OPSTAT Working Group
INTERNET-DRAFT

M. H. Lambert
Pittsburgh Supercomputing Center
June 1995

A Model for Common Operational Statistics
<draft-ietf-opstat-oper-model-01.ps>

Status of this Memo

Abstract

This memo describes a model for operational statistics in the Internet. It gives recommendations for metrics, measurements and polling periods relevant to the analysis of network performance and utilization. It suggests presentation formats for these measurements. It defines a format for the exchange of operational statistical data to facilitate collaborative planning and problem solving.

Table of Contents

This page intentionally left blank.

This page intentionally left blank.

1. Introduction

Many network administrations commonly collect and archive network management metrics that indicate network utilization, growth and reliability. The primary goals of this activity are to facilitate near-term problem isolation and longer-term network planning within the organization. There is also the broader goal of cooperative problem isolation and network planning among network administrations. This broader goal is likely to become increasingly important as the Internet continues to grow, particularly as the number of Internet service providers expands and the quality of service between providers becomes more of a concern.

There exist a variety of network management tools for the collection and presentation of network management metrics. However, different kinds of measurement and presentation techniques make it difficult to compare data among networks. In addition, there is not general consensus on what metrics should be regularly collected or how they should be presented.

There needs to be an agreed-upon model, called the Operational Statistics model, which specifies

(1) A minimal set of common network management metrics to satisfy the goals stated above,

(2) Tools for collecting these metrics,

(3) A common interchange format to facilitate the usage of these data by common presentation tools and

(4) Common presentation formats.

Under this model, collection tools will collect and store data to be retrieved later in a given format by presentation tools displaying the data in a predefined way. (See figure below.)

The Operational Statistics Model

(Collection of common metrics, by commonly available tools, stored in a common format, displayed in common formats by commonly available presentation tools.)

```
             !----------------------!
             !       Network        !
             !---+---------------+---!
                /                 \
               /                   \
              /                     \
     --------+------           ----+---------
     !     New     !          !    Old     !
     !  Collection !          ! Collection !
     !     Tool    !          !    Tool    !
     !---------+---!          !------+-----!
                \                    !
                 \           !-------+--------!
                  \          ! Post-Processor !
                   \         !--+-------------!
                    \           /
                     \         /
                      \       /
                  !--+-------+---!
                  !    Common    !
                  !  Statistics  !
                  !   Database   !
                  !-+--------+---!
                   /          \
                  /            \
                 /              \
                /           !-+-------------!
               /            ! Pre-Processor !
              /             !-------+-------!
    !-----------+--!                !
    !     New      !        !-------+-------!
    ! Presentation !        !     Old       !
    !     Tool     !        ! Presentation  !
    !---------+----!        !     Tool      !
             \              !--+------------!
              \                /
               \              /
             !-+---------------+-!
             ! Graphical Output  !
             ! (e.g., to paper   !
             ! or X Window)      !
             !-------------------!
```

This memo gives an overview of this model for common operational statistics. The model defines the collection, storage and presentation of network operational statistics and classifies the types of information that should be available at each network operation center (NOC) conforming to this model.

The model defines a minimal set of metrics and discusses how these metrics should be gathered and stored. It gives recommendations for the content and layout of statistical reports which make possible the easy comparison of network statistics among NOCs.

The primary purpose of this model is to define mechanisms by which NOCs could share most effectively their operational statistics. One intent of this model is to specify a baseline capability that NOCs conforming to the model may support with minimal development effort and minimal ongoing effort.

2. The Model

The model defines three areas of interest upon which all underlying concepts are based:

(1) The definition of a minimal set of metrics to be gathered,

(2) The definition of a format for storing collected statistical data and

(3) The definition of methods and formats for generating reports.

The model indicates that old tools currently in use could be retrofitted into the new paradigm. This could be done, for example, by providing conversion filters between old and new tools. In this sense this model intends to advocate the development of freely redistributable software for use by participating NOCs.

One basic idea of the model is that statistical data stored at one place could be retrieved and displayed at some other place. This is the primary motivation for the definition of the interchange format.

2.1. Metrics and Polling Periods

The intent here is to define a minimal set of metrics that could be gathered easily using standard SNMP-based network management tools. Thus, these metrics should be available as variables in the Internet Standard MIB (by this is meant MIB-II, as described in RFC 1213).

If the Internet Standard MIB were changed, this minimal set of metrics should be reconsidered, as there are many metrics regarded as important, but not currently defined in the standard MIB. Some metrics which are highly desirable to collect are probably not retrievable using SNMP. Therefore, tools and methods for gathering such metrics should be defined explicitly if such metrics are to be considered. This is, however, outside of the scope of this memo.

2.2. Format for the Interchange of Collected Data

A format for the interchange of data is defined. The intent is to minimize redundant information by using a single header structure wherein all information relevant to a certain set of statistical data is stored. This header section will give information about when and where the corresponding statistical data were collected.

2.3. Reports

Some basic classes of reports are suggested, addressing different views of network behavior.

(1) Reports of total octets and packets over some time period are regarded as essential to give an overall view of the traffic flow in a network.

(2) Differentiation between applications and protocols is regarded as needed to help identify which type of traffic is dominant.

(3) Reports on resource utilization are recommended.

The time period which a report spans may vary according to its intent. In engineering and operations, daily or weekly reports may be sufficient, whereas for capacity planning there may be a need for longer-term reports.

