memory so as to reduce the number of idle flows in memory. When the situation returns to normal, the manager may request the meter to switch back to its normal rule set.

5 Meter Readers

Usage data is accumulated by a meter (e.g. in a router) as memory permits. It is collected at regular reporting intervals by meter readers, as specified by a manager. The collected data is recorded in stable storage as a FLOW DATA FILE, as a sequence of USAGE RECORDS.

The following sections describe the contents of usage records and flow data files. Note, however, that at this stage the details of such records and files is not specified in the architecture. Specifying a common format for them would be a worthwhile future development.

5.1 Identifying Flows in Flow Records

Once a packet has been classified and is ready to be counted, an appropriate flow data record must already exist in the flow table; otherwise one must be created. The flow record has a flexible format where unnecessary identification attributes may be omitted. The determination of which attributes of the flow record to use, and of what values to put in them, is specified by the current rule set.

Note that the combination of start time, rule set number and flow subscript (row number in the flow table) provide a unique flow identifier, regardless of the values of its other attributes.

The current rule set may specify additional information, e.g. a computed attribute value such as FlowKind, which is to be placed in the attribute section of the usage record. That is, if a particular flow is matched by the rule set, then the corresponding flow record should be marked not only with the qualifying identification attributes, but also with the additional information. Using this feature, several flows may each carry the same FlowKind value, so that the resulting usage records can be used in post-processing or between meter reader and meter as a criterion for collection.

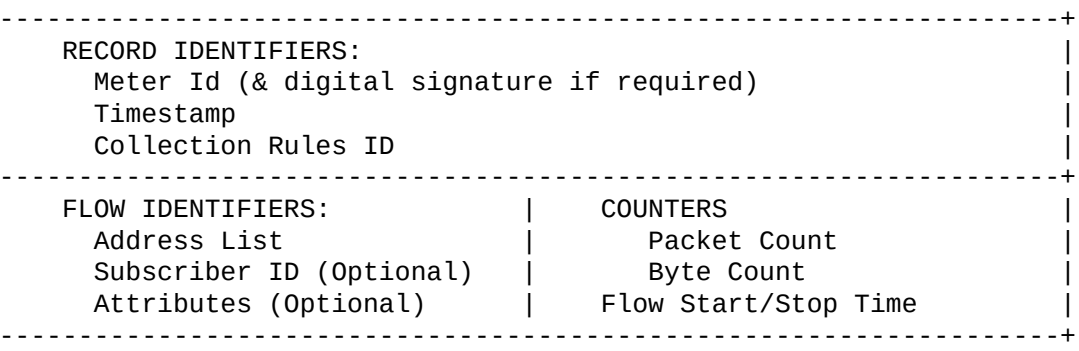
5.2 Usage Records, Flow Data Files

The collected usage data will be stored in flow data files on the meter reader, one file for each meter. As well as containing the measured usage data, flow data files must contain information uniquely identifiying the meter from which it was collected.

A USAGE RECORD contains the descriptions of and values for one or more flows. Quantities are counted in terms of number of packets and number of bytes per flow. Other quantities, e.g. short-term flow rates, may be added later; work on such extensions is described in the RTFM 'New Attributes' document [RTFM-NEW].

Each usage record contains the metered traffic group identifier of the meter (a set of network addresses), a time stamp and a list of reported flows (FLOW DATA RECORDS). A meter reader will build up a file of usage records by regularly collecting flow data from a meter, using this data to build usage records and concatenating them to the tail of a file. Such a file is called a FLOW DATA FILE.

A usage record contains the following information in some form:



5.3 Meter to Meter Reader: Usage Record Transmission

The usage record contents are the raison d'etre of the system. The accuracy, reliability, and security of transmission are the primary concerns of the meter/meter reader exchange. Since errors may occur on networks, and Internet packets may be dropped, some mechanism for ensuring that the usage information is transmitted intact is needed.

