Network Working Group N. Brownlee
Request for Comments: 2063 The University of Auckland
Category: Experimental C. Mills
BBN Systems and Technologies
G. Ruth
GTE Laboratories, Inc.
January 1997

Traffic Flow Measurement: Architecture

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. This memo does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Abstract

This document describes an architecture for the measurement and reporting of network traffic flows, discusses how this relates to an overall network traffic flow architecture, and describes how it can be used within the Internet. It is intended to provide a starting point for the Realtime Traffic Flow Measurement Working Group.

Table of Contents

1 Statement of Purpose and Scope

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/application at any network layer (e.g. network, transport, application layers).

- 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 in a simple manner, 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 reduces the volume of data to be obtained (and transmitted across the network for storage), and minimises the amount of processing required in traffic flow analysis applications.

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 so that specific protocol implementations may extend coverage to multi-protocol environments and to other protocol layers, such as usage measurement for application-level services. Use of the same model for both network- and application-level measurement may simplify the development of generic analysis applications which process and/or correlate any or all levels of traffic and usage information. Within this docuemt the term 'usage data' is used as a generic term for the data obtained using the traffic flow measurement architecture.

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 'Traffic Flow Measurement: Background' RFC [1].

2 Traffic Flow Measurement Architecture

A traffic flow measurement system is used by network Operations personnel for 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 [2]. Future extensions are anticipated as the model is refined to address additional protocol layers.

2.1 Meters and Traffic Flows

At the heart of the traffic measurement model are network entities called traffic METERS. Meters count certain attributes (such as numbers of packets and bytes) and classify them as belonging to ACCOUNTABLE ENTITIES using other attributes (such as source and destination addresses). An accountable entity is someone who (or something which) is responsible for some activitiy on the network. It may be a user, a host system, a network, a group of networks, etc, depending on the granularity specified by 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 'Traffic Flow Measurement: Background' RFC [1]. An important aspect of meters is that they provide a way of succinctly aggregating entity usage information.

For the purpose of traffic flow measurement we define the concept of a TRAFFIC FLOW, which is an artificial logical equivalent to a call or connection. A flow is a portion of traffic, delimited by a start and stop time, that was generated by a particular accountable entity. 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 end 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 ascribe it to accountable entities.

Usage information which is not deriveable 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 will 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 APPLICAIONS, 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
                   /       \
              2.3 /         \ 2.4
                 /           \
                /             \                       ANALYSIS
           METER   <----->   METER READER  <----->   APPLICATION
                     2.2                     2.7
```

- MANAGER: A traffic measurement manager is an application which configures 'meter' entities and controls 'meter reader' entities. It uses the data requirements of analysis applications to determine the appropriate configurations for each meter, and the proper operation of each 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 reliably 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, indicating how flow rates vary with 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. At the time of writing a meter can only be controlled by a single manager; in the future this restriction may be relaxed. 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 maximum size of its flow table, 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 rate. Normally every packet will be observed. It may sometimes be necessary to use sampling techniques to observe only some of the packets. (Sampling algorithms are not prescribed by the architecture; it should be noted that before using sampling one should verify the statistical validity of the algorithm used). Current experience with the measurement architecture shows that a carefully-designed and implemented meter compresses the data such that in normal LANs and WANs of today sampling is really not needed.

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 is 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 active flows, the whole flow table, flows seen 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