2.4. Security Issues

There are legal, ethical and political concerns about data sharing. People, in particular Network Service Providers, are concerned about showing data that may make one of their networks look bad.

For this reason there is a need to insure integrity, conformity and confidentiality of the shared data. To be useful, the same data should be collected from all involved sites and it should be collected at the same interval.

3. Categorization of Metrics

3.1. Overview

This section gives a classification of metrics with regard to scope and ease of retrieval. A recommendation of a minimal set of metrics is given. This section also gives some hints on metrics to be considered for future inclusion when available in the network management environment. Finally some thoughts on storage requirements are presented.

3.2. Categorization of Metrics Based on Measurement Areas

The metrics used in evaluating network traffic could be classified into (at least) four major categories:

- Utilization metrics
- Performance metrics
- Availability metrics
- Stability metrics

3.2.1. Utilization Metrics

This category describes different aspects of the total traffic being forwarded through the network. Possible metrics include:

- Total input and output packets and octets
- Various peak metrics
- Per protocol and per application metrics

3.2.2. Performance Metrics

These metrics relate to quality of service issues such as delays and congestion situations. Possible metrics include:

- RTT metrics on different protocol layers
- Number of collisions on a bus network

- Number of ICMP Source Quench messages
- Number of packets dropped

3.2.3. Availability Metrics

These metrics could be viewed as gauging long-term accessibility on different protocol layers. Possible metrics include:

- Line availability as a percentage of uptime
- Route availability
- Application availability

3.2.4. Stability Metrics

These metrics describe short-term fluctuations in the network which degrade the service level. Changes in traffic patterns also could be recognized using these metrics. Possible metrics include:

- Number of fast line status transitions
- Number of fast route changes (also known as route flapping)
- Number of routes per interface in the tables
- Next hop count stability
- Short term ICMP behavior

3.3. Categorization Based on Availability of Metrics

To be able to retrieve metrics, the corresponding variables must be accessible at every network object which is part of the management domain for which statistics are being collected.

Some metrics are easily retrievable because they are defined as variables in the Internet Standard MIB. Other metrics may be retrievable because they are part of some vendor's private enterprise MIB subtree. Finally, some metrics are considered irretrievable, either because they are not possible to include in the SNMP concept or because their measurement would require extensive polling (loading the network with management traffic).

The metrics categorized below could each be judged as important in evaluating network behavior. This list may serve as a basis for revisiting the decisions on which metrics are to be regarded as reasonable and desirable to collect. If the availability of the metrics listed below were to change, these decisions might change.

3.3.1. Per Interface Variables Already in Internet Standard MIB

These variables are easy to retrieve:

`interfaces.ifTable.ifEntry.ifInUcastPkts`	unicast packets in
`interfaces.ifTable.ifEntry.ifOutUcastPkts`	unicast packets out
`interfaces.ifTable.ifEntry.ifInNUcastPkts`	non-unicast packets in
`interfaces.ifTable.ifEntry.ifOutNUcastPkts`	non-unicast packets out
`interfaces.ifTable.ifEntry.ifInOctets`	octets in
`interfaces.ifTable.ifEntry.ifOutOctets`	octets out

`interfaces.ifTable.ifEntry.ifOperStatus` line status

3.3.2. Per Interface Variables in Internet Private Enterprise MIB

These variables are easy to retrieve, but only if included in the vendor's MIB.

discarded packets in
discarded packets out
congestion events in
congestion events out
aggregate errors
interface resets

3.3.3. Per Interface Variables Needing High Resolution Polling

These variables are desirable, but the process of polling them at the necessary rate could bias the data.

interface queue length
seconds missing stats
interface unavailable
route changes
interface next hop count

3.3.4. Per Interface Variables not in any Known MIB

These variables could be retrieved via SNMP if they were included in a MIB.

link layer packets in
link layer packets out
link layer octets in
link layer octets out
packet interarrival times
packet size distribution

3.3.5. Per Node Variables

These are some miscellaneous variables retrievable via SNMP.

per-protocol packets in
per-protocol packets out
per-protocol octets in
per-protocol octets out
packets discarded in
packets discarded out
packet size distribution

system uptime
poll delta time
reboot count

3.3.6. Metrics not Retrievable with SNMP

Although of interest, these variables cannot be retrieved with SNMP, but must be obtained by some other mechanism.

delays (RTTs) on different protocol layers
application layer availabilities
peak behavior metrics

3.4. Recommended Metrics

A large number of metrics could be considered for collection in the process of doing network statistics. To facilitate general consensus for this model, there is a need to define a minimal set of metrics that are both essential and retrievable in a majority of today's network objects. General retrievability is equated with presence in the Internet Standard MIB.

The following metrics from the Internet Standard MIB were chosen as being desirable and reasonable:

For each interface:

`interfaces.ifTable.ifEntry.ifInOctets`	octets in
`interfaces.ifTable.ifEntry.ifOutOctets`	octets out
`interfaces.ifTable.ifEntry.ifInUcastPkts`	unicast packets in
`interfaces.ifTable.ifEntry.ifOutUcastPkts`	unicast packets out
`interfaces.ifTable.ifEntry.ifInNUcastPkts`	non-unicast packets in
`interfaces.ifTable.ifEntry.ifOutNUcastPkts`	non-unicast packets out
`interfaces.ifTable.ifEntry.ifInDiscards`	in discards
`interfaces.ifTable.ifEntry.ifOutDiscards`	out discards
`interfaces.ifTable.ifEntry.ifOperStatus`	line status

For each node:

`ip.ipForwDatagrams`	IP forwards
`ip.ipInDiscards`	IP in discards
`system.sysUpTime`	system uptime

4. Polling Frequencies

The purpose of polling at specified intervals is to gather statistics to serve as a basis for trend and capacity planning. From the operational data it should be possible to derive engineering and management data. It should be noted that all polling and retention values given below are recommendations and are not mandatory.