Flow data is moved from meter to meter reader via a series of protocol exchanges between them. This may be carried out in various ways, moving individual attribute values, complete flows, or the entire flow table (i.e. all the active and idle flows). One possible method of achieving this transfer is to use SNMP; the 'Traffic Flow Measurement: Meter MIB' RFC [RTFM-MIB] gives details. Note that this is simply one example; the transfer of flow data from meter to meter reader is not specified in this document.

The reliability of the data transfer method under light, normal, and extreme network loads should be understood before selecting among collection methods.

In normal operation the meter will be running a rule file which provides the required degree of flow reporting granularity, and the meter reader(s) will collect the flow data often enough to allow the meter's garbage collection mechanism to maintain a stable level of memory usage.

In the worst case traffic may increase to the point where the meter is in danger of running completely out of flow memory. The meter implementor must decide how to handle this, for example by switching to a default (extremely coarse granularity) rule set, by sending a trap message to the manager, or by attempting to dump flow data to the meter reader.

Users of the Traffic Flow Measurement system should analyse their requirements carefully and assess for themselves whether it is more important to attempt to collect flow data at normal granularity (increasing the collection frequency as needed to keep up with traffic volumes), or to accept flow data with a coarser granularity. Similarly, it may be acceptable to lose flow data for a short time in return for being sure that the meter keeps running properly, i.e. is not overwhelmed by rising traffic levels.

6 Managers

A manager configures meters and controls meter readers. It does this via the interactions described below.

6.1 Between Manager and Meter: Control Functions

- DOWNLOAD RULE SET: A meter may hold an array of rule sets. One of these, the 'default' rule set, is built in to the meter and cannot be changed; this is a diagnostic feature, ensuring that when a meter starts up it will be running a known ruleset.

  All other rule sets must be downloaded by the manager. A manager may use any suitable protocol exchange to achieve this, for example an FTP file transfer or a series of SNMP SETs, one for each row of the rule set.

- SPECIFY METER TASK: Once the rule sets have been downloaded, the manager must instruct the meter which rule sets will be the 'current' and 'standby' ones for each task the meter is to perform.

- SET HIGH WATER MARK: A percentage of the flow table capacity, used by the meter to determine when to switch to its standby rule set (so as to increase the granularity of the flows and conserve the meter's flow memory). Once this has happened, the manager

may also change the polling frequency or the meter's control parameters (so as to increase the rate at which the meter can recover memory from idle flows). The meter has a separate high water mark value for each task it is currently running.

If the high traffic levels persist, the meter's normal rule set may have to be rewritten to permanently reduce the reporting granularity.

- SET FLOW TERMINATION PARAMETERS: The meter should have the good sense in situations where lack of resources may cause data loss to purge flow records from its tables. Such records may include:

  - Flows that have already been reported to all registered meter readers, and show no activity since the last report,
  - Oldest flows, or
  - Flows with the smallest number of observed packets.

- SET INACTIVITY TIMEOUT: This is a time in seconds since the last packet was seen for a flow. Flow records may be reclaimed if they have been idle for at least this amount of time, and have been collected in accordance with the current collection criteria.

It might be useful if a manager could set the FLOW TERMINATION PARAMETERS to different values for different tasks. Current meter implementations have only single ('whole meter') values for these parameters, and experience to date suggests that this provides an adequate degree of control for the tasks.

6.2 Between Manager and Meter Reader: Control Functions

Because there are a number of parameters that must be set for traffic flow measurement to function properly, and viable settings may change as a result of network traffic characteristics, it is desirable to have dynamic network management as opposed to static meter configurations. Many of these operations have to do with space tradeoffs - if memory at the meter is exhausted, either the collection interval must be decreased or a coarser granularity of aggregation must be used to reduce the number of active flows.

Increasing the collection interval effectively stores data in the meter; usage data in transit is limited by the effective bandwidth of the virtual link between the meter and the meter reader, and since these limited network resources are usually also used to carry user data (the purpose of the network), the level of traffic flow measurement traffic should be kept to an affordable fraction of the bandwidth. ("Affordable" is a policy decision made by the Network

Operations personnel). At any rate, it must be understood that the operations below do not represent the setting of independent variables; on the contrary, each of the values set has a direct and measurable effect on the behaviour of the other variables.

