MGEN User's and Reference Guide Version 5.0

Abstract

The Multi-Generator (MGEN) is open source software by the Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) group which provides the ability to perform IP network performance tests and measurements using UDP and TCP IP traffic. The toolset generates real-time traffic patterns so that the network can be loaded in a variety of ways. The generated traffic can also be received and logged for analyses. Script files are used to drive the generated loading patterns over the course of time. These script files can be used to emulate the traffic patterns of unicast and/or multicast UDP and TCP IP applications. The tool set can be scripted to dynamically join and leave IP multicast groups. MGEN log data can be used to calculate performance statistics on throughput, packet loss rates, communication delay, and more. MGEN currently runs on various Unix-based (including MacOS X) and WIN32 platforms.

The principal tool is the mgen program which can generate, receive, and log test traffic. This document provides information on mgen usage, message payload, and script and log file formats. Additional tools are available to facilitate automated script file creation and log file analyses.

Table of Contents

1. Quick Links

The Multi-Generator (MGEN) is open source software by the Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) group which provides the ability to perform IP network performance tests and measurements using UDP and TCP IP traffic. The toolset generates real-time traffic patterns so that the network can be loaded in a variety of ways. The generated traffic can also be received and logged for analyses. Script files are used to drive the generated loading patterns over the course of time. These script files can be used to emulate the traffic patterns of unicast and/or multicast UDP and TCP IP applications. The tool set can be scripted to dynamically join and leave IP multicast groups. MGEN log data can be used to calculate performance statistics on throughput, packet loss rates, communication delay, and more. MGEN currently runs on various Unix-based (including MacOS X) and WIN32 platforms.

The principal tool is the mgen program which can generate, receive, and log test traffic. This document provides information on mgen usage, message payload, and script and log file formats. Additional tools are available to facilitate automated script file creation and log file analyses.

- Mgen Usage
- Mgen Script File Format
- Mgen Log File Format
- Mgen Message Payload
- TRPR Log File Analysis Program
- gpsLogger GPS Utility
- Mgen Technical Documentation

2. Mgen Usage

The mgen version 5.0 program must currently be launched from a command-line. In the future, a simple graphical user interface similar to that of mgen version 3.x will be provided to simplify management of multiple sender and receiver instances. To launch mgen use the following command-line syntax:

```
mgen [ipv4][ipv6][input <scriptFile>][save <saveFile>]
     [output <logFile>][log <logFile>]
     [binary][txlog][nolog][flush][hostAddr {on|off}]
     [event "<mgen event>"][port <recvPortList>]
     [instance <name>][command <cmdInput>]
     [sink <sinkFile>][block][source <sourceFile>]
     [interface <interfaceName>][ttl <multicastTimeToLive>]
     [unicast_ttl <unicastTimeToLive>]
     [tos <typeOfService>][label <value>]
     [txbuffer <txSocketBufferSize>]
     [rxbuffer <rxSocketBufferSize>]
     [start <hr:min:sec>[GMT]][offset <sec>]
     [precise {on|off}][ifinfo <ifName>]
     [txcheck][rxcheck][check][stop]
     [convert <binaryLog>][debug <debugLevel>]
     [localtime <localtime>] [queue <queue>]
     [broadcast {on|off}] [logdata {on|off}]
     [loggpsdata {on|off}] [gpsfile <fileName>]
     [df {on|off}][analytics] [window <secs>]]
     [report] [suspend <flowId(s)>] [resume <flowId(s)>] [reset <flowId(s)>]
     [retry <count>[/<delay>]
```

2.1. Example Usage

To run mgen with script file "script.mgn" and log to stdout (by default):

```
mgen input script.mgn
```

To monitor ports 5000,5004,5005, and 5006 for received UDP traffic and log to a specific file "log.drc":

```
mgen port 5000,5004-5006 output log.drc
```

The "event" command can be used to achieve equivalent operation with the command-line syntax:

```
mgen event "listen udp 5000,5004-5006" output log.drc
```

The "event" command allows for use of mgen without script files for "quick and dirty" runs. In the future, MGEN will be capable of being dynamically scripted during run time with "event" commands passed to MGEN via inter-process communication.

Note: In previous versions, two different programs (each with different scripts) were used to separately generate and receive test traffic. The traffic generation capability of the former mgen program and the receive-side functionality of the Dynamic-Receiver (drec) program have now been integrated into a single executable mgen program.

The file extensions ".mgn" for MGEN scripts and ".drc" for MGEN log files are suggested conventions that users might wish to use for consistency. The ".drc" naming is in honor of the deprecated drec program.

The "sink" and "source" commands can be used to stream MGEN messages via alternative transport processes (e.g. reliable multicast, ssh, peer-to-peer protocols, etc). Here is an example using ssh to set up a TCP connection to a remote machine, start an mgen receiver at the remote machine to log receive messages, and attempt to transmit a 2 Mbps stream of MGEN messages via the ssh connection:

```
mgen event "ON 1 SINK DST 127.0.0.1/5001 PERIODIC [200 1250]" \ sink STDOUT output /dev/null
| ssh <remoteHost> sh -c "cat | \ mgen source STDIN output mgenLog.drc"
```

Note that the mgen executable must be present on the remote machine. Also note the importance of directing the MGEN sender's log output to /dev/null so that it doesn't get piped to the ssh process, mixed with the binary "sink" message stream.

2.2. Example Script

Below is an example MGEN script which generates two "flows" of UDP traffic and sends a single 1 megabyte TCP message. In this example, UDP flow 1 is sent to the loopback interface address (127.0.0.1) port 5001 and UDP flow 2 is sent to an IP multicast group on port number 5002. Flow 3 will send a 1 megabyte tcp "message" as it is turned off immediately after being started. (Note that the targeted TCP server must be listening for tcp connections on the port specified). Locally, a LISTEN command is used to monitor these (and other) ports so that mgen can receive its own traffic for demonstration purposes. This script illustrates the usage of a number of MGEN script commands.

```
# MGEN script begins here
# These are some "Transmission Event" script lines

# Originate two UDP flows
```

```
0.0 ON 1 UDP SRC 5001 DST 127.0.0.1/5001 PERIODIC [1 1024]
0.0 ON 2 UDP SRC 5002 DST 224.225.1.2/5002 PERIODIC [1 512]

# These script lines send a single 1 megabyte TCP mgen message
# Note that the "mgen message" will be received in multiple
# "mgen fragments" by the target node.

0.0 ON 1 TCP DST 10.0.0.1/5000 PERIODIC [1 1048576] COUNT 1

# Modify the pattern/rate of flow 2 4 seconds into the test

4.0 MOD 2 POISSON [10 1024]

# These are some "Reception Event" script lines
# Monitor some ports for UDP traffic

0.0 LISTEN UDP 5000-5002,6000,6003

# Join an IP multicast group
0.0 JOIN 224.225.1.2 INTERFACE eth0

# Join a SSM multicast group (Supported only for *nix)
#0.0 JOIN 232.1.1.1 SRC 25.25.25.1 INTERFACE eth0

# For WIN32, use the "PORT" option
0.0 JOIN 224.225.1.2 PORT 5002

# For OSX use the "interface" option if a default multicast route is not defined
#0.0 JOIN 224.1.2.3 interface en0

# Join a multicast group across a range of ports
0.0 JOIN 224.1.2.5 PORT 5005-5010

# On IPv6 systems set the port option when IPv4 group membership is requested
0.0 JOIN 224.1.2.4 port 5001

# Later, leave the group
5.0 LEAVE 224.225.1.2 INTERFACE eth0

# This SSM LEAVE is for UNIX (Currently SSM is supported only in UNIX)
#5.0 LEAVE 224.224.1.2 SRC 25.25.25.1 INTERFACE eth0

# Incrementally ignore some receive traffic
6.0 IGNORE UDP 5000-500
18.0 IGNORE UDP 5001,6000,6003

# More MGEN lines (terminate the flows)
10.0 OFF
110.0 OFF 2

# MGEN script ends here
```

3. Command-line Options

Some of these command-line options can also be included in MGEN script files as "global" commands or defaults. Note the command-line (or commands sent via MGEN's remote control interface) will always override settings from script files.