4.1. Variables Needing High Resolution Polling

To be able to detect peak behavior, it is recommended that a period of one minute (60 seconds) at a maximum be used in gathering traffic data. The metrics to be collected at this frequency are:

For each interface

`interfaces.ifTable.ifEntry.ifInOctets`	octets in
`interfaces.ifTable.ifEntry.ifOutOctets`	octets out
`interfaces.ifTable.ifEntry.ifInUcastPkts`	unicast packets in
`interfaces.ifTable.ifEntry.ifOutUcastPkts`	unicast packets out

If it is not possible to gather data at this high polling frequency, it is recommended that an exact multiple of 60 seconds be used. The initial polling frequency value will be part of the stored statistical data as described in section 6.1.2 below.

4.2. Variables not Needing High Resolution Polling

The remainder of the recommended variables to be gathered, *i.e.*, for each interface

`interfaces.ifTable.ifEntry.ifInNUcastPkts`	non-unicast packets in
`interfaces.ifTable.ifEntry.ifOutNUcastPkts`	non-unicast packets out
`interfaces.ifTable.ifEntry.ifInDiscards`	in discards
`interfaces.ifTable.ifEntry.ifOutDiscards`	out discards
`interfaces.ifTable.ifEntry.ifOperStatus`	line status

and for each node

`ip.ipForwDatagrams`	IP forwards
`ip.ipInDiscards`	IP in discards
`system.sysUpTime`	system uptime

could be collected at a lower polling rate. No polling rate is specified, but it is recommended that the period chosen be an exact multiple of 60 seconds.

5. Pre-Processing of Raw Statistical Data

5.1. Optimizing and Concentrating Data to Resources

To avoid storing redundant data in what might be a shared file system, it is desirable to preprocess the raw data. For example, if a link is down there is no need to continuously store a counter which is not changing. The use of the variables `sysUpTime` and `ifOperStatus` makes it possible to avoid continuously storing data collected from links and nodes where no traffic has been transmitted for some period of time.

Another aspect of processing is to decouple the data from the raw interface being polled. The intent should be to convert such data into the resource of interest as, for example, the traffic on a given link. Changes of interface in a gateway for a given link should not be visible in the resulting data.

5.2. Aggregation of Data

At many sites, the volume of data generated by a polling period of one minute will make aggregation of the stored data desirable if not necessary.

Aggregation here refers to the replacement of data values on a number of time intervals by some function of the values over the union of the intervals. Either raw data or shorter-term aggregates may be aggregated. Note that aggregation reduces the amount of data, but also reduces the available information.

In this model, the function used for the aggregation is either the arithmetic mean or the maximum, depending on whether it is desired to track the average or peak value of a variable.

Details of the layout of the aggregated entries in the data file are given in section 6.1.3.

Suggestions for aggregation periods:

Over a

24 hour period	aggregate to 15 minutes,
1 month period	aggregate to 1 hour,
1 year period	aggregate to 1 day.

6. Storage of Statistical Data for Interchange

This section describes a format for the storage of statistical data. The goal is to facilitate a common set of tools for the gathering, storage and analysis of statistical data. The format is defined with the intent of minimizing redundant information and thus minimizing storage requirements. If a client-server based model for retrieving remote statistical data were later developed, the specified storage format could be used as the transmission protocol.

This model is intended to define an interchange file format, which would not necessarily be used for actual data storage. That means its goal is to provide complete, self-contained, portable files, rather than to describe a full database for storing them.

6.1. The Storage Format

All white space (including tabs, line feeds and carriage returns) within a file is ignored. In addition, all text from a # symbol to the following end of line (inclusive) is also ignored.

```
stat-data ::= <stat-section> [ <FS> <stat-section> ] [ <FS> ]
stat-section ::= <device-section> | <label-section> | <data-section>
```

A data file must contain at least one device section and at least one label section. At least one data section must be associated with each label section. A device section must precede any data section which uses tags defined within it.

A data section may appear in the file (in which case it is called an internal data section and is preceded by a label section) or in another file (in which case it is called an external data section and is specified in an external label section). Such an external file may contain one and only one data section.

A label section indicates the start and finish times for its associated data-section(s) and a list of the names of the tags they contain. Within a data file there is an ordering of label sections. This depends only upon their relative position in the file. All internal data sections associated with the first label record must precede those associated with the second label record, and so on.

Here are some examples of valid data files:

```
<label-s> <device-s> <data-s> <data-s>
<label-s> <device-s> <data-s> <device-s> <data-s> <data-s>
```

Both these files start with a label section giving the times and tag-name lists for the device- and data-sections which follow.

```
<dev-s> <label-s> <label-s> <label-s>
```

This file begins with a device section (which specifies tags used in its data sections) then has three ‘external’ label sections, each of which points to a separate data section. The data sections need not use all the tags defined in the device section; this is indicated by the tag-name lists in their label sections.

```
<default-device> <dev-1> <label-1> <dev-2> <label-2> ...
```

In this example, default-device is a full device section, including a complete tag table, with initial polling and aggregation periods specified for each variable in each variable field. There is no label or data for default-device—it is there purely to provide default tag-list information. Dev-1, dev-2, ... are device sections for a series of different devices. They each have their description fields (network-name, device-name, *etc.*), but no tag-table. Instead they rely on using the tag-table from default-device. A default-device record, if present, must be the first item in the data file. Label-1, label-2, *etc.* are label sections which point to files containing data sections for each device.