Network management operations follow:

- MANAGER and METER READER IDENTIFICATION: The manager should ensure that meters are read by the correct set of meter readers, and take steps to prevent unauthorised access to usage information. The meter readers so identified should be prepared to poll if necessary and accept data from the appropriate meters. Alternate meter readers may be identified in case both the primary manager and the primary meter reader are unavailable. Similarly, alternate managers may be identified.

- REPORTING INTERVAL CONTROL: The usual reporting interval should be selected to cope with normal traffic patterns. However, it may be possible for a meter to exhaust its memory during traffic spikes even with a correctly set reporting interval. Some mechanism should be available for the meter to tell the manager that it is in danger of exhausting its memory (by declaring a ' high water' condition), and for the manager to arbitrate (by decreasing the polling interval, letting nature take its course, or by telling the meter to ask for help sooner next time).

- GRANULARITY CONTROL: Granularity control is a catch-all for all the parameters that can be tuned and traded to optimise the system's ability to reliably measure and store information on all the traffic (or as close to all the traffic as an administration requires). Granularity:

    - Controls the amount of address information identifying each flow, and
    - Determines the number of buckets into which user traffic will be lumped together.

  Since granularity is controlled by the meter's current rule set, the manager can only change it by requesting the meter to switch to a different rule set. The new rule set could be downloaded when required, or it could have been downloaded as part of the meter's initial configuration.

- FLOW LIFETIME CONTROL: Flow termination parameters include timeout parameters for obsoleting inactive flows and removing them from tables, and maximum flow lifetimes. This is intertwined with reporting interval and granularity, and must be set in accordance with the other parameters.

6.3 Exception Conditions

Exception conditions must be handled, particularly occasions when the meter runs out of space for flow data. Since - to prevent an active task from counting any packet twice - packets can only be counted in a single flow, discarding records will result in the loss of information. The mechanisms to deal with this are as follows:

- METER OUTAGES: In case of impending meter outages (controlled restarts, etc.) the meter could send a trap to the manager. The manager could then request one or more meter readers to pick up the data from the meter.

  Following an uncontrolled meter outage such as a power failure, the meter could send a trap to the manager indicating that it has restarted. The manager could then download the meter's correct rule set and advise the meter reader(s) that the meter is running again. Alternatively, the meter reader may discover from its regular poll that a meter has failed and restarted. It could then advise the manager of this, instead of relying on a trap from the meter.

- METER READER OUTAGES: If the collection system is down or isolated, the meter should try to inform the manager of its failure to communicate with the collection system. Usage data is maintained in the flows' rolling counters, and can be recovered when the meter reader is restarted.

- MANAGER OUTAGES: If the manager fails for any reason, the meter should continue measuring and the meter reader(s) should keep gathering usage records.

- BUFFER PROBLEMS: The network manager may realise that there is a 'low memory' condition in the meter. This can usually be attributed to the interaction between the following controls:

  - The reporting interval is too infrequent, or
  - The reporting granularity is too fine.

Either of these may be exacerbated by low throughput or bandwidth of circuits carrying the usage data. The manager may change any of these parameters in response to the meter (or meter reader's) plea for help.

6.4 Standard Rule Sets

Although the rule table is a flexible tool, it can also become very complex. It may be helpful to develop some rule sets for common applications:

- PROTOCOL TYPE: The meter records packets by protocol type. This will be the default rule table for Traffic Flow Meters.

- ADJACENT SYSTEMS: The meter records packets by the MAC address of the Adjacent Systems (neighbouring originator or next-hop). (Variants on this table are "report source" or "report sink" only.) This strategy might be used by a regional or backbone network which wants to know how much aggregate traffic flows to or from its subscriber networks.

- END SYSTEMS: The meter records packets by the IP address pair contained in the packet. (Variants on this table are "report source" or "report sink" only.) This strategy might be used by an End System network to get detailed host traffic matrix usage data.