```
                 -- METER READER A --
                /         |          \
               /          |           \
       =====METER 1     METER 2=====METER 3    METER 4=====
                           \           |          /
                            \          |         /
                             -- METER READER B --
```

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 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.

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 neccesarily 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 precisely synchronized collections are required this can be achieved by having one manager request each meter to begin collecting a new set of flows, then allowing all meter readers to collect the usage data from the old sets of flows.

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 an ACCOUNTABLE ENTITY."

In practical terms, a flow is a stream of packets passing across a network between two end points (or being sent 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), 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. information computed by the meter.

A flow's ACCOUNTABLE ENTITY is specified by the values of its ADDRESS attributes. For example, if a flow's address attributes specified only that "source address = IP address 10.1.0.1," then all IP packets from and to that address would be counted in that flow. If a flow's address list were specified as "source address = IP address 10.1.0.1, destination address = IP address 26.1.0.1" then only IP packets between 10.1.0.1 and 26.1.0.1 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 [n-1] layer address of the immediate source or destination on the path of the packet. For example, if flow measurement is being performed at the IP layer on an Ethernet LAN [3], an adjacent address is 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 PEER-LEVEL packet. The form of a peer address will depend on the network-layer protocol in use, and the network layer [n] at which traffic measurement is being performed.

- 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. Further work is needed here, particularly in selecting attributes which will be suitable for the higher layers of the OSI reference model.

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 'Traffic Flow Measurement: Background' RFC [1]. 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. mobile subscribers), further subscriber identification will be necessary if flows are to ascribed to individual users. Provision is made to further specify the accountable entity 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 proprietary means within a meter, for example via protocol exchanges with one or more (multi-user) hosts. At this time a subscriber ID is an arbitrary text string; later versions of the architecture may specify its contents on more detail.

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 the level of detail supplied. 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 the accountable entity, the attributes, or the time intervals. 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.

Flow granularity is controlled by adjusting the level of detail at which the following are reported:

- The accountable entity (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 accountable entity 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 ID. Granularity of accountable entities is further specified by additional ATTRIBUTES. These attributes include:

- Meter variables such as the index of the flow's record in the flow table and the rule set id 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.

- 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.

- Higher-layer (especially application-level) attributes. 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.

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 must be sure that the flow's data has been collected by at least one meter reader.

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 an 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, or MAX LIFETIME policy), 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.

Each packet is counted in one and only one flow, 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.

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.' 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 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.

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 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.

The discussion in the following sections assumes that a meter may only run a single rule set. It is, however, possible for a meter to run several rule sets concurrently, matching each packet against every active rule set and producing a single flow table with flows from all the active rule sets. The overall effect of doing this would be similar to running several independent meters, one for each rule set.

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, 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 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.

```
                  packet                +------------------+
                  header                | Current Rule Set |
                    |                   +--------+---------+
                    |                            |
           +--------*---------+       +----------*-------------+
           | Packet Processor |<----->| Packet Matching Engine |
           +--+------------+--+       +------------------------+
              |            |
       Ignore *            | Count via flow key
                           |
                        +--*--------------+
                        | ’Search’ index  |
                        +--------+--------+
                                 |
                        +--------*--------+
                        |                 |
                        |   Flow Table    |
                        |                 |
                        +--------+--------+
                                 |
                        +--------*--------+
                        | ’Collect’ index |
                        +--------+--------+
                                 |
                                 *
                            Meter Reader
```

4.2 Flow Table

Every traffic meter maintains a table of TRAFFIC FLOW RECORDS for flows seen by the meter. 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), and the number 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 counts for the matching flow record can then be incremented.

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 an interface number, source address (S) and destination address (D) values. It does not, however, contain any attribute masks for its attributes, only their values.

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, 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 MATCHES IN BOTH DIRECTIONS. 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.

'Fail' 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).

```
                    Ignore
    --- match(S->D) -------------------------------------------------+
         | Suc   | Fail                                              |
         |       |          Ignore                                   |
         |      match(D->S) -----------------------------------------+
         |       | Suc   | Fail                                      |
         |       |       |                                           |
         |       |       +-------------------------------------------+
         |       |                                                   |
         |       |             Suc                                   |
         |      current(D->S) ---------- count(D->S,r) --------------+
         |       | Fail                                              |
         |       |                                                   |
         |      create(D->S) ----------- count(D->S,r) --------------+
         |                                                           |
         |             Suc                                           |
        current(S->D) ------------------ count(S->D,f) --------------+
         | Fail                                                      |
         |             Suc                                           |
        current(D->S) ------------------ count(D->S,r) --------------+
         | Fail                                                      |
         |                                                           |
        create(S->D) ------------------- count(S->D,f) --------------+
                                                                     |
                                                                     *
```

When writing rule sets one must remember that the meter will normally try to match each packet in both directions. 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. Another approach would be to write a 'rule set consistency checker' program, which could detect problems in hand-written rule sets.

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

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. One member of this array is the CURRENT RULE SET, in that it is the one which is currently being used by the meter to classify incoming packets.

Rule set 1 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.

If the meter 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 structured as follows:

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

Opcodes contain two flags: 'goto' and 'test.' The PME maintains a Boolean indicator called the 'test indicator,' which is initially set (on). Execution begins with rule 1, the first in the rule set. It proceeds as follows:

```
If the test indicator is on:
   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 failure.

If the test indicator is off, or the test (above) succeeded:
   Set the test indicator to this rule'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, Fail, 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. The second, 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 and mask information from the pattern stack in the order it was enqueued.

The opcodes are:

	opcode	goto	test
1	Ignore	0	-
2	Fail	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

The actions they perform are:

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

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

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

CountPkt: As for Count, except that the masked value from the packet 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 value specified in 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 name, 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 name, mask, together with the masked value from the packet, 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 using a specified mask. The test in PushPktTo rules will almost never be executed.

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.

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 name 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. Since their values are only defined when matching is complete (and the flow key is built) their values may not be tested in rules.

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 seen for a flow which is not in the current flow set a flow record is set up 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 a ’inactive’ record - there is no particular significance in the ordering of records within the table.

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 a LastCollectTime variable which remembers the time the last collection was made. 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. 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. This would be suitable for use when one was interested in measuring flow lifetimes.

Another recovery strategy is to leave idle flows alone as long as possible, which would be suitable 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 out of '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, network Operations personnel should be able to specify the minimum number of meters - or perhaps a specific list of meters - which should collect a flow's data before its memory can be recovered. This issue will be further developed in the future.

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 memory usage rises above the high-water mark the meter should switch to a STANDBY RULE SET so as to increase the granularity of flow collection and 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. The meter should 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 a disk file called 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 id and 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. Each usage record contains the entity identifier of the meter (a network address), 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:

```
   +--------------------------------------------------------------------+
   |    RECORD IDENTIFIERS:                                             |
   |      Meter Id (& digital signature if required)                    |
   |      Timestamp                                                     |
   |      Collection Rules ID                                           |
   +--------------------------------------------------------------------+
   |    FLOW IDENTIFIERS:            |    COUNTERS                      |
   |      Address List               |       Packet Count               |
   |      Subscriber ID (Optional)   |       Byte Count                 |
   |      Attributes (Optional)      |    Flow Start/Stop Time          |
   +--------------------------------------------------------------------+
```

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 flows). One possible method of achieving this transfer is to use SNMP; the 'Traffic Flow Measurement: Meter MIB' document [4] 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 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; the others 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.

- SWITCH TO SPECIFIED RULE SET: Once the rule sets have been downloaded, the manager must instruct the meter which rule set it is to actually run (i.e. which is to be the current rule set), and which is to be the standby rule set.

- SET HIGH WATER MARK: A percentage value interpreted by the meter which tells the meter 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).

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 at least one meter reader, and show no activity since the last report,

  - Oldest flows, or

  - Flows with the smallest number of unreported 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.

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 reporting interval must be decreased or a coarser granularity of aggregation must be used so that more data fits into less space.

Increasing the reporting 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 report to the correct set of collection stations, and take steps to prevent unauthorised access to usage information. The collection stations so identified should be prepared to poll if necessary and accept data from the appropriate meters. Alternate collection stations may be identified in case both the primary manager and the primary collection station 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 must 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 flow-id granularities for each interface, 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 buffer space. Since, to prevent counting any packet twice, packets can only be counted in a single flow at any given time, 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 crashes, etc.) the meter could send a trap to the manager. The manager could then request one or more meter readers to pick up the usage record 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,

  - The reporting granularity is too fine, or

  - The throughput/bandwidth of circuits carrying the usage data is too low.