6.1.1. The Label Section

```
label-section ::= BEGIN_LABEL <data-location> <FS> <tag-name-list> <FS>
                  <start-time> <FS> <stop-time> <FS> END_LABEL
data-location ::= <FS> <data-file-name> | <empty>
tag-name-list ::= <LEFT> <tag> [ <FS> <tag> ] <RIGHT>
```

The label section gives the start and stop times for its corresponding data section(s) and a list of the tags it uses. If a data location is given, it specifies the name of a file containing its data section; otherwise the data section follows in this file.

```
start-time     ::= <time-string>
stop-time      ::= <time-string>
data-file-name ::= <ASCII-string>
```

```
time-string    ::= <year><month><day><hour><minute><second>
year           ::= <digit><digit><digit><digit>
month          ::= 01..12
day            ::= 01..31
hour           ::= 00..23
minute         ::= 00..59
second         ::= <float>
```

The start-time and stop-time are specified in UTC.

To allow arbitrary timing granularity, seconds can be a floating-point value. However, it is anticipated that in normal use it will be an integer. A maximum of 60.0 is specified for seconds so as to allow for leap seconds, as is done (for example) by the Network Time Protocol.

6.1.2. The Device Section

```
device-section  ::= BEGIN_DEVICE <FS> <device-field> <FS> END_DEVICE
device-field    ::= <network-name><FS><router-name><FS><link-name<FS>
                    <bw-value><FS><proto-type><FS><proto-addr><FS>
                    <time-zone> <optional-tag-table>
optional-tag-table  ::= <FS> <tag-table> | <empty>

network-name    ::= <ASCII-string>
device-name     ::= <ASCII-string>
link-name       ::= <ASCII-string>
bw-value        ::= <float>
proto-type      ::= IP | DECNET | X.25 | CLNS | IPX | AppleTalk
proto-addr      ::= <ASCII-string>
time-zone       ::= [+|-] [00..13] [00..59]

tag-table       ::= <LEFT> <tag-desc> [ <FS> <tag-desc> ] <RIGHT>
tag-desc        ::= <tag> <FS> <tag-class> <FS> <variable-field-list>

tag             ::= <ASCII-string>
tag-class       ::= total | peak

variable-field-list    ::= <LEFT> <variable-field>
                           [ <FS> <variable-field> ] <RIGHT>
variable-field         ::= <variable-name><FS><initial-polling-period>
                           <FS> <aggregation-period>

variable-name          ::= <ASCII-string>
initial-polling-period ::= <integer>
aggregation-period     ::= <integer>
```

The network-name is a human readable string indicating to which network the logged data belong.

The device-name is given as an ASCII string, allowing for styles other than IP domain names (which are names of interfaces, not devices).

The link-name is a human readable string indicating the connectivity of the link where from the logged data is gathered.

The units for bandwidth (bw-value) are bits per second, and are given as a floating-point number, *e.g.*, 1536000 or 1.536e6. A zero value indicates that the actual bandwidth is unknown; one instance of this would be a Frame Relay link with Committed Information Rate different from Burst Rate.

The proto-type field describes to which network architecture the interface being logged is connected. Valid types are IP, DECNET, X.25, CLNS, IPX and AppleTalk. The mechanism for extending this list of protocols is not discussed in this memo.

The network address (proto-addr) is the unique numeric address of the interface being logged. The actual form of this address is dependent on the protocol type as indicated in the proto-type field. For IP connected interfaces the dotted-quad notation should be used.

The time-zone indicates the time difference that should be added to the time-stamp in the data-section to give the local time for the logged interface. Note that the range for time-zone is sufficient to allow for all possibilities, not just those which fall on 30-minute multiples. If a time-zone changes during a data file—*e.g.*, because daylight savings time has ended—this should be recorded by ending the current data section, writing a device section with the new time zone and starting a new data section.

The tag-table lists all variables being polled. To minimize ambiguity, variable names should be fully qualified Internet MIB names. The table may contain multiple tags. Each tag must be associated with only one polling and one aggregation period. If variables are being polled or aggregated at different periods, a separate tag in the table must be used for each period.

As variables may be polled with different polling periods within the same set of logged data, there is a need to explicitly associate a polling period with each variable. After processing, the actual period covered may have changed compared to the initial polling period and this should be noted in the aggregation period field. The initial polling and aggregation periods are given in seconds.

Original data values, and data values which have been aggregated by adding them together, will have a tag-class of '`total`'. Data values which have been aggregated by finding the maximum over an aggregation time interval will have a tag-class of '`peak`'.

The tag-table and variable-field-lists are enclosed in brackets, making the extent of each obvious. Without the brackets a parser would have difficulty distinguishing between a variable name (continuing the variable-field list for this tag) or a tag (starting the next tag of the tag table). To make the distinction clearer to a human reader one should use different kinds of brackets for each, for example { } for the tag-table list and [] for the variable-field lists.