`ipv4`	Forces mgen to open sockets for IPv4 operation (i.e. AF_INET domain sockets) only. The default behavior for mgen is to open sockets with the domain based on environment and the type of IP addresses used in the script file used.
`ipv6`	Forces mgen to open sockets for IPv6 operation (i.e. AF_INET6 domain sockets) only. The default behavior for mgen is to open sockets with the domain based on en-

	vironment and the type of IP addresses used in the script file used.
`df {ON|OFF}`	Controls whether the DF fragmentation bit is set. {ON|OFF}
`input<scriptFile>`	Causes mgen to parse the given <scriptFile> at startup and schedule any transmission or reception events given in the script.
`save<saveFile>`	Causes mgen to save the sequence number state of any pending transmit flows and the current relative script "offset" time to <saveFile> in the form of an MGEN script. The <saveFile> may be used as an additional input script on a subsequent launch of mgento return mgen to the same state as when previously exited. See the equivalent global SAVE command for further detail on usage.
`output<logFile>`	Cause mgen to output logged information to the indicated <logFile>. By default, mgen will log to stdout. With the output command, an existing <logFile> of the same name will be overwritten. Use the log command to append to an existing log file.
`log<logFile>`	This is the same as the output command except that if <logFile> already exists, it will be appended instead of replaced.
`binary`	Causes mgen to save output logging information in a smaller-sized binary file format. This option should come before the output or log command.
`txlog`	This enables transmission logging. This results in SEND events being added to the log file every time a packet is sent by mgen.
`nolog`	This disables logging completely.
`flush`	This causes the output log file to be flushed with each line written. This is useful for real-time monitoring of MGEN logging
`hostAddr {on|off}`	Turning this option on causes mgen to include the "host" field in MGEN messages sent. The "host" field contains an educated guess of the machines local IP address to help identify the source of messages in log files. When the "host" field is present, MGEN log file SEND and RECV events contain a "host>" field indicating the sender's original address. This can be useful when Network Address Translation (NAT) or other tunneling occurs in test networks.
`event"<mgen event>"`	The event command allows the user to enter the equivalent of MGEN script lines into mgen via the command-line. Multiple event commands can be used to pass the equivalent of a multi-line script to MGEN. Note that MGEN script events generally contain spaces and thus must be encapsulated in quotes on the command line. Note that the <eventTime> may be omitted and the action indicated will be taken by mgen immediately. When the *event*

	command is issued during run-time, the <eventTime> (if provided) specifies a delay relative to the current time (e.g. the event will occur with after the given delay).
`instance<instanceName>`	If a pre-existing mgen application instance is _not_ already running, this command registers the running mgen program as an instance identified by the <instanceName>. On UNIX, this corresponds to a Unix-domain datagram socket named "/tmp/<instanceName>" being opened and monitored for MGEN commands (On WIN32, a "mailslot" named "\\.\mailslot\<instanceName>" is created and used). These interprocess channels allow for run-time control of mgen processes. This is the preferred methodology for run-time control of the mgen application.If an application instance as identified by the <instanceName> parameter is already running, any subsequent command-line options are transmitted to the remote instance already running, and the new mgen instance will then exit.This allows run-time control of possibly multiple background mgeninstances from the "shell" or via scripting. The *event* command may be used to dispatch MGEN script events to mgen instances at run-time.
`command{<path>|STDIN}`	This specifies a file or device which mgen will monitor for run-time command input. If the "STDIN" key is used, mgenmonitors the "stdin" (console) input which can provide a crude run-time user interface for mgen. Commands sent to mgen in this fashion must be delimited by line-breaks or the ';' character. See the instance command for a more flexible, and the preferred option for mgen run-time control.
`port<recvPortList>`	Causes mgen to monitor the given port numbers for received UDP traffic. The format of the <recvPortList> is a comma-delimited list of individual or inclusive ranges of port values (No spaces allowed in the list). Note this is the equivalent of a scripted 0.0 LISTEN UDP <recvPortList> reception event and can also be equivalently achieved with the *event* command using the syntax: `mgen event "LISTEN UDP <portList>"Example:mgen port 5000,5002,5005-5009`
`sink<sinkFile>`	Causes mgento use the file or device (e.g. stdout) indicated as a "sink" or destination for transmitted message flows of protocol type "SINK". I.e., MGEN message flows of type "SINK" are written to the "sink" device instead of to a UDP or TCP socket. Piping mgen output to stdout allows MGEN messages to use alternative transport provided by another process (e.g. ssh, norm, etc). The special <sinkFile> value "STDOUT" will direct MGEN SINK flows to the mgen process stdout.
`source<sourceFile>`	This is the complement to the *sink*

	command. This allows mgen to directly receive a binary stream of MGEN messaging from the <sourceFile> which may be the piped stdoutfrom another process (e.g. ssh, norm, etc). The special <sourceFile> string "STDIN" causes mgen to get input from its stdin stream. Messages read from the <sourceFile> (or stream) are time-stamped and logged in the MGEN log file as usual.
`start<hr:min:sec>[GMT]`	Causes mgen to delay processing events in script file relative to the indicated absolute time. The optional "GMT" keyword indicates the time is Greenwich Mean Time instead of the default local time. This command establishes an absolute time for the relative script time of 0.0 seconds.
`offset<sec>`	Causes mgen to skip <sec> seconds of relative time into the execution of the script file used. Note that if an absolute start time is given using the `start command,` the offset into the script will correspond to that absolute time. The default offset for MGEN is 0.0 seconds.
`precise{on|off}`	When the precise mode is enable, mgen performs polling (only as needed) to precisely time packet transmission. While this is sometimes helpful at high packet transmission rates, it comes at a cost of high CPU utilization by mgen. The default for this option is "off".
`ifinfo<interfaceName>`	This option can be used to have MGEN print a summary of statistics to stderr upon exit for the specified network interface. These stats include counts of frames sent/received. This can be used to augment/verify MGEN performance with or without logging enabled
`convert<binaryLogFile>`	Causes mgen to convert the indicated <binaryLogFile> to a text-based log file. The text-based log file information will be directed to stdout unless you specify a filename with the *output* or *log* command. Mgen will exit after the file conversion is complete.
`interface<interfaceName>`	Causes mgen to set the default network interface for IP multicast and/or root node flow transmission to <interfaceName>. <interfaceName> will override any default interface specified within an mgenscript file. <interfaceName> is a "per socket" attribute, and in its absence, MGEN will behave according to the operating system's default behavior.
`ttl <multicastTimeToLive>`	Causes mgen to set the hop count for IP multicast traffic generated by MGEN. <multicastTimeToLive> will override any default multicast ttl indicated within an mgen script file. <timeToLive> is a "per socket" attribute. If no ttl option is used, MGEN will set the default multicast ttl to 1.
`unicast_ttl <unicastTimeToLive>`	Causes mgen to set the hop count for IP unicast traffic generated by MGEN. <unicastTimeToLive> will override any default unicast ttl indicated within an mgen script file. <unicastTimeToLive> is a "per socket" attribute. If no unicast_ttl option is used, MGEN will set the default unicast ttl to 255.
`tos<typeOfService>`	Causes mgen to set the IPv4 type-of-service field (within the packet header) to <typeOfService>. <typeOfService> will override any default tos indicated within an mgen

	script file. As with ttl and interface, tos is a "per socket" attribute. If no tos option is used, MGEN will behave according to the operating system's default behavior.
`label<value>`	Causes mgen to set <value> as the *default* flow label for IPv6 flows. The <value> corresponds to the 28-bit IPv6 flow label field and may be specified in decimal or hex.
`txbuffer<bufferSize>`	Causes mgen to set the socket transmit buffer size to a value ?at least? as large as <bufferSize>. If <bufferSize> is larger that the maximum allowed by the system, <bufferSize> will be set to the system maximum.
`rxbuffer<bufferSize>`	Causes mgento set the socket receive buffer size to a value ?at least? as large as <bufferSize>. If <bufferSize> is larger that the maximum allowed by the system, <bufferSize> will be set to the system maximum.
`txcheck`	Causes mgen to include an optional 32-bit cyclic redundancy checksum (CRC) at the end of its messages. The `CHECKSUM` flag is set to indicate the presence of the checksum content.
`rxcheck`	Forces mgen receivers to validate the checksum portion (last 4 bytes) of MGEN messages whether or not the CHECKSUM flag is set in the MGEN "flags" message field. Use this option when it is _known_ that the MGEN sender is supplying checksums to cover the case when the "flags" field itself is possibly corrupted.
`check`	Sets mgen behavior as if both the *txcheck* _and_ *rxcheck* commands were applied. This is the recommended option when MGEN checksum operation is desired so that both senders and receivers are providing and validating checksums, respectively.
`stop`	This command causes mgen to exit. This is useful for runtime control of mgen instances.
`localtime`	This enables logging of events and error messages in localtime. By default, events are logged in Greenwich Mean Time.
`queue<queueSize>`	This global command will cause mgen to buffer <queueSize> mgen packets for each flow during periods of congestion. (Note that flow specific limits specified at the transmission event level will override this global). When the number of pending messages for a flow exceeds this limit, the message transmission timer will be temporarily deactivated and any pending messages will transmitted as quickly as possible. The timer will be reactivated once the pending message count falls below the queue limit, and message transmission will return to the previously scheduled rate of transmission. If no global command is specified, a default <queueSize> of "0" will be in effect which will result in no queuing behavior, e.g. the transmission timer will continue to fire at its regularly scheduled interval regardless of transport congestion. No pending message count will be accumulated and message transmission

	will suceed or fail depending on transport availability. See QUEUE for more details about the queueing mechanism.
`broadcast {on|off}`	Causes MGEN to set the socket option SO_BROADCAST to allow or disallow sending (and sometimes receiving) broadcasts from the socket. As with tos, ttl and interface, broadcast is a "per socket" attribute. By default BROADCAST is set to ON.
`logdata {on|off}`	Controls whether MGEN will log the optional data attribute field at MGEN receivers (including within MGEN binary log files). It does not affect whether MGEN senders send the requested data attribute. By default LOGDATA is set to ON.
`loggpsdata {on|off}`	Controls whether MGEN will log the gps data fields at MGEN receivers. It does not affect whether MGEN senders send the GPS data. By default LOGGPSDATA is set to ON. Note that as opposed to the logdata attribute, GPS data will be saved in any interim binary log file regardless of this flag. This flag only controls whether the gps data is logged in the formatted log files.
`boost`	The boost option sets the mgen process to realtime process priority. Care should be taken using the "precise" and "boost" options together as the mgen process can take over a machine at high packet rates (e.g. ctrl-c may not be handled).
`gpsfile <gpsFile>`	Changes the default location of the gps shared memory file to <gpsFile>
`analytics`	Enables built-in, per-flow analysis of received flows and outputs "REPORT" events to the mgen log file. The analysis (similar to NRL's TRPR tool) includes: 1. average received goodput rate for the given REPORT "window" time 2. loss fraction measured for the given REPORT "window" time 3. Average, minimum, and maximum latency for the given REPORT "window" time See analytics for more detailed information.
`window <secs>`	Sets the analysis measurement window for analytics. See analytics for more detailed information.
`report`	Enables analytic reporting in all transmitting flows. See report for more detailed information.
`suspend <flowId(s)|range>`	Suspends *local* mgen flow(s) matching the given flow id, list of comma separated flow ids, or range. See SUSPEND for more detailed information.
`resume <flowId(s)|range>`	Resumes *local* mgen flow(s) matching the given flow id, list of comma separated flow ids, or range. See RESUME for more detailed information.
`reset <flowId(s)|range>`	Resets *local* mgen flow(s) matching the given flow id, list of comma separated flow ids, or range.