- TRANSPORT TYPE: The meter records packets by transport address; for IP packets this provides usage information for the various IP services.

- HYBRID SYSTEMS: Combinations of the above, e.g. for one interface report End Systems, for another interface report Adjacent Systems. This strategy might be used by an enterprise network to learn detail about local usage and use an aggregate count for the shared regional network.

7 Security Considerations

7.1 Threat Analysis

A traffic flow measurement system may be subject to the following kinds of attacks:

- ATTEMPTS TO DISABLE A TRAFFIC METER: An attacker may attempt to disrupt traffic measurement so as to prevent users being charged for network usage. For example, a network probe sending packets

to a large number of destination and transport addresses could produce a sudden rise in the number of flows in a meter's flow table, thus forcing it to use its coarser standby rule set.

- UNAUTHORIZED USE OF SYSTEM RESOURCES: An attacker may wish to gain advantage or cause mischief (e.g. denial of service) by subverting any of the system elements - meters, meter readers or managers.

- UNAUTHORIZED DISCLOSURE OF DATA: Any data that is sensitive to disclosure can be read through active or passive attacks unless it is suitably protected. Usage data may or may not be of this type. Control messages, traps, etc. are not likely to be considered sensitive to disclosure.

- UNAUTHORIZED ALTERATION, REPLACEMENT OR DESTRUCTION OF DATA: Similarly, any data whose integrity is sensitive can be altered, replaced/injected or deleted through active or passive attacks unless it is suitably protected. Attackers may modify message streams to falsify usage data or interfere with the proper operation of the traffic flow measurement system. Therefore, all messages, both those containing usage data and those containing control data, should be considered vulnerable to such attacks.

7.2 Countermeasures

The following countermeasures are recommended to address the possible threats enumerated above:

- ATTEMPTS TO DISABLE A TRAFFIC METER can't be completely countered. In practice, flow data records from network security attacks have proved very useful in determining what happened. The most effective approach is first to configure the meter so that it has three or more times as much flow memory as it needs in normal operation, and second to collect the flow data fairly frequently so as to minimise the time needed to recover flow memory after such an attack.

- UNAUTHORIZED USE OF SYSTEM RESOURCES is countered through the use of authentication and access control services.

- UNAUTHORIZED DISCLOSURE OF DATA is countered through the use of a confidentiality (encryption) service.

- UNAUTHORIZED ALTERATION, REPLACEMENT OR DESTRUCTION OF DATA is countered through the use of an integrity service.

A Traffic Measurement system must address all of these concerns. Since a high degree of protection is required, the use of strong cryptographic methodologies is recommended. The security requirements for communication between pairs of traffic measurmement system elements are summarized in the table below. It is assumed that meters do not communicate with other meters, and that meter readers do not communicate directly with other meter readers (if synchronization is required, it is handled by the manager, see Section 2.5). Each entry in the table indicates which kinds of security services are required. Basically, the requirements are as follows:

Security Service Requirements for RTFM elements

from\to	meter	meter reader	application	manager
meter	N/A	authent acc ctrl integrity confid **	N/A	authent acc ctrl
meter reader	authent acc ctrl	N/A	authent acc ctrl integrity confid **	authent acc ctrl
appl	N/A	authent acc ctrl	##	##
manager	authent acc ctrl integrity	authent acc ctrl integrity	##	authent acc ctrl integrity

N/A = Not Applicable ** = optional ## = outside RTFM scope

- When any two elements intercommunicate they should mutually authenticate themselves to one another. This is indicated by ’ authent’ in the table. Once authentication is complete, an element should check that the requested type of access is allowed; this is indicated on the table by ’acc ctrl’.

- Whenever there is a transfer of information its integrity should be protected.

- Whenever there is a transfer of usage data it should be possible to ensure its confidentiality if it is deemed sensitive to disclosure. This is indicated by 'confid' in the table.

Security protocols are not specified in this document. The system elements' management and collection protocols are responsible for providing sufficient data integrity, confidentiality, authentication and access control services.