6.1.3. The Data Section

```
data-section   ::= BEGIN_DATA <FS> <data-field>
                         [ <FS> <data-field> ] <FS> END_DATA
data-field     ::= <time-string> <FS> <tag> <FS>
                         <poll-delta> <FS> <delta-val-list>

delta-val-list ::= LEFT <delta-val> [ <FS> <delta-val> ] RIGHT

poll-delta     ::= <integer>
delta-val      ::= <integer>
```

```
FS              ::= , | ; | :
LEFT            ::= ( | [ | {
RIGHT           ::= ) | ] | }
```

A data field contains values for each variable in the specified tag. A new data field should be written for each separate poll; there should be a one-to-one mapping betwen variables and values. Each data-field begins with the timestamp for this poll followed by the tag defining the polled variables followed by a polling delta value giving the period of time in seconds since the previous poll. The variable values are stored as delta values for counters and as absolute values for non-counter values such as OperStatus. The timestamp is in UTC and the time-zone field in the device section is used to compute the local time for the device being logged. The first entries in a data section are for all tags corresponding to original data and are written at start-time as defined in the corresponding label section.

Comma, semicolon or colon may be used as a field separator. Normally one would use commas within a line, semicolon at the end of a line and a colon after keywords such as BEGIN_LABEL.

Parentheses (), brackets [] or braces {} may be used as LEFT and RIGHT brackets around tag-name, tag-table and delta-val lists. These should be used in corresponding pairs, although combinations such as (], [} etc. are syntactically valid.

6.2. Storage Requirement Estimations

The header sections are not counted in this example. Assuming that the maximum polling intensity is used for all 12 recommended variables, that the size in ASCII of each variable is eight bytes and that there are no timestamps which are fractional seconds, the following calculations will give an estimate of storage requirements for one year of storing and aggregating statistical data.

Assuming that data is saved according to the scheme

Time Period	Aggregated?	Save for
1 minute	No	1 day
15 minutes	Yes	1 week
1 hour	Yes	1 month
1 day	Yes	1 year

this will give:

Size, in byte, of one entry for each aggregation period:

	Aggregation periods			
	1 min	15 min	1 hour	1 day
Timestamp	14	14	14	14
Tag	5	5	5	5
Poll-Delta	2	3	4	5
Total values	96	96	96	96
Peak values	0	96	192	288

Field separators	14	28	42	56
Total entry size	131	242	353	464

For each day, 60×24 = 1440 entries with a total size of 1440×131 bytes = 189 kB.

For each week, $4 \times 24 \times 7$ = 672 entries are stored with a total size of 672×242 bytes = 163 kB.

For each month, 24×30 = 720 entries are stored with a total size of 720×353 bytes = 254 kB.

For each year, 365 entries are stored with a total size of 365×464 bytes = 169 kB.

Grand total estimated peak storage for one year = 775 kB.

7. Report Formats

This section suggests some report formats and defines the metrics to be used in such reports.

7.1. Report Types and Contents

There are longer-term needs for monthly and yearly reports showing long-term tendencies in the network. There are short-term weekly reports giving information about medium-term changes in network behavior which could serve as input to the medium-term engineering approach. Finally, there are daily reports giving the instantaneous overviews needed in the daily operations of a network.

These reports should give information on:

Offered Load	Total traffic at external interfaces
Offered Load	Segmented by ‘‘Customer’’
Offered Load	Segmented protocol/application.
Resource Utilization	Link/Router

7.2. Content of the Reports

Here we discuss what information should be included in each type of report.

7.2.1. Offered Load by Link

Metric categories:
- input octets per external interface
- output octets per external interface
- input packets per external interface
- output packets per external interface

The intent is to visualize the overall trend of network traffic on each connected external interface. This could be done as a bar-chart giving the totals for each of the four metric categories. Based on the time period selected this could be done on a hourly, daily, monthly or yearly basis.

7.2.2. Offered Load by Customer

Metric categories:	input octets per customer output octets per customer input packets per customer output packets per customer

The recommendation here is to sort the offered load (in decreasing order) by customer. Plot the function F, where $F(n)$ is percentage of total traffic offered to the top n customers or the function f where $f(n)$ is the percentage of traffic offered by the nth ranked customers.

The definition of what is meant by a ‘‘customer’’ has to be done locally at the site where the statistics are being gathered—it could be anything from an entire national network provider to a single host.

A cumulative plot could be useful as an overview of how traffic is distributed among users since it enables one to quickly pick off what fraction of the traffic comes from what number of ‘‘users’’.

A method of displaying both average and peak behavior in one bar chart is to compute both the average value over some period and the peak value during the same period. The average and peak values are then displayed in the same bar.

7.2.3. Resource Utilization Reporting

7.2.3.1. Utilization as Maximum Peak Behavior

Link utilization is used to capture information on network loading. The polling interval must be small enough to be significant with respect to variations in human activity, since this is the activity that drives variations in network loading. On the other hand, there is no need to make it smaller than an interval over which excessive delay would notably impact productivity. For this reason, 30 minutes is a good estimate of the time at which people remain in one activity and over which prolonged high delay will affect their productivity. To track 30 minute variations, there is a need to sample twice as frequently, *i.e.*, every 15 minutes. Use of a polling period of five or ten minutes should be sufficient to capture variations in utilization.

A possible format for reporting utilization as peak behavior is to use a method of combining averages and peak measurements onto the same diagram. Compare, for example, peak meters on audio equipment. If a diagram contains the daily totals for some period, then the peaks would be the busiest hour during each day. If the diagram were totals on an hourly basis, then the peak would be the maximum ten-minute period in each hour.

By combining the average and the maximum values for a certain time period, it should be possible to detect line utilization and bottlenecks due to temporary high loads.