	See RESET for more detailed information.
`retry <count>[/<delay>]`	Sets the default retry behavior for all data flows. If the underlying TCP socket for a flow fails to connect, or disconnects, attempt to reconnect the socket <count> times after <delay> seconds. The default retry interval is 5 seconds. The transport will attempt to reconnect over the life of the flow if the retry count is set to -1. If multiple flows are sharing the transport a common retry pattern should be used by all the flows or the last attribute set "wins".

4. MGEN Run-Time Remote Control

To use the mgen "remote control interface":

1. Start one (or more) instance(s) of mgen to control:

   ```
   mgen instance mgen1
   ```

2. Subsequent invocations of mgen with the same instance name will pass provided commands to the first instance and then exit:

   ```
   mgen instance mgen1 event "on 1 udp dst 127.0.0.1/5000 periodic [1 1024]"
   ```

The second instance (Step #2) will exit after it has passed its commands to the first running instance as identified by the <instanceName>. Note this can allow run-time control of multiple mgen instances by user(s), shell scripts, or other processes. A programmer comfortable with use of Unix-domain sockets (or WIN32 mailslots) or using the NRL Protolib "ProtoPipe" C++ class can also write software for run-time control of MGEN processes.

5. MGEN Script Format

MGEN scripts are text files containing a sequence of commands and scheduled events describing traffic generation patterns, ports and/or multicast groups to be monitored, and other options. Each line in the script corresponds to either a "Transmission Event", or "Reception Event", or "Global Command". Lengthy script lines can be continued to multiple text file lines by using a trailing backslash '\' character at the end of the line. Additionally, blank lines are permitted and comment lines can be included by providing a leading '#' character at the beginning of lines. (Note comment lines cannot be inserted in between "continued" script lines). Currently mgen is case-sensitive in parsing the script file format (commands, options, etc are all upper case), but will be modified to be case-insensitive in the future.

Scheduled transmission and reception events in the script use lines in the format of:

```
[<eventTime>] <eventType> <parameters ...> [<options ...>]
```

These "events" are scheduled to be executed by MGEN at the relative time given by the <eventTime> field. The value of this field is a floating point number which denotes the relative time (in seconds) of the associated event. The time is relative to the start of the MGEN program or the time dictated by the global START command. If the <eventTime> is omitted, an <eventTime> of 0.0 (or immediately if MGEN is already started) is assumed (This can used with MGEN's "event" command to directly control the operation of ns-2 Agent/MGEN instances within an ns-2 TCL (Tool Command Language) script without use of an external MGEN script.

Global commands are generally used to define default behaviors for MGEN operation or other options independent of event scheduling. The format for global command script lines is:

```
<commandType> [<command parameters ...>]
```

5.1. Transmission Events

MGEN "Transmission Event" script lines are used to schedule and characterize mgen traffic generation. An instance of mgen can simultaneously transmit traffic to multiple destinations with different patterns of transmission.

The MGEN script format uses "flow identifiers" (<flowIds>) to tag specific "threads" of MGEN traffic generation. While the <flowIds> are placed in the payload of associated MGEN messages, the primary purpose of the <flowId> is to simply tie together a sequence of script "transmission events" as a single "flow" or "thread".

The sequence of events pertaining to a "flow" of MGEN traffic generation consist of ON, MOD, and OFF . The script line syntax for these event types is:

```
<eventTime> {ON|MOD|OFF} <flowId> [<options ...>]
```

The first scripted event for a given flow identified by a <flowId> must be an ON event. Subsequently, MOD, events can be used to modify characteristics of the given flow until it is terminated with an OFF event. After a flow has been terminated with the OFF command, a flow with the same <flowId> value may be initiated with another ON event. The <options> fields are used to describe the characteristics of flows initiated with ON events and modified with subsequence MOD events. The OFF event uses no options.

5.1.1. ON Event

Script syntax:

```
<eventTime> ON <flowId> <protocol> [connect] DST <addr>/<port> <pattern [params]> [<options ...>] [DATA [<hex><hex>]]
```

This transmission event type is used to initiate a new flow at the time given by the <eventTime>. The <flowId> is used to identify the flow within the script and can be used by subsequent MOD, or OFF events to reference the flow initiated here.

The <protocol> field indicates the transport protocol to be used for the generated MGEN test messages. Current supported <protocol> types include "UDP", "TCP", and "SINK". The flow destination address and port must be specified for the ON event using the DST option and the <pattern> of message generation must be given as well. Other flow <options> may be specified to further characterize the flow. User defined message payload can be specified with the DATA command. The data should be a hexadecimal representation of the user data where each pair of characters corresponds to one byte of user data.

The "UDP" and "TCP" protocol types encapsulate generated MGEN messages for the flow into IP packets for the appropriate protocol and transmit them over the network. (Note that an mgen instance must be "listening" for a TCP connection on the destination port at the target node or the connection attempt will fail and the flow will be turned off). Messages for the "SINK" protocol type are written to the file/device/stream indicated by the mgen "sink" command-line option. In the future, other protocol types will be available for MGEN traffic flows.

The optional UDP CONNECT attribute will direct MGEN to open a "connected" UDP socket. If the connection cannot be established or is not available for a time period, MGEN will continue to attempt to send packets until the flow is stopped. (Note that Windows and some Unix implementations may not always report IGMP port unreachable messages returned by the destination address when a socket is not listening to the requested port.)

Example:

This script line will originate a "flow" of MGEN UDP destined for the loopback address (IP address 127.0.0.1) port number 5000 beginning immediately when the script is executed. The messages will consist of 1024 byte messages at a regular rate of 1.0 per second:

```
0.0 ON 1 UDP DST 127.0.0.1/5000 PERIODIC [1.0 1024]
```

5.1.2. MOD Event

Script syntax:

```
<eventTime> MOD <flowId> [<options ...>]
```

This transmission event type is used to modify the characteristics of an existing flow identified by the <flowId> field. The given transmission event <options> determine which specific characteristics of the flow (e.g. PATTERN, TOS, destination (DST), connection status (CONNECT), broadcast, etc) will be affected. Multiple options may be specified in the script line. Note that the protocol type and source port number (SRC) cannot be changed with the MOD event type (the referenced flow should be terminated with an OFF event and re-initiated with an ON event to accomplish this goal). If no <options>are given, the flow will remain unaltered. A script parse error will result if the identified flow was not previously initiated with an ONevent.

Example:

This script line will modify "flow 1" to change it packet transmission pattern 5.0 seconds after script execution. The changed "flow 1" will then generate messages 512 bytes in size at an average rate of 10.0 messages per second following a Poisson (exponentially-distributed interval)

```
5.0 MOD 1 POISSON [10.0 512]
```

```
Example:
```

```
These commands will connect a previously unconnected UDP socket. Note that the original
socket will be closed and any pending queue for the flow will be cleared.
```

```
ON 1 UDP SRC 5000 DST 10.0.0.1/5001 PERIODIC [1 1024]
```

```
5.0 MOD 1 CONNECT
```

```
Example:
```

```
These commands will disconnect a previously connected socket and change the source port
to 5001. To keep the socket connected you must also specify the CONNECT attribute on the
MOD command.
```

```
ON 1 UDP CONNECT SRC 5000 DST 10.0.0.1/5001 PERIODIC [1 1024]
```

```
5.0 MOD 1 SRC 5001
```

5.1.3. OFF Event

Script syntax:

```
<eventTime> OFF <flowId>
```

This transmission event type terminates message transmission for the flow identified by the <flowId> field at the time given in the <eventTime> field. There are no options applicable to this event type. A script parse error will result if the identified flow was not previously initiated with an ON event.

Example:

This script line will terminate generation of MGEN message traffic for "flow 1" at 10.0 seconds after script execution.

```
10.0 OFF 1
```

5.2. Transmission Event Options

This section describes options which may be applied to ON or MOD, transmission events in MGEN script files. Note that ON event lines require specification of at least the <protocol>, <destination>, and <pattern> options, while only the options to be changed need to be specified as part of MOD event lines.

5.2.1. Protocol (UDP/TCP/SINK)

Option syntax:

```
... <protocolType> ...
```

The transport protocol for MGEN messages generated by a flow must be specified as part of any ON events.

Example:

```
0.0 ON 1 UDP DST 127.0.0.1/5000 PERIODIC [1.0 1024]

0.0 ON 2 TCP DST 127.0.0.1/5000 PERIODIC [1.0 1024]

0.0 ON 3 SINK DST 127.0.0.1/5000 PERIODIC [1.0 1024]
```

5.2.2. Destination (DST)

Option syntax:

```
... DST <addr>/<port> ...
```

The destination address for a flow must be specified for ON events and may be altered as part of MOD, events. The <addr> field specifies the destination IP address (IPv4 or IPv6) and the <port> field specifies the destination host port number. The destination address may be a unicast (point-to-point) or multicast address.