8 IANA Considerations

The RTFM Architecture, as set out in this document, has two sets of assigned numbers. Considerations for assigning them are discussed in this section, using the example policies as set out in the "Guidelines for IANA Considerations" document [IANA-RFC].

8.1 PME Opcodes

The Pattern Matching Engine (PME) is a virtual machine, executing RTFM rules as its instructions. The PME opcodes appear in the 'action' field of an RTFM rule. The current list of opcodes, and their values for the PME's 'goto' and 'test' flags, are set out in section 4.4 above ("Rules and Rulesets).

The PME opcodes are pivotal to the RTFM architecture, since they must be implemented in every RTFM meter. Any new opcodes must therefore be allocated through an IETF Consensus action [IANA-RFC].

Opcodes are simply non-negative integers, but new opcodes should be allocated sequentially so as to keep the total opcode range as small as possible.

8.2 RTFM Attributes

Attribute numbers in the range of 0-511 are globally unique and are allocated according to an IETF Consensus action [IANA-RFC]. Appendix C of this document allocates a basic (i.e. useful minimum) set of attribtes; they are assigned numbers in the range 0 to 63. The RTFM working group is working on an extended set of attributes, which will have numbers in the range 64 to 127.

Vendor-specific attribute numbers are in the range 512-1023, and will be allocated using the First Come FIrst Served policy [IANA-RFC]. Vendors requiring attribute numbers should submit a request to IANA giving the attribute names: IANA will allocate them the next available numbers.

Attribute numbers 1024 and higher are Reserved for Private Use [IANA-RFC]. Implementors wishing to experiment with further new attributes should use attribute numbers in this range.

Attribute numbers are simply non-negative integers. When writing specifications for attributes, implementors must give sufficient detail for the new attributes to be easily added to the RTFM Meter MIB [RTFM-MIB]. In particular, they must indicate whether the new attributes may be:

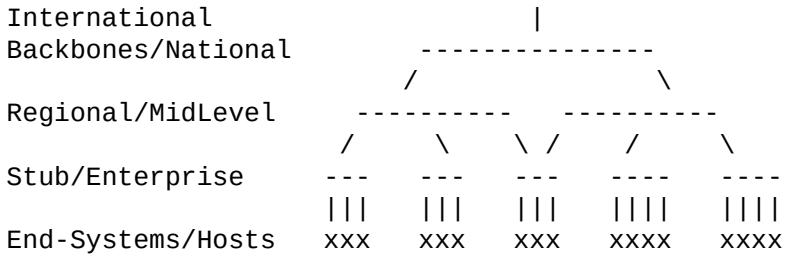
- tested in an IF statement
- saved by a SAVE statement or set by a STORE statement
- read from an RTFM meter

(IF, SAVE and STORE are statements in the SRL Ruleset Language [RTFM-SRL]).

9 APPENDICES

9.1 Appendix A: Network Characterisation

Internet users have extraordinarily diverse requirements. Networks differ in size, speed, throughput, and processing power, among other factors. There is a range of traffic flow measurement capabilities and requirements. For traffic flow measurement purposes, the Internet may be viewed as a continuum which changes in character as traffic passes through the following representative levels:



Note that mesh architectures can also be built out of these components, and that these are merely descriptive terms. The nature of a single network may encompass any or all of the descriptions below, although some networks can be clearly identified as a single type.

BACKBONE networks are typically bulk carriers that connect other networks. Individual hosts (with the exception of network management devices and backbone service hosts) typically are not directly connected to backbones.

REGIONAL networks are closely related to backbones, and differ only in size, the number of networks connected via each port, and geographical coverage. Regionals may have directly connected hosts, acting as hybrid backbone/stub networks. A regional network is a SUBSCRIBER to the backbone.

STUB/ENTERPRISE networks connect hosts and local area networks. STUB/ENTERPRISE networks are SUBSCRIBERS to regional and backbone networks.

END SYSTEMS, colloquially HOSTS, are SUBSCRIBERS to any of the above networks.