7.2.3.2. Utilization Visualized as a Frequency Distribution of Peaks

Another way of visualizing line utilization is to put the ten-minute samples in a histogram showing the relative frequency among the samples versus the load.

8. Considerations for Future Development

This memo is the first effort at formalizing a common basis for operational statistics. One major guideline in this work has been to keep the model simple to facilitate the easy integration of this model by vendors and NOCs into their operational tools.

There are, however, some ideas that could progress further to expand the scope and usability of the model.

8.1. A Client/Server Based Statistical Exchange System

A possible path for development could be the definition of a client/server based architecture for providing remote access to operational statistics. Such an architecture envisions that each NOC install a server which provides locally collected information in a variety of forms for clients.

Using a query language, the client should be able to define the network object, the interface, the metrics and the time period to be provided. Using a TCP-based protocol, the server will transmit the requested data. Once these data are received by the client, they could be processed and presented by a variety of tools. One possibility is to have an X-Window based tool that displays defined diagrams from data, supporting the feeding of such diagrams into the X-Window tool directly from the statistical server. Another complementary method would be to generate PostScript output to print the diagrams. In all cases it should be possible to store the retrieved data locally for later processing.

The client/server approach is discussed further by Henry Clark (in an Internet Draft current at the time of writing of this memo).

8.2. Inclusion of Variables not in the Internet Standard MIB

As has been pointed out above in the categorization of metrics, there are metrics which certainly could have been recommended if they were available in the Internet Standard MIB. To facilitate the inclusion of such metrics in the set of recommended metrics, it will be necessary to specify a subtree in the Internet Standard MIB containing variables judged necessary in the scope of performing operational statistics.

8.3. Detailed Resource Utilization Statistics

One area of interest not covered in the above description of metrics and presentation formats is to present statistics on detailed views of traffic flows. Such views could include statistics on a per application basis and on a per protocol basis. Today such metrics are not part of the Internet Standard MIB. Tools like the NSF NNStat are being used to gather information of this kind. A possible way to achieve such data could be to define an NNStat MIB or to include such variables in the above suggested operational statistics MIB subtree. The Accounting Meter Services MIB, discussed by Nevil Brownlee (in an Internet Draft current at the time of writing of this memo), provides another source for traffic flow statistics.

Appendix A. Some Formulas for Statistical Aggregation

Let *poll* be a variable of class `total` which is being polled or aggregated. Then $poll(n)_j$ is the total of *poll* over the jth n-second period.

For example, $\text{ifInOctets}(900)_3$ is the total number of input octets between the 31st and 45th minutes (the third 900-second period), inclusive.

Let *peak* be a variable of class `total` which is being polled or aggregated. Then $peak(n, m)_j$ is the m-second peak value of *peak* over the jth n-second period.

For example, ifInOctets$(3600, 900)_3$ is the maximum number of input octets during a 15-minute period in the third hour.

Assume a polling granularity of one minute over a 24-hour period, giving a total of 1440 logged entries.

Without any aggregation we have

$$poll(60)_{1,}\ poll(60)_2, \ldots, poll(60)_{1440.}$$

If we aggregate these values at 15-minute intervals we have 1440/15 = 96 total values

$$poll(900)_{1,}\ poll(900)_2, \ldots, poll(900)_{96}$$

where

$$poll(900)_k = \sum_{j=15(k-1)+1}^{15k} poll(60)_j, \quad \text{for } k = 1, 2, \ldots, 96$$

and 96 one-minute peak values

$$peak(900, 60)_{1,}\ peak(900, 60)_2, \ldots, peak(900, 60)_{96}$$

where

$$peak(900, 60)_k = \max_{j=15(k-1)+1}^{15k} poll(60)_j, \quad \text{for } k = 1, 2, \ldots, 96.$$

The next aggregation step is from 15 minutes to 1 hour, giving 24 totals:

$$poll(3600)_k = \sum_{j=4(k-1)+1}^{4k} poll(900)_j, \quad \text{for } k = 1, 2, \ldots, 24,$$

twenty-four one-minute peaks calculated over each hour:

$$peak(3600, 60)_k = \max_{j=4(k-1)+1}^{4k} peak(900, 60)_j, \quad \text{for } k = 1, 2, \ldots, 24$$

and finally 24 15-minute peaks calculated over each hour:

$$peak(3600, 900)_k = \max_{j=4(k-1)+1}^{4k} poll(900)_j, \quad \text{for } k = 1, 2, \ldots, 24.$$

The next aggregation step is from one hour to 24 hours. For each day, k, with 1440 entries as above, this will give one total:

$$poll(86400)_k = \sum_{j=24(k-1)+1}^{24k} poll(3600)_j, \quad \text{for } k = 1, 2, ...$$

and three peaks:

$$peak(86400, 60)_k = \max_{j=24(k-1)+1}^{24k} peak(3600, 60)_j, \quad \text{for } k = 1, 2, ...,$$

the busiest one-minute period over 24 hours,

$$peak(86400, 900)_k = \max_{j=24(k-1)+1}^{24k} peak(3600, 900)_j, \quad \text{for } k = 1, 2, ...,$$

the busiest 15-minute period over 24 hours and

$$peak(86400, 3600)_k = \max_{j=4(k-1)+1}^{4k} poll(3600)_j, \quad \text{for } k = 1, 2, ...,$$

the busiest hour over 24 hours.