Examples:

```
#Start a flow to loopback address port 5000

0.0 ON 1 UDP DST 127.0.0.1/5000 PERIODIC [1.0 1024]

#Modify flow 1 to a different destination port

0.0 MOD 1 DST 127.0.0.1/5001
```

5.2.3. Source Port (SRC)

Option syntax:

```
... SRC <port> ...
```

The source port number used for generated traffic may be optionally specified as part of an ON event. The <port> field identifies the host port number to be used. When the SRC option is not specified or set to "0", the flow will use a free port number picked by the operating system. Note that MGEN UDP flows may share the same source port and the associated flow will "inherit" some attributes (e.g.TOS, TTL, BROADCAST etc) which may have been set for other flows which use that same source port. This is because some of these attributes tend to be maintained by operating systems on a "per socket" basis. Also, any such attributes set for this flow will affect other existing flows using the same source port. Thus, the SRC option is useful when it is desired to explicitly create different flows with distinct "per socket" attributes such as TOS or TTL.

NOTE: Under the windows operating system, the ability to reestablish TCP connections to a common SRC addr/port DST addr/port pair is limited by TCP's TIME_WAIT interval which can range from 2 minutes to 30 seconds. During this operating system dependent interval, any attempt to reuse the socket pair will fail. Allowing the operating system to provide the SRC port will allow connections to a common dst/port to be successful within this interval. This behavior may also manifest under certain Linux distributions as well.

Example:

Here, two flows are created with the same destination address, but different source ports. Flow 1 is also assigned non-default type-of-service using the TOS option. The use of the SRC option ensures that two different sockets are used to support the two different types of service.

```
#Start flow 1 using source port 5001(TOS = 0x10) and flow 2 using port 5002

0.0 ON 1 UDP DST 127.0.0.1/5000 SRC 5001 PERIODIC [1.0 1024] TOS 0x10

0.0 ON 2 UDP DST 127.0.0.1/5000 SRC 5002 PERIODIC [10.0 512]
```

5.2.4. COUNT

Option syntax:

```
... COUNT <msgCount> ...
```

The optional COUNT attribute specifies the number of messages that are to be sent for the flow, e.g. a COUNT value of 1 means that one and only one mgen message will be sent. This attribute defaults to "-1", meaning mgen will send an unlimited number of messages until an OFF event occurs. If a message count is specified, the mgen flow will be paused after the requested number of messages has been sent.

Note that an OFF event will override any message COUNT specified (e.g. the flow will be terminated even if <msgCount> messages have not been sent) and that the QUEUE attribute will override (defer) OFF events.

If a flow is paused (e.g. packet rate is set to 0) before the packet count as been reached, the initial message limit will still be in effect when the flow is resumed, e.g. in the following example 10 total packets will be sent.

```
0.0 ON 1 UDP dst 127.0.0.1/5000 periodic [1 1024] count 10
# Pause flow
5.0 mod 1 periodic [0 1024]
# Resume flow
10.0 mod 1 periodic [1 1024]
60.0 off 1
```

To ensure that the count is only applicable to the first transmission command, reset the count to -1 (unlimited messaging) or another count value:

```
0.0 ON 1 UDP dst 127.0.0.1/5000 periodic [1 1024] count 10
# Pause flow
5.0 mod 1 periodic [0 1024]
# Resume flow with no message limit
10.0 mod 1 periodic [1 1024] COUNT -1
# Pause flow
15.0 mod 1 periodic [0 1024]
# Restart flow and send 5 packets
20.0 mod 1 periodic [1 1024] count 5
60.0 off 1
```

Message counts are kept on a per flow basis regardless of how many flows are sharing a transport.

5.2.5. Pattern (PERIODIC, POISSON, BURST, JITTER, CLONE, <sizeMin:sizeMax>)

Option syntax:

```
... <patternType> [parameters ...] ...
```

(Note: The '[' and ']' characters are explicitly required at the beginning and end of the pattern parameter set. Different pattern types may use different parameter sets.)

Traffic generated by MGEN consists of a series of sequence-numbered messages. The messaging generated by MGEN may vary in size and frequency of transmission to stress the network in a controlled fashion or possibly emulate other network applications. The "Pattern" of message generation must be specified in ON events and may be altered as part of subsequent MOD, events. Currently MGEN supports four pattern types, "PERIODIC", "POISSON", "BURST", "JITTER", and "CLONE". Complex traffic patterns can be created by using a compound of multiple "flows" (with the same SRC/DST) with different pattern types and parameters. Other pattern types (e.g. MARKOV), including ones with statistically varying payload sizes, will be added eventually.

5.2.5.1. PERIODIC Pattern:

Option syntax:

```
... PERIODIC [<rate> <size>]...
```

This pattern type generates messages of a fixed <size> (in bytes) at a very regular <rate> (in messages/second). For UDP protocol, the <size> field must be greater or equal to the minimum MGEN message size and less than or equal to the maximum UDP message size of 8192 bytes. For TCP protocol, <size> parameter is unlimited. Note the <rate> must be greater than or equal to 0.0 messages/second for the TCP and UDP protocols.

Example:

```
#Start an MGEN flow sending 1024 byte messages

#at a rate of 10.0 per second

0.0 ON 1 UDP DST 127.0.0.1/5000 PERIODIC [10.0 1024]

#Reduce the flow rate to one 512 byte message

#every 2.0 seconds

0.0 MOD 1 PERIODIC [0.5 512]
```

5.2.5.2. POISSON Pattern:

Option syntax:

```
... POISSON [<aveRate (msg/sec)> <size (bytes)>] ...
```

This pattern type generates messages of a fixed <size> (in bytes) at statistically varying intervals at an average <rate> (in messages/second). For UDP protocol, the <size> field must be greater or equal to the minimum MGEN message size and less than or equal to the maximum UDP message size of 8192 bytes. For TCP protocol, <size> parameter is unlimited. Note the <rate> must be greater than or equal to 0.0 messages/second for the TCP and UDP protocols.

Example:

```
#Start an MGEN flow sending 1024 byte messages

#at an average rate of 10.0 per second

0.0 ON 1 UDP DST 127.0.0.1/5000 POISSON [10.0 1024]

#Reduce the flow rate to an average of one

#512 byte message every 2.0 seconds

0.0 MOD 1 POISSON [0.5 512]
```

5.2.5.3. BURST Pattern:

Option syntax:

```
... BURST [REGULAR|RANDOM <aveInterval (sec)> <patternType> [<patternParams>] FIXED|EXPO-
NENTIAL <aveDuration (sec)>] ...
```

The BURST pattern generates bursts of other MGEN pattern types at a specified average interval. The first parameter of the BURST pattern is either "REGULAR" resulting in periodic burst uniformly distributed in time by the <aveInterval> value, or "RANDOM" which exponentially distributes the traffic generation bursts in time with an average burst interval as specified by the <aveInterval> parameter value. The characteristics of the MGEN messages generated during a burst is given by the <patternType> and associated <patternParams> parameters. The <patternType> may any MGEN pattern type including PERIODIC, POISSON, or, yes, even BURST. The <patternParams> must be appropriate for the given <patternType>. When a traffic generation burst occurs, its duration is either of a FIXED value as given by the <aveDuration> or a randomly varying duration with EXPONENTIAL statistics and an average duration as given by the <aveDuration> parameter.

An example use of the BURST pattern would be to roughly emulate the "talk spurts" which might result from Voice Over IP (VOIP) applications. As a voice conversation commences, a user's burst of activity (talk spurts) might be RANDOM with some average interval and the duration talk spurts approximate EXPONENTIAL statistics. > When the talk spurt (burst) occurs, the voice compression codec might generate messages following something like a PERIODIC flow with packet rates and packet sizes dependent upon the voice codec in use.

Other uses of the BURST pattern might be to roughly model message/packet generation occurring with random use of a network such as web browsing, etc. The BURST model provided by MGEN does not presuppose any specific traffic model, but might be useful in approximating some models of regular or intermittent network activity.

The average traffic generation rate for this pattern should be approximately the average transmission rate of the core <patternType> and <patternParams> multiplied by the burst duty cycle (<aveDuration> / <aveInterval>). Note that when average burst duration tends to exceed the average burst interval, the flow will tend to follow the characteristics of the core pattern (i.e. 100% duty cycle).

Example:

```
#Start a bursty MGEN flow with bursts of 1024 byte messages

#with a periodic rate of 10.0 messages per second. The

#bursts will occur at random intervals with an average

#interval from the start of one burst until the start of

#the next of 10.0 seconds. The duration of each burst is

#of exponential statistics with an average burst duration

#of 5.0 seconds.

0.0 ON 1 UDP DST 127.0.0.1/5000 BURST [RANDOM 10.0 PERIODIC [10.0 1024] EXP 5.0]
```

5.2.5.4. JITTER Pattern:

Option syntax:

```
... JITTER [<rate> <size> <jitterFraction>]...
```

This pattern type generates messages of a fixed <size> (in bytes) with the specified jitter pattern defined by the <rate> (in messages/second) and <jitterFraction>. The jitterFraction defines the interval of deviation from the rate and must be greater than zero and less than 0.5. A jitter pattern of "JITTER [1 1024 .5]" will result in packets being sent at a random interval between 0.5 seconds and 1.5 seconds. For the UDP protocol the <size> field must be greater or equal to the minimum MGEN message size and less than or equal to the maximum UDP message size of 8192 bytes. For TCP protocol, <size> parameter is unlimited. Note the <rate> must be greater than or equal to 0.0 messages/second for the TCP and UDP protocols.