Providing a uniform identification of the SUBSCRIBER in finer granularity than that of end-system, (e.g. user/account), is beyond the scope of the current architecture, although an optional attribute in the traffic flow measurement record may carry system-specific

'user identification' labels so that meters can implement proprietary or non-standard schemes for the attribution of network traffic to responsible parties.

9.2 Appendix B: Recommended Traffic Flow Measurement Capabilities

Initial recommended traffic flow measurement conventions are outlined here according to the following Internet building blocks. It is important to understand what complexity reporting introduces at each network level. Whereas the hierarchy is described top-down in the previous section, reporting requirements are more easily addressed bottom-up.

End-Systems
Stub Networks
Enterprise Networks
Regional Networks
Backbone Networks

END-SYSTEMS are currently responsible for allocating network usage to end-users, if this capability is desired. From the Internet Protocol perspective, end-systems are the finest granularity that can be identified without protocol modifications. Even if a meter violated protocol boundaries and tracked higher-level protocols, not all packets could be correctly allocated by user, and the definition of user itself varies widely from operating system to operating system (e.g. how to trace network usage back to users from shared processes).

STUB and ENTERPRISE networks will usually collect traffic data either by end-system network address or network address pair if detailed reporting is required in the local area network. If no local reporting is required, they may record usage information in the exit router to track external traffic only. (These are the only networks which routinely use attributes to perform reporting at granularities finer than end-system or intermediate-system network address.)

REGIONAL networks are intermediate networks. In some cases, subscribers will be enterprise networks, in which case the intermediate system network address is sufficient to identify the regional's immediate subscriber. In other cases, individual hosts or a disjoint group of hosts may constitute a subscriber. Then end-system network address pairs need to be tracked for those subscribers. When the source may be an aggregate entity (such as a network, or adjacent router representing traffic from a world of hosts beyond) and the destination is a singular entity (or vice versa), the meter is said to be operating as a HYBRID system.

At the regional level, if the overhead is tolerable it may be advantageous to report usage both by intermediate system network address (e.g. adjacent router address) and by end-system network address or end-system network address pair.

BACKBONE networks are the highest level networks operating at higher link speeds and traffic levels. The high volume of traffic will in most cases preclude detailed traffic flow measurement. Backbone networks will usually account for traffic by adjacent routers' network addresses.

9.3 Appendix C: List of Defined Flow Attributes

This Appendix provides a checklist of the attributes defined to date; others will be added later as the Traffic Measurement Architecture is further developed.

Note that this table gives only a very brief summary. The Meter MIB [RTFM-MIB] provides the definitive specification of attributes and their allowed values. The MIB variables which represent flow attributes have 'flowData' prepended to their names to indicate that they belong to the MIB's flowData table.

0	Null		
4	SourceInterface	Integer	Source Address
5	SourceAdjacentType	Integer	
6	SourceAdjacentAddress	String	
7	SourceAdjacentMask	String	
8	SourcePeerType	Integer	
9	SourcePeerAddress	String	
10	SourcePeerMask	String	
11	SourceTransType	Integer	
12	SourceTransAddress	String	
13	SourceTransMask	String	
14	DestInterface	Integer	Destination Address
15	DestAdjacentType	Integer	
16	DestAdjacentAddress	String	
17	DestAdjacentMask	String	
18	DestPeerType	Integer	
19	DestPeerAddress	String	
20	DestPeerMask	String	
21	DestTransType	Integer	
22	DestTransAddress	String	
23	DestTransMask	String	

26	RuleSet	Integer	Meter attribute
27	ToOctets	Integer	Source-to-Dest counters
28	ToPDUs	Integer	
29	FromOctets	Integer	Dest-to-Source counters
30	FromPDUs	Integer	
31	FirstTime	Timestamp	Activity times
32	LastActiveTime	Timestamp	
33	SourceSubscriberID	String	Session attributes
34	DestSubscriberID	String	
35	SessionID	String	
36	SourceClass	Integer	'Computed' attributes
37	DestClass	Integer	
38	FlowClass	Integer	
39	SourceKind	Integer	
40	DestKind	Integer	
41	FlowKind	Integer	
50	MatchingStoD	Integer	PME variable
51	v1	Integer	Meter Variables
52	v2	Integer	
53	v3	Integer	
54	v4	Integer	
55	v5	Integer	
65 .. 127	'Extended' attributes (to be defined by the RTFM working group)		