There will probably be differences among the three peak values in the final 24 hour aggregation. A smaller peak period will give higher values than a longer one, if the values are scaled by the length of the period:

$$poll(86400)/3600 \le peak(86400, 3600) \le 4\,peak(86400, 900) \le 60\,peak(86400, 60).$$

Appendix B. Example

We shall now present a somewhat schematic example of a data file in the interchange format. Note that in what follows, names such as `ifInOctets` should really be the fully qualified MIB names. We abbreviate for brevity, not obfuscation.

Let us assume that a portion of the device section for the original collection of raw data is

```
BEGIN_DEVICE:
   ...
{
   UNI-1,total: [ifInOctets,     60, 60,ifOutOctets,     60, 60];
   BRD-1,total: [ifInNUcastPkts,300,300,ifOutNUcastPkts,300,300]
}
END_DEVICE;
```

Here there are two tags, `UNI-1` and `BRD-1`, each of class `total`. `UNI-1` has associated with it two variables, `ifInOctets` and `ifOutOctets`, each of which is polled every 60 seconds and aggregated every 60 seconds (*i.e.*, they are not aggregated—this is a convention to indicate that the raw data is being written to the file). Associated with the tag `BRD-1` are the variables `ifInNUcastPkts` and `ifOutNUcastPkts`, each recorded every 300 seconds.

This device section has an associated data section

```
BEGIN_DATA:
   19950730000000,UNI-1,60:(val1-1, val2-1);
   19950730000000,BRD-1,300:(val1-2, val2-2);
   19950730000100,UNI-1,60:(val1-3, val2-3);
   19950730000200,UNI-1,60:(val1-4, val2-4);
   19950730000300,UNI-1,60:(val1-5, val2-5);
   19950730000400,UNI-1,60:(val1-6, val2-6);
   19950730000500,UNI-1,60:(val1-7, val2-7);
   19950730000500,BRD-1,300:(val1-8, val2-8);
   19950730000600,UNI-1,60:(val1-9, val2-9);
   19950730000700,UNI-1,60:(val1-10, val2-10);
   ...
END_DATA;
```

Here we see that the first record was written at midnight UTC on 30 July 1995. It is for tag `UNI-1` and indicates a Δt of 60 seconds (this is artificial for any data points written at the starting time of the file). The list of data values contains deltas for counter variables and actual values for non-counter variables. The values *val*1 correspond to the first variable associated with the tag, *val*2 with the second variable, *etc*. Since both variables associated with `UNI-1` are counters, the initial deltas are zero (*val*1 → `ifInOctets` and *val*2 → `ifOutOctets`).

Aggregation of the raw data to 15 minutes gives a device section

```
BEGIN_DEVICE:
    ...
{
    UNI-1,total: [ifInOctets,     60,900,ifOutOctets,     60,900];
    BRD-1,total: [ifInNUcastPkts,300,900,ifOutNUcastPkts,300,900];
    UNI-2,peak:  [ifInOctets,     60,900,ifOutOctets,     60,900];
    BRD-2,peak:  [ifInNUcastPkts,300,900,ifOutNUcastPkts,300,900]
}
END_DEVICE;
```

where UNI-1 is the 15-minute total
BRD-1 is the 15-minute total
UNI-2 is the 1-minute peak over 15 minutes (peak = *peak*(1))
BRD-2 is the 5-minute peak over 15 minutes (peak = *peak*(1)).

The `UNI-1` and `BRD-1` tags are of class `total` and reflect the original polling periods (60 or 300 seconds) and the new aggregation period (900 seconds). `UNI-2` and `BRD-2` are the corresponding tags of class `peak`.

Accompanying this new device section is a data section

```
BEGIN_DATA:
   19950730001500,UNI-1,900:(total(val1),total(val2));
   19950730001500,BRD-1,900:(total(val1),total(val2));
   19950730001500,UNI-2,900:(peak(1)(val1),peak(1)(val2));
   19950730001500,BRD-2,900:(peak(1)(val1),peak(1)(val2));
   19950730003000,UNI-1,900:(total(val1),total(val2));
   19950730003000,BRD-1,900:(total(val1),total(val2));
   19950730003000,UNI-2,900:(peak(1)(val1),peak(1)(val2));
   19950730003000,BRD-2,900:(peak(1)(val1),peak(1)(val2));
   ...
END_DATA;
```

Four records were written at 00:15:00 UTC, *i.e.*, at the end of the first 15-minute aggregation interval. The first record, for tag `UNI-1`, has 15-minute total values for `ifInOctets` and `ifOutOctets`. The fourth record, for tag `BRD-2`, contains five-minute peaks over the 15-minute interval for `ifInNUcastPkts` and `ifOutNUcastPkts`. Additional records are written every 15 minutes.

The next step is to aggregate these 15-minute aggregates to one hour. This yields a device section

```
BEGIN_DEVICE:
```

```
    ...
{
   UNI-1,total: [ifInOctets,     60,3600,ifOutOctets,     60,3600];
   BRD-1,total: [ifInNUcastPkts,300,3600,ifOutNUcastPkts,300,3600];
   UNI-2,peak:  [ifInOctets,     60,3600,ifOutOctets,     60,3600];
   BRD-2,peak:  [ifInNUcastPkts,300,3600,ifOutNUcastPkts,300,3600];
   UNI-3,peak:  [ifInOctets,    900,3600,ifOutOctets,    900,3600];
   BRD-3,peak:  [ifInNUcastPkts,900,3600,ifOutNUcastPkts,900,3600]
}
END_DEVICE;
```

where UNI-1 is the one hour total
BRD-1 is the one hour total
UNI-2 is the 1-minute peak over 1 hour (peak of peak = *peak*(2))
BRD-2 is the 5-minute peak over 1 hour (peak of peak = *peak*(2))
UNI-3 is the 15-minute peak over 1 hour (peak = *peak*(1))
BRD-3 is the 15-minute peak over 1 hour (peak = *peak*(1)).