Example:

```
#Start an MGEN flow sending 1024 byte messages

#at a random interval between 0.5 and 1.5 seconds.

0.0 ON 1 UDP DST 127.0.0.1/5000 JITTER [1.0 1024 .5]
```

5.2.5.5. CLONE Pattern:

Option syntax:

```
... CLONE [<fileType> <fileName> [<repeatCount>]]...
```

This pattern type will incrementally read a file of the specified <fileType> to determine mgen packet sizes and message transmission intervals. Currently only tcpdump binary files are supported. It is assumed that the tcpdump file has been filtered to contain only the traffic that is to be "cloned".

At the flow event start time mgen will send a packet corresponding to the size of the first packet read from the file. Note that mgen assumes the records contain IPv4 UDP headers and therefore subtracts 42 bytes from the captured frame size reported by tcpdump. The second packet will be read from the file and a the second mgen packet of the same size will be sent as scheduled by the interval between the first and second packets. When the file is rewound, the first packet will not be transmitted. The second packet in the file will be scheduled to be sent after the last packet in the file according to the interval between the first and second packets in the file.

At present the only valid <fileType> is "tcpdump". The tcpdump file must be in binary format (created with tcpdump's -w option) and is assumed to be filtered to contain only the traffic that is to be cloned.

<fileName> is the name of the file containing the data to be used as the template for the MGEN pattern timing (packet intervals) and packet size(s).

<repeatCount> is an optional parameter that specifies the number of times the file is to be processed. <repeat-Count> can be set to "-1" (the default), "0", or a positive integer. "-1" causes the file to be continuously read until the flow is stopped by an OFF event or the mgen program ends. "0" directs mgen to clone the file once and stop. A positive integer "N" indicates the number of repititions through the file, e.g. a value of 1 will cause the file to be read twice, once plus the repeat.

Out of sequence time stamps have been seen occasionally in tcpdump output. MGEN will schedule these packets for immediate transmission and if running at debug level 2, will log a warning message.

Note that some Linux distributions enable "segmentation/reassembly offload". This feature causes the network driver to do TCP segmentation and reassembly. In such case, larger packets than MGEN can clone will be logged in the pcap files. Disable this feature to successfully process this data. (e.g. ethtool --offload eth0 gso off; ethtool --offload eth0 tso off; ethtool --offload eth0 gro off).

Example:

```
#Start an MGEN flow and clone the contents of the specified file once
```

```
0.0 ON 1 UDP DST 127.0.0.1/5000 CLONE [tcpdump tcpdump.dat [0]]
```

5.2.5.6. Uniform random message size:

Option syntax:

```
... <patternType> [<rate> <sizeMin:sizeMax>]...
```

Setting a message size range will send uniformly random message sizes within the range sizeMin to sizeMax. The minimum <sizeMin> must be as big as the minimum allowed message size.

Example:

```
#Start an MGEN flow and one message per second, uniformaly random in size from 824 bytes
```

# to 1224 bytes (e.g. 1024 byes average).

```
0.0 ON 1 UDP DST 127.0.0.1/5000 PERIODIC [1 824:1224]
```

5.2.6. BROADCAST

Option syntax:

```
... BROADCAST {ON|OFF} ...
```

This sets the SO_BROADCAST socket option to enable or disable the sending (and sometimes receiving) of broadcast messages. By default BROADCAST is ON.

5.2.7. LOGDATA

Option syntax:

```
... LOGDATA {ON|OFF} ...
```

Controls whether MGEN will log the optional data attribute field at MGEN receivers (including within MGEN binary log files). It does not affect whether MGEN senders send the requested data attribute. By default LOGDATA is ON.

5.2.8. Type-Of-Service (TOS)

Option syntax:

```
... TOS <value> ...
```

The IP TOS (type-of-service) field can be controlled for IP packets associated with MGEN traffic generation. The <value> field specifies the value of the 8-bit TOS field in IPv4 packets. (IPv6 packets do not have a TOS field. MGEN will soon support control of the similar FLOW_ID field for IPv6 operation.) The <value> field must be in the range of 0-255 in decimal or hexadecimal notation. The interpretation of the TOS value by different computer operating systems and network devices may vary. In some cases, computer hosts will not allow all possible values to be used, and in others "super user" (root) privileges may be required to set the IP TOS field to certain values. Below are some notes on suggested interpretation by the Internet Engineering Task Force (IETF). Note that TOS is maintained on a "per socket" basis and that setting the TOS for a flow will affect other flows sharing the same network socket. See the SRC option to make sure different flows use different sockets.

Example:

```
#Start flow 1 with default TOS

0.0 ON 1 UDP DST 127.0.0.1/5000 PERIODIC [1.0 1024]

#Modify flow 1 to TOS = 0x10 (low delay)

5.0 MOD 1 TOS 0x10
```

Notes on the value of the IP TOS field:

```
  0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+-----+
|   PRECEDENCE    |          TOS          | MBZ |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

The Type-of-Service byte in the IP header is divided into three sections: the Precedence field (high-order 3 bits), a field that is called Type of Service or TOS (next 4 bits), and a reserved bit (the low order bit). The TOS bits can be set to 5 different settings including the default setting of 0000, while the PRECEDENCE can be set to 8 different setting including default 000.

TOS definitions:

	IPTOS_TOS_MASKIP-TOS_TOS(tos)	0x1E= ((tos) & IP-TOS_TOS_MASK)	
1000 --	IP_TOS_LOWDELAY	0x10	TOS = 16
0100 --	IP_TOS_THROUGHPUT	0x08	TOS = 8
0010 --	IPTOS_RELIABILITY	0x04	TOS = 4
0001 --	IPTOS_LOWCOST	0x02	TOS = 2
0000 --	normal service	0x00	TOS = 0

Precedence definitions:

111 --	IPTOS_PREC_MASKIP-TOS_PREC(tos)	0xe0= ((tos) &IP-TOS_PREC_MASK)	
111 --	IPTOS_PREC_NETCON-TROL	0xe0	TOS = 224

110 --	IPTOS_PREC_INTER-NETCONTROL	0xc0	TOS = 192
101 --	IP-TOS_PREC_CRITIC_ECP	0xa0	TOS = 160
100 --	IP-TOS_PREC_FLASHOVER-RIDE	0x80	TOS = 128
011 --	IPTOS_PREC_FLASH	0x60	TOS = 96
010 --	IP-TOS_PREC_IMMEDIATE	0x40	TOS = 64
001 --	IPTOS_PREC_PRIORI-TY	0x20	TOS = 32
000 --	IPTOS_PREC_ROUTINE	0x00	TOS = 0

If `TOS = 164 (or 0xa4)`, the Precedence would be `IPTOS_PREC_CRITIC_ECP` and the `TOS` would be `IP-TOS_RELIABILITY`. The `IP TOS` field bits would be set as `10100100`.

5.2.9. Multicast/Unicast Time-To-Live (TTL)

Option syntax:

```
... TTL <value> ...
```

The time-to-live (TTL) hop count can be controlled for IP multicast traffic or Ip unicast traffic (as defined by the dstAddr) generated by MGEN. As with TOS, this is generally a "per socket" attribute and care should be taken if it is desired to specify different TTL values for different MGEN flows. This can be accomplished by using different SRC (source ports) for different MGEN flows. The <value> field must be in the range of 1-255. The default multicast TTL assumed by MGEN is 1. The default unicast TTL assumed by MGEN is 255.

Example:

```
#Start an IP multicast flow with a maximum multicast hop count ttl = 2

0.0 ON 1 UDP DST 224.1.2.3/5000 PERIODIC [1.0 256] TTL 2

#Start an IP unicast flow with a maximum unicast hop count ttl = 2

0.0 ON 1 UDP DST 10.0.0.1/5000 PERIODIC [1.0 256] TTL 2
```

5.2.10. Socket Transmit Buffer Size (TXBUFFER)

Option syntax:

```
... TXBUFFER <txBufferSize> ...
```

This option allows users to set the socket transmit buffer size to a value at least as large as <txBufferSize>. If <txBufferSize> is larger that the maximum allowed by the system, <txBufferSize> will be set to the system maximum. To date, this option has only been tested on *linux* systems.

5.2.11. Socket Receive Buffer Size (RXBUFFER)

Option syntax:

```
... RXBUFFER <rxBufferSize> ...
```

This option allows users to set the socket receive buffer size to a value at least as large as <rxBufferSize>. If <rxBufferSize> is larger that the maximum allowed by the system, <rxBufferSize> will be set to the system maximum. To date, this option has only been tested on *linux* systems.

5.2.12. IPv6 Flow Label (LABEL)

Option syntax:

```
... label <value> ...
```

This option allows users to specify the value applied to the IPv6 packet header "flow label" field. Although this field is 28 bits, different operating systems may restrict which portions of the field may be set. For example, the current Linux kernel (circa Jan 2003) only allows bits in the first octet of the flow label to be set. Other values are invalid on Linux, and generate an error message. Thus, using hexadecimal format for the <value> specified, legal values for Linux are restricted to `<value> = 0x0??00000` where "??" specifies the first octet of the flow label. Other operating systems may behave differently.

Example:

```
# Start an IPv6 flow with flow label = 0x03d00000

0.0 ON 1 UDP LABEL 0x03d00000 SRC 5000 DST5f1b:df00:ce3e:e200:0800:2078:e3e3/5001 PERIODIC
[1 1024]
```

5.2.13. Multicast Interface (INTERFACE)

Option syntax:`... INTERFACE <interfaceName> ...`

The network interface to use for IP multicast flow transmission can be controlled with this option. The <interfaceName> is the network interface device name to be used for IP multicast transmission for the associated flow. Again, as with TOS and TTL, this is generally a "per socket" attribute and care should be taken if it is desired to specify different multicast interfaces for different MGEN flows. This can be accomplished by using different SRC (source ports) for different MGEN flows. If no INTERFACE option is used, MGEN will behave according to the operating system's default behavior.

Example:

```
#Start an IP multicast flow on Ethernet interface named "eth1"

0.0 ON 1 UDP DST 224.1.2.3/5000 PERIODIC [1.0 256] INTERFACE eth1 SRC 5001
```

5.2.14. Sequence Number Initialization (SEQUENCE)

Option syntax:

```
... SEQUENCE <sequenceNumber> ...
```

This option sets the sequence number of the next message transmitted for the flow. MGEN flows are normally initialized to a sequence number of zero upon the first "ON" event for the flow. The sequence number is incremented by one with each message transmitted. The SEQUENCE option allows the user to override this behavior. It (along with the OFFSET command) is used by the SAVE command with MOD events for pending flows when it is desired that mgen return to a particular point in a script after being stopped and restarted.

Example:

```
#Modify the sequence number of an existing flow such that

#the next message is transmitted with sequence number 452.

12.0 MOD 1 SEQUENCE 452
```

5.2.15. UDP Connect (CONNECT)

Option syntax:

... CONNECT ...

The optional UDP CONNECT attribute will direct MGEN to open a "connected" UDP socket. If the connection cannot be established or is not available for a time period, MGEN will continue to attempt to send packets until the flow is stopped.

Example:

```
#Open up a CONNECTED UDP socket

1.0 ON 1 UDP CONNECT DST 10.0.0.1/500 PER [1 1024]
```

5.2.16. DF (fragmentation bit)

Option syntax:

... DF {ON|OFF} ...

The optional DF command controls whether the df fragmentation bit is set for the flow.

Example:

```
#Set fragmentation bit

1.0 ON 1 UDP DST 10.0.0.1/500 PER [1 1024] DF ON
```

5.2.17. REPORT

Option syntax:

... REPORT ...

Instructs the flow to include per-flow analytic reports in its transmitted MGEN message payload. MGEN nodes that receive these "report" payloads will log REPORT events in their log files in addition to RECV events if the analytics global has been set at the receiving node. The reporting is "in-band" of scripted message generation. The typical size of a per-flow report is 24-28 bytes but is "hidden" in the normally "dummy" payload content of MGEN messages so it doesn't really add overhead. The metrics reported are efficiently quantized to enable the compact reports and that enables reporting potentially on many received flows in a single transmitted MGEN message.

The "in-band" reports are subject to the message size and rate limits of the transmitting flow(s). A round-robin scheme, with precedence to unreported and active flows, is used to populate the message payload content with reports to provide assurance of delivery of relevant report. Note that reports are sent redundantly when possible when the message rate is sufficiently high. This provides some assurance of report delivery even with message loss. Post-processing of log files will require filtering of the redundant REPORT lines logged.

An example flow command that enables this analytic reporting is:

```
mgen event "ON 1 UDP DST 127.0.0.1/5000 PER [1 1024] report" window 10.0
```

Here an MGEN message with report content for any analytics for flows received by that MGEN node would be sent once per second. The report content would be updated once per 10 seconds but the receiver(s) of this reporting flow would still log a report once per second. Of course, an MGEN node receiving nothing will have nothing to report unless it has received prior flows. The REPORT "offset" will become "stale" (a large value) for flows that have become inactive.

Locally generated REPORT events can be distinguished by a 0.0 offset> value and have the format:

```
01:17:01.983235 REPORT proto>UDP flow>3 src>127.0.0.1/63684 dst>127.0.0.1/5002 offset>0.000000 window>1.9
```

REPORT events from remote nodes include "sent>" timestamp:

```
01:17:01.983235 REPORT proto>UDP flow>3 src>127.0.0.1/63684 dst>127.0.0.1/5002 sent>01:16.58.675309 offse
```

The interpretation of the "sent>" field and the "offset>" fields is the reported analytics measurement window, *End* time is "sentTime - offset". The "offset" is a quantized delay time (in seconds) from the time the included analytics were computed (at window end) until they were included in the MGEN message payload sent by the remote node.

When filtering duplicate REPORT items in a log file containing REPORTs received from remote nodes, the filtering algorithm should favor the REPORT event (among duplicates) that has the smallest "offset" value. The quantization algorithm used to generate compact reports yields more precision for small "offset" values as compared to large "offset" values.

The usual time synchronization requirements for latency measurement validity still stand, but it should be noted that the latency "max/min" can be used to assess flow message delivery "jitter" regardless of the validity of the absolute average latency measurement.

There is also a global "report" command that can be used to make all transmitting flows include the analytic reports by default, but thought should be given to what reporting is required (i.e., a single flow can report for all received flows and multiple reporting flows will be redundant even though this reporting is "in-band" of scripted message generation).

5.2.18. SUSPEND

Option syntax:

... SUSPEND <flowId(s)|range> ...

Uses in-band MGEN message payload content to suspend *remote* MGEN flows matching the given flow id, comma separated list of flow id(s), or flow id range.

Example Usage:

```
mgen event "ON 1 UDP DST 127.0.0.1/5000 PER [1 1024] suspend 1,3-5"
```

Typically, it is expected that this command be used to manage remote flows that were specified with a "count" limit option, but this is not a requirement. Currently, the remote controlled flows configured MUST have flowId values in the range of 1-40. (This could be expanded with a minor update to the source code if a larger number of flowId values need to be managed). Here is an example scenario:

Remote MGEN (listening on port 5001 and sending to destination port 5002):

```
mgen event "listen udp 5001" event "on 1 udp dst 127.0.0.1/5002 per [1 1250]"
```

Local MGEN (listening on port 5002 and sending to destination port 5001):

```
mgen event "listen udp 5002" event "on 1 udp dst 127.0.0.1/5001 per [1 1250]"
```

At this point both the "remote" and "local" MGEN instances will be sending their flows and receiving from one another. As an example, the "local" MGEN can command the remote to suspend transmission of its flow '1' using:

```
mgen event "listen udp 5002" event "on 1 udp dst 127.0.0.1/5001 per [1 1250] suspend 1"
```

Note that these suspend/resume/reset flow options can have comma-delimited ranges of flowId values so the following examples are valid:

```
mgen event "on 1 udp dst 127.0.0.1/5001 per [1 1250] suspend 1-3,5,8-11
```

Also, multiple options can be specified. For example:

```
mgen event "on 1 udp dst 127.0.0.1/5001 per [1 1250] suspend 2,4, resume 5"
```

The "suspend" and "resume" options simply pause and resume flow transmission, respectively. The "reset" command can be used to act as a "keep-alive" for remote flows that have the "count" option set. When a "reset" command is received, the sender flow message counter is reset so the flow will continue transmitting for as long as "reset" commands are actively received. When "reset" reception stops, the count-limited flow will end transmission once its count decrements to zero per usual.

The use of the "reset" command can act as a "safe", fault-tolerant mechanism so that a remote MGEN will automatically terminate its transmission when messages are no longer received from the local MGEN controller. Thus, if a network become congested, etc., the transmissions will halt. Similarly, in wireless networks, transmissions by the remote would eventually cease when out of communications range, but then resume once back in range and once again actively receiving the "reset" commands from the controller. Note that remote flows that don't have a "count" limit will *not* expire and the "suspend" option must be used to pause those.

Care should be taken to have a single controller managing a remote MGEN. Note that multicast flow commands can be sent as well, but again, care should be taken. TCP flows could even bue used for the commands and reporting if desired.

The "reserver" field in the MGEN message format (byte prior to the "payload_length" field is now used as a "payload_type" to indicate whether the payload is MGEN_DATA or USER_DATA (i.e. user-defined content). When the "report" flow is set, this supersedes any user-defined payload content set for the flow. Hence, if one is also using user-defined payloads, one should be judicious in which flows are configured with the "report" option. Both analytic report content and remote flow command content can be carried in the MGEN_DATA payload type. Supported report and command content may be extended over time.

HINT: MGEN "mod" events can be used to dynamically control these features. For example, the averaging "window" can be changed on the fly, etc. as well as dynamic use of the "suspend", "resume", and "reset" remote control commands as desired.

5.2.19. RESUME

Option syntax:

... RESUME <flowId(s)|range> ...

Uses in-band MGEN message payload content to resume *remote* MGEN flows matching the given flow id, comma separated list of flow id(s), or flow id range.

For example:

```
mgen event "ON 1 UDP DST 127.0.0.1/5000 PER [1 1024] resume 1-3,5,8-11"
```

Typically, it is expected that this command be used to manage remote flows that were specified with a "count" limit option, but this is not a requirement. Currently, the remote controlled flows configured MUST have flowId values in the range of 1-40. (This could be expanded with a minor update to the source code if a larger number of flowId values need to be managed). See SUSPEND for usage examples.

5.2.20. RESET

Option syntax:

... RESET <flowId(s)|range> ...

Uses in-band MGEN message payload content to reset *remote* MGEN flows matching the given flow id, comma separated list of flow id(s), or flow id range.

For example:

```
mgen event "ON 1 UDP DST 127.0.0.1/5000 PER [1 1024] reset 1-3,5,8-11"
```

The "reset" command can be used to act as a keep-alive for remote flows that have the COUNT option set. When a "reset" command is received, the sender flow message counter is reset so the flow will continue transmitting for as long as "reset" commands are actively received. When "reset" reception stops, the count limited flow will end transmission once its count decrements to zero per usual. Note that the sequence number for the remote flow is *not* reset.

The use of the "reset" command can act as a "safe", fault-tolerant mechanism so that a remote MGEN will automatically terminate its transmission when messages are no longer received from the local MGEN controller. Thus if a network becomes congested, etc., the transmissions will halt. Similarly, in wireless networks, transmissions by the remote would eventually cease when out of communications range, but then resume once back in range and once again actively receiving the "reset" commands from the controller. Note that remote flows that don't have a "count" limit will *not* expire and the "suspend" option must be used to pause these.

Currently, the remote controlled flows configured MUST have flowId values in the range of 1-40. (This could be expanded with a minor update to the source code if a larger number of flowId values need to be managed). See SUSPEND for an example scenario.

5.2.21. RETRY

Option syntax:

... RETRY <count>[|<delay>] ...

If the underlying TCP socket for the flow fails to connect, or disconnects, attempt to reconnect the socket <count> times after <delay> seconds. The default retry interval is 5 seconds. The transport will attempt to reconnect over the life of the flow if the retry count is set to -1. If multiple flows are sharing the transport a common retry pattern should be used by all the flows or the last attribute set "wins".

For example:

Attempt to connect every five seconds:

```
ON 1 TCP DST 127.0.0.1/5000 PERIODIC [1 1024] RETRY -1/5
```

Attempt to connect twice after initial failure every 3 seconds

```
ON 1 TCP DST 127.0.0.1/5000 PERIODIC [1 1024] RETRY 2/3
```

RECONNECT events are logged at each reconnect attempt:

```
17:36:12.762619 RECONNECT flow>1 srcPort>35859 dst>127.0.0.1/5000
17:36:17.763483 RECONNECT flow>1 srcPort>46864 dst>127.0.0.1/5000
17:36:22.764783 RECONNECT flow>1 srcPort>46866 dst>127.0.0.1/5000
```

5.3. Reception Events

For simple reception and logging of unicast traffic, it is generally sufficient just to launch mgen with the *port* command line option specifying the port numbers to monitor. However, for IP multicast operation or more complex behavior, an MGEN script with "Reception Events" is required.; "Reception Events" in the MGEN script file format include LISTEN and IGNORE types to control which ports are being monitored when; and JOIN and LEAVE types to dynamically control IP group membership. The MGEN script syntax of "Reception Events" is:

```
<eventTime> <eventType> <parameters ...> [<options ...>]
```

5.3.1. LISTEN

Script syntax:

```
<eventTime> LISTEN <protocol> <portList>
```

The LISTEN event is used to prompt mgen to begin monitoring one or more ports for received traffic. The <eventTime> denotes the time (in seconds) relative to script execution. The <protocol> field specifies the transport protocol type. Currently, "UDP" and "TCP" transports are supported. The <portList> field is a comma-delimited list of individual or inclusive ranges of the port numbers (no spaces allowed) to begin monitoring. Port ranges within the list are specified in the format "<lowValue>-<hiValue>".

Example:

```
#Monitor UDP port numbers 5000, 5003, 5004, 5005, 5009

#and TCP port number 6000, 6003, 6004, 6005
```

```
#beginning at time 0.0

0.0 LISTEN UDP 5000,5003-5005,5009

0.0 LISTEN TCP 6000,6003-6005
```

5.3.2. IGNORE

Script syntax:

```
<eventTime> IGNORE <protocol> <portList>
```

The IGNORE event type is the converse to the LISTEN event type. An IGNORE event causes mgen tostop monitoring (and logging) received traffic on the specified <portList>. The <eventTime> denotes the time (in seconds) relative to script execution. The <protocol> field specifies the transport protocol type. "UDP" and "TCP" transports are supported. The <portList> field is a comma-delimited list of individual or inclusive ranges of the port numbers (no spaces allowed) to begin monitoring. Port ranges within the list are specified in the format `"<lowValue>-<hiValue>"`.

Example:

```
#Stop monitoring UDP port numbers 5003, 5004, 5005, 5009

#and TCP port numbers 6003, 6004, 6005

#beginning at time 10.0

10.0 IGNORE UDP 5003-5005,5009

10.0 IGNORE TCP 6003-6005
```

5.3.3. JOIN

Script syntax:

```
<eventTime> JOIN <groupAddress> [SRC <srcAddress>] [INTERFACE <interfaceName>] [PORT <portNumber| portList>]
```

The JOIN event is used to prompt mgen to "join" the specific IP multicast group indicated by the <groupAddress> field. The SRC option can be used to join a source specific multicast (SSM) channel. Note that the SRC option is not presently available on windows. The INTERFACE option forces the membership join request on the network interface identified by the <interfaceName> field. If no INTERFACE option is given, the operating system's default behavior is observed. Note it is possible to join the same group on multiple, different interfaces.

The PORT option should be used on WIN32 systems where the IP multicast join must be performed on the same socket bound to a specific <portNumber>.

Unix-based operating systems generally allow for IP multicast group membership to be independent of specific socket port bindings. Note that a corresponding LISTEN event for the indicated <portNumber> is required in order to receive traffic if not set as a JOIN command option. Specify the PORT option as a JOIN command option if your Unix based system does not support port binding independence.

Not that on IPv6 systems (and when mgen is compiled with the HAVE_IPV6 compile option) the JOIN port option must be specified when joining an IPv4 group.

On OSX systems, the interface option must be used if a default multicast route is not defined on the system.

As many IP group memberships as the operating system will support is permitted by mgen. This is generally a limit of the maximum number of open sockets per process or in the system at large if multiple mgen instances are used. Note that WIN32 imposes a limitation of one IP multicast group membership per socket while Unix-based systems can allow for many memberships (often 20, but OS-specific) per socket.

Examples:

```
#JOIN group 224.1.2.3 at time 0.0

0.0 JOIN 224.1.2.3

#JOIN group 224.1.2.4 on interface "eth1"

0.0 JOIN 224.1.2.4 INTERFACE eth1

#JOIN SSM channel 232.1.1.1 with source 26.26.26.1 on interface "eth1"

0.0 JOIN 224.1.2.4 SRC 25.25.25.1 INTERFACE eth1

#JOIN group 224.1.2.5 using socket bound to port 5000

0.0 JOIN 224.1.2.5 PORT 5000
```

#JOIN group 224.1.2.6 using sockets bound to ports 5001-5005

0.0 JOIN 224.1.2.6 PORT 5001-5005

5.3.4. LEAVE

Script syntax:

```
<eventTime> LEAVE <groupAddress> [SRC <srcAddress>] [INTERFACE <interfaceName>] [PORT <port-
Number]
```

The LEAVE event is used to prompt mgen to "leave" the specific IP multicast group indicated by the <groupAddress> field. The <groupAddress> must have been joined with a prior JOIN event. The INTERFACE and/or PORT options *must* be used if they were used with the corresponding JOIN event. Note that the SSM SRC option is not presently available on windows.

Examples:

```
#LEAVE group 224.1.2.3 at time 10.0

10.0 LEAVE 224.1.2.3

#LEAVE group 224.1.2.4 on interface "eth1" at time 10.0

10.0 LEAVE 224.1.2.4 INTERFACE eth1

#LEAVE SSM channel 232.1.1.1 with source 25.25.25.1 on interface "eth1" at time 10.0