9.4 Appendix D: List of Meter Control Variables

```
Meter variables:
   Flood Mark                     Percentage
   Inactivity Timeout (seconds)   Integer

'per task' variables:
   Current Rule Set Number        Integer
   Standby Rule Set Number        Integer
   High Water Mark                Percentage

'per reader' variables:
   Reader Last Time               Timestamp
```

9.5 Appendix E: Changes Introduced Since RFC 2063

The first version of the Traffic Flow Measurement Architecture was published as RFC 2063 in January 1997. The most significant changes made since then are summarised below.

- A Traffic Meter can now run multiple rule sets concurrently. This makes a meter much more useful, and required only minimal changes to the architecture.

- 'NoMatch' replaces 'Fail' as an action. This name was agreed to at the Working Group 1996 meeting in Montreal; it better indicates that although a particular match has failed, it may be tried again with the packet's addresses reversed.

- The 'MatchingStoD' attribute has been added. This is a Packet Matching Engine (PME) attribute indicating that addresses are being matched in StoD (i.e. 'wire') order. It can be used to perform different actions when the match is retried, thereby simplifying some kinds of rule sets. It was discussed and agreed to at the San Jose meeting in 1996.

- Computed attributes (Class and Kind) may now be tested within a rule set. This lifts an unneccessary earlier restriction.

- The list of attribute numbers has been extended to define ranges for 'basic' attributes (in this document) and 'extended' attributes (currently being developed by the RTFM Working Group).

- The 'Security Considerations' section has been completely rewritten. It provides an evaluation of traffic measurement security risks and their countermeasures.

10 Acknowledgments

An initial draft of this document was produced under the auspices of the IETF's Internet Accounting Working Group with assistance from SNMP, RMON and SAAG working groups. Particular thanks are due to Stephen Stibler (IBM Research) for his patient and careful comments during the preparation of this memo.

11 References

[802-3] IEEE 802.3/ISO 8802-3 Information Processing Systems - Local Area Networks - Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, 2nd edition, September 21, 1990.

[ACT-BKG] Mills, C., Hirsch, G. and G. Ruth, "Internet Accounting Background", RFC 1272, November 1991.

[IANA-RFC] Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[IPPM-FRM] Paxson, V., Almes, G., Mahdavi, J. and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.

[OSI-ACT] International Standards Organisation (ISO), "Management Framework", Part 4 of Information Processing Systems Open Systems Interconnection Basic Reference Model, ISO 7498-4, 1994.

[RTFM-MIB] Brownlee, N., "Traffic Flow Measurement: Meter MIB", RFC 2720, October 1999.

[RTFM-NEW] Handelman, S., Stibler, S., Brownlee, N. and G. Ruth, "RTFM: New Attributes for Traffic Flow Measurment", RFC 2724, October 1999.

[RTFM-SRL] Brownlee, N., "SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups", RFC 2723, October 1999.

12 Authors' Addresses

Nevil Brownlee
Information Technology Systems & Services
The University of Auckland
Private Bag 92-019
Auckland, New Zealand

Phone: +64 9 373 7599 x8941
EMail: n.brownlee@auckland.ac.nz

Cyndi Mills
GTE Laboratories, Inc
40 Sylvan Rd.
Waltham, MA 02451, U.S.A.

Phone: +1 781 466 4278
EMail: cmills@gte.com

Greg Ruth
GTE Internetworking
3 Van de Graaff Drive
P.O. Box 3073
Burlington, MA 01803, U.S.A.

Phone: +1 781 262 4831
EMail: gruth@bbn.com

13 Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.