Here there are one-hour totals, one- (or five-) minute peaks over the hour and 15-minute peaks over the hour. Note that in `UNI-2` the initial polling period is 60 seconds whereas in `UNI-3` the initial polling period is 900 seconds—the variable in `UNI-3` are derived from quantities which have already been aggregated.

The resulting data section is

```
BEGIN_DATA:
   19950730010000,UNI-1,3600:(total(val1),total(val2));
   19950730010000,BRD-1,3600:(total(val1),total(val2));
   19950730010000,UNI-2,3600:(peak(2)(val1), peak(2)(val2));
   19950730010000,BRD-2,3600:(peak(2)(val1), peak(2)(val2));
   19950730010000,UNI-3,3600:(peak(1)(val1), peak(1)(val2));
   19950730010000,BRD-3,3600:(peak(1)(val1), peak(1)(val2));
   19950730020000,UNI-1,3600:(total(val1),total(val2));
   19950730020000,BRD-1,3600:(total(val1),total(val2));
   19950730020000,UNI-2,3600:(peak(2)(val1), peak(2)(val2));
   19950730020000,BRD-2,3600:(peak(2)(val1), peak(2)(val2));
   19950730020000,UNI-3,3600:(peak(1)(val1), peak(1)(val2));
   19950730020000,BRD-3,3600:(peak(1)(val1), peak(1)(val2));
   ...
END_DATA;
```

Six additional records are written each hour.

The final aggregation is from one hour to 24 hours, giving the device section

```
BEGIN_DEVICE:
   ...
{
```

```
    UNI-1,total: [ifInOctet,       60,86400,ifOutOctet,       60,86400];
    BRD-1,total: [ifInNUcastPkts, 300,86400,ifOutNUcastPkts, 300,86400];
    UNI-2,peak:  [ifInOctet,       60,86400,ifOutOctet,       60,86400];
    BRD-2,peak:  [ifInNUcastPkts, 300,86400,ifOutNUcastPkts, 300,86400];
    UNI-3,peak:  [ifInOctet,      900,86400,ifOutOctet,      900,86400];
    BRD-3,peak:  [ifInNUcastPkts, 900,86400,ifOutNUcastPkts, 900,86400];
    UNI-4,peak:  [ifInOctet,     3600,86400,ifOutOctet,     3600,86400];
    BRD-4,peak:  [ifInNUcastPkts,3600,86400,ifOutNUcastPkts,3600,86400]
}
END_DEVICE;
```

where UNI-1 is the 24 hour total
BRD-1 is the 24 hour total
UNI-2 is the 1 minute peak over 24 hour (peak of peak of peak = *peak*(3))
UNI-3 is the 15 minute peak over 24 hour (peak of peak = *peak*(2))
UNI-4 is the 1 hour peak over 24 hour (peak = *peak*(1))
BRD-2 is the 5 minute peak over 24 hour (peak of peak of peak = *peak*(3))
BRD-3 is the 15 minute peak over 24 hour (peak of peak = *peak*(2))
BRD-4 is the 1 hour peak over 24 hour (peak = *peak*(1)).

The corresponding data section is:

```
BEGIN_DATA:
    19950731000000,UNI-1,86400:(total(val1),total(val2));
    19950731000000,BRD-1,86400:(total(val1),total(val2));
    19950731000000,UNI-2,86400:(peak(3)(val1),peak(3)(val2));
    19950731000000,BRD-2,86400:(peak(3)(val1),peak(3)(val2));
    19950731000000,UNI-3,86400:(peak(2)(val1),peak(2)(val2));
    19950731000000,BRD-3,86400:(peak(2)(val1),peak(2)(val2));
    19950731000000,UNI-4,86400:(peak(1)(val1),peak(1)(val2));
    19950731000000,BRD-4,86400:(peak(1)(val1),peak(1)(val2));
    19950801000000,UNI-1,86400:(total(val1),total(val2));
    19950801000000,BRD-1,86400:(total(val1),total(val2));
    19950801000000,UNI-2,86400:(peak(3)(val1),peak(3)(val2));
    19950801000000,BRD-2,86400:(peak(3)(val1),peak(3)(val2));
    19950801000000,UNI-3,86400:(peak(2)(val1),peak(2)(val2));
    19950801000000,UNI-3,86400:(peak(2)(val1),peak(2)(val2));
    19950801000000,UNI-4,86400:(peak(1)(val1),peak(1)(val2));
    19950801000000,BRD-4,86400:(peak(1)(val1),peak(1)(val2));
    ...
END_DATA;
```

Security Considerations

Security issues are discussed in Section 2.4.

Acknowledgements

Bernhard Stockman is the author of RFC 1404, upon which this memo is based. Nevil Brownlee produced the revised BNF description of the model. There were a number of useful and constructive comments submitted on the Operation Statistics Working Group mailing list.

Author's Address

Michael H. Lambert
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh, PA 15213
USA

Phone: +1 412 268-4960
Fax: +1 412 268-8200
Email: `lambert@psc.edu`

Table of Contents