  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 APPENDICES

7.1 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:

```
        International                    |
        Backbones/National        ---------------
                                 /              \
        Regional/MidLevel     ----------   ----------
                             /   \     \  /   /     \
        Stub/Enterprise     ---   ---   ---   ----   ----
                            |||   |||   |||   ||||   ||||
        End-Systems/Hosts   xxx   xxx   xxx   xxxx   xxxx
```

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 "accountable (billable) party" labels so that meters can implement proprietary or non-standard schemes for the attribution of network traffic to responsible parties.

7.2 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 too 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.

7.3 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.

0	Null		
1	Flow Subscript	Integer	Flow table info
2	Flow Status	Integer	
4	Source Interface	Integer	Source Address
5	Source Adjacent Type	Integer	
6	Source Adjacent Address	String	
7	Source Adjacent Mask	String	
8	Source Peer Type	Integer	
9	Source Peer Address	String	
10	Source Peer Mask	String	
11	Source Trans Type	Integer	
12	Source Trans Address	String	
13	Source Trans Mask	String	
14	Destination Interface	Integer	Destination Address
15	Destination Adjacent Type	Integer	
16	Destination Adjacent Address	String	
17	Destination AdjacentMask	String	
18	Destination PeerType	Integer	
19	Destination PeerAddress	String	
20	Destination PeerMask	String	
21	Destination TransType	Integer	
22	Destination TransAddress	String	
23	Destination TransMask	String	
24	Packet Scale Factor	Integer	'Other' attributes
25	Byte Scale Factor	Integer	
26	Rule Set Number	Integer	
27	Forward Bytes	Counter	Source-to-Dest counters
28	Forward Packets	Counter	

29	Reverse Bytes	Counter	Dest-to-Source counters
30	Reverse Packets	Counter	
31	First Time	TimeTicks	Activity times
32	Last Active Time	TimeTicks	
33	Source Subscriber ID	String	Session attributes
34	Destination Subscriber ID	String	
35	Session ID	String	
36	Source Class	Integer	'Computed' attributes
37	Destination Class	Integer	
38	Flow Class	Integer	
39	Source Kind	Integer	
40	Destination Kind	Integer	
41	Flow Kind	Integer	
51	V1	Integer	Meter variables
52	V2	Integer	
53	V3	Integer	
54	V4	Integer	
55	V5	Integer	

7.4 Appendix D: List of Meter Control Variables

Current Rule Set Number	Integer
Standby Rule Set Number	Integer
High Water Mark	Percentage
Flood Mark	Percentage
Inactivity Timeout (seconds)	Integer
Last Collect Time	TimeTicks

8 Acknowledgments

This document was initially produced under the auspices of the IETF's Internet Accounting Working Group with assistance from SNMP, RMON and SAAG working groups. This version documents the implementation work done by the Internet Accounting Working Group, and is intended to provide a starting point for the Realtime Traffic Flow Measurement Working Group. Particular thanks are due to Stephen Stibler (IBM Research) for his patient and careful comments during the preparation of this memo.

9 References

[1] Mills, C., Hirsch, G. and G. Ruth, "Internet Accounting Background", RFC 1272, Bolt Beranek and Newman Inc., Meridian Technology Corporation, November 1991.

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

[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.

[4] Brownlee, N., "Traffic Flow Measurement: Meter MIB", RFC 2064, The University of Auckland, January 1997.

10 Security Considerations

Security issues are not discussed in detail in this document. The meter's management and collection protocols are responsible for providing sufficient data integrity and confidentiality.

11 Authors' Addresses

Nevil Brownlee
Information Technology Systems & Services
The University of Auckland

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

Cyndi Mills
BBN Systems and Technologies

Phone: +1 617 873 4143
EMail: cmills@bbn.com

Greg Ruth
GTE Laboratories, Inc

Phone: +1 617 466 2448
EMail: gruth@gte.com