10.0 LEAVE 224.1.2.4 SRC 25.25.25.1 INTERFACE eth1

#LEAVE group 224.1.2.4 on interface "eth1"and port 5000 at time 10.0

10.0 LEAVE 224.1.2.4 INTERFACE eth1 PORT 5000
```

6. Global Commands

The MGEN script file format supports a subset of commands which are independent of normal transmission and reception event scheduling. These are referred to as "Global Commands". This subset includes commands to specify an absolute script execution start time (START command) and to specify default traffic generation characteristics (e.g. multicast INTERFACE, multicast TTL, IP TOS, etc).

The format of script lines containing global commands is:`<command> <parameters>`

In general, a script file should contain only one occurrence of each global command type. If there are multiple occurrences of a command type, the last occurrence will determine mgen's behavior. These commands can also be given on the mgen command-line. Global commands given on the mgen command-line will override any corresponding global commands in the script file(s).

The MGEN "Global Command" set includes:

START	Specifies an absolute start time for script processing.
OFFSET	Specifies an offset time into script for MGEN activity.
TOS	Specifies a default IPv4 TOS value for IPv4 flows.
LABEL	Specifies a default IPv6 Flow Label for IPv6 flows.
TTL	Specifies a default TTL (time-to-live) hop count for transmitted multicast packets.
UNICAST_TTL	Specifies a default TTL (time-to-live) hop count for transmitted unicast packets.
INTERFACE	Specifies the name of the default interface to use for IP multicast.
INPUT	Specifies the name of a script file to be loaded and parsed.
OUTPUT	Specifies the name of the log file to record logged events. If the named log file pre-exists, it is overwritten.
LOG	Same as OUTPUT, except that pr-existing log files are appended instead of overwritten.
SAVE	Specifies a file to which MGEN should store sequence number state for any pending or active flows as well as the current relative script offset time when mgen is terminated.
TXBUFFER	Specifies a default socket transmit buffer size.
RXBUFFER	Specifies a default socket receive buffer size.
LOCALTIME	Specifies that mgen events and error messages are logged in localtime rather than the default Greenwich Mean Time.
QUEUE	Specifies the default number of mgen messages that should be queued before mgen turns off the transmission timer for a flow.
LOGDATA	Controls whether MGEN will log the optional data attribute field at MGEN receivers (including within MGEN binary log files). It does not affect whether MGEN senders send the requested data attribute. By default LOGDATA is set to ON.
IPV6	Forces mgen to open sockets for IPv6 operation (i.e. AF_INET6 domain sockets) only. The default behavior for mgen is to open sockets with the domain based on environment and the type of IP addresses used in the script file used.
IPV4	Forces mgen to open sockets for IPv4 operation (i.e. AF_INET domain sockets) only. The default behavior for mgen is to open sockets with the domain based on environment and the type of IP addresses used in the script file used.
DF	Controls whether the DF fragmentation bit is set. {ON|OFF}
ANALYTICS	Enables built-in, per-flow analysis of received flows and outputs "REPORT" events to the mgen log file. See analytics for more detailed information.

WINDOW	Sets the analysis measurement window for analytics. See analytics for more detailed information.
REPORT	Enables analytic reporting in all transmitting flows. See report for more detailed information.

SUSPEND <flowId|range|all>	Suspends a *local* flow (i.e. as opposed to the control of remote flows that inserting this options into a flow event provides). The main use for this would be to suspend flows upon startup so that they may be activated later upon receipt of remote flow command "resume" or "reset" messages. For example:
	`mgen event "on 1 udp dst 127.0.0.1/5001 per [1 1024]" suspe`
	would create a flow with flowid "1" that is immediately suspended (e.g. not transmitting) upon startup. Then later, if a 'resume' or 'reset' remote flow command is received, the flow would be activated and begin transmitting upon command. They keyword "all" can also be used with the suspend command (but not with the corresponding flow event option) to take action on *all* flows. For example:
	`mgen event "on 1 udp dst 127.0.0.1/5001 per [1 1024]" event`
	would create two flows that are suspended upon MGEN startup.

RESUME <flowId|range|all>	Resumes a *local* flow (i.e. as opposed to the control of remote flows that inserting this options into a flow event provides). The main use for this would be to resume flows that have been previously suspended in a local mgen instance. For example:
	`mgen instance mgen1 event "on 1 udp dst 127.0.0.1/5001 per`
	would create a flow with flowid "1" that is immediately suspended (e.g. not transmitting) upon startup. Then later, the global 'resume' command can be used to activate the suspended flow.
	`mgen instance mgen1 resume 1`
	They keyword "all" can also be used with the resume command (but not with the corresponding flow event option) to take action on *all* flows. For example:
	`mgen instance mgen1 resume all`
	would resume all flow(s) suspended in mgen instance mgen1.

RESET <flowId|range|all>	Resets a *local* flow (i.e. as opposed to the control of remote flows that inserting these options into a flow event provides). The main use for this would be to reset flows managed by local mgen instances. When a "reset" command is received, the sender flow message counter is reset and the flow will continue transmitting count messages. For example: `mgen instance mgen1 event "on 1 udp dst 127.0.0.1/5001 per`

	would create a flow with flowid "1" that is scheduled to send 10 messages upon startup. Then later, when a 'reset' remote flow command is received, the flow would be activated and begin retransmitting 10 messages. For example: flows. For example: `mgen instance mgen1 reset 1` would cause the counter for the flow to be reset and an additional 10 messages would be sent. If the counter has already expired the reset message will have no effect.
RETRY<count>[/<delay>]	Sets the default retry count and delay for all data flows. If the underlying TCP socket for the flow fails to connect, or disconnects, attempt to reconnect the socket <count> times after <delay> seconds. The default retry interval is 5 seconds. The transport will attempt to reconnect over the life of the flow if the retry count is set to -1. If multiple flows are sharing the transport a common retry pattern should be used by all the flows or the last attribute set "wins".

6.1. START

Script syntax:

```
START <hour:min:sec>[GMT]
```

The START command designates an absolute time as indicated by the `<hour:min:sec>` field to correspond to the relative script time of 0.0 seconds. All transmission and reception events will be scheduled relative to this absolute start time. The optional GMT suffix (no white space after the time) indicates that the clock time given is Greenwich Mean Time (GMT) rather than the operating systems local time zone. If no START command is given, mgen schedules transmission and reception events relative to program startup.

Example:

```
#Start MGEN exactly at 1:30PM local time

START 13:30:00

#Start MGEN at 30 seconds past 8:30 GMT

START 8:30:30GMT
```

6.2. OFFSET

Script syntax:

```
OFFSET <seconds>
```

The OFFSET global command specifies a relative time offset (in seconds)into script processing where MGEN should begin its activity. This allows multiple instances of MGEN using the same script to be dithered as desired. Additionally, this command may be used to immediately restore MGEN to a specific scripted state other than the beginning of the script upon launch.

Example:

#Skip the first 10 seconds of an MGEN script

```
OFFSET 10.0
```

6.3. TOS

Script syntax:

```
TOS <value>
```

The TOS command specifies ...

Example:

```
#Specify default tos = 0x10 (low delay)

TOS 0x10
```

6.4. LABEL

Script syntax:

```
LABEL <value>
```

The LABEL command specifies a default value to be used as the "flow label" for IPv6 flows. The "flow label" is the corresponding 28-bit field in the IPv6 packet header. Refer to the Transmission Event LABEL option for further details..

Example:

```
#Specify default IPv6 flow label = 0x02500000

LABEL 0x02500000
```

6.5. TTL

Script syntax:

```
TTL <value>
```

The TTL command specifies the default time-to-live (TTL) hop count for generated IP multicast traffic according to the <value> field. The <value> must be in the range of 1-255. If the global TTL command is not used, mgen assumes a default multicast TTL value of 1. Note that the transmission event TTL option will override the default specification given by this global command.

Example:

```
#Specify default multicast flow ttl = 32

TTL 32
```

6.6. UNICAST_TTL

Script syntax:

```
UNICAST_TTL <value>
```

The TTL command specifies the default unicast time-to-live (TTL) hop count for generated IP unicast traffic according to the <value> field. The <value> must be in the range of 1-255. If the global UNICAST_TTL command is not used, mgen assumes a default unicast TTL value of 255. Note that the transmission event TTL option will override the default specification given by this global command.

Example:

```
#Specify default unicast flow ttl = 32
```

```
UNICAST_TTL 32
```

6.7. TXBUFFER

Script syntax:

```
TXBUFFER <txBufferSize> ...
```

This option allows users to set the default socket transmit buffer size to a value at least as large as <txBufferSize>. The exact behavior of this option may be operating system dependent. The TXBUFFER option given on transmission event script lines will override this default for the socket used by the corresponding flow.

6.8. RXBUFFER

Script syntax:

```
RXBUFFER <rxBufferSize> ...
```

This option allows users to set the default socket receive buffer size to a value ?at least? as large as <rxBufferSize>. The exact behavior of this option may be operating system dependent. The RXBUFFER option given on transmission event script lines will override this default for the socket used by the corresponding flow.

6.9. LOCALTIME

Script syntax:

```
LOCALTIME
```

This option allows users to specify that events and error messages be logged in localtime rather than the default Greenwich Mean Time.

6.10. QUEUE

Script syntax:

```
QUEUE
```

This global command will cause mgen to buffer <queueSize> mgen packets for each flow during periods of congestion. (Note that flow specific limits specified at the transmission event level will override this global). When the number of pending messages for a flow exceeds this limit, the message transmission timer will be temporarily deactivated and any pending messages will transmitted as quickly as possible. The timer will be reactivated once the pending message count falls below the queue limit, and message transmission will return to the previously scheduled rate of transmission.

If no global command is specified, a default <queueSize> of "0" will be in effect which will result in no queuing behavior, e.g. the transmission timer will continue to fire at its regularly scheduled interval regardless of transport congestion. No pending message count will be accumulated and message transmission will suceed or fail depending on transport availability.

When multiple flows are sending messages over a common mgen transport, the flows that have exceeded their pending message limit(s) will be serviced in a round robin fashion until all pending messages have been sent for all flows. Transmissions for flows that have fallen below this threshold will be interleaved as scheduled.

A <queueSize> threshold of "-1" sets an unlimited queue size. This means that a congested flow's transmission timer will continue to fire (thereby building up it's pending message count), but any pending messages will be sent as quickly as possible until congestion clears.

Any pending messages for a flow will be sent before the flow will be shutdown by a scheduled OFF event. Likewise, the pending message queue for a flow that is being restarted will be cleared. Note however, if any of the content of the mgen message header is changed (src or dst addresses, etc.) the pending message count will be reset. All other flow attribute changes (rage, message size, payload content, ttl, etc.) will be effected immediately, including any pending messages.

6.11. DATA

Script Syntax:

```
DATA [<hex>,<hex>]
```

User defined message payload can be specified with the DATA command. The data should be a hexadecimal representation of the user data where each pair of characters corresponds to one byte of user data. The contents will be placed in every packet generated by the flow.

6.12. INTERFACE

Script syntax:

```
<interfaceName>
```

The INTERFACE command specifies a default IP network interface to use for multicast traffic generation and group membership. If no INTERFACE command is given, the default operating system behavior is observed. Note that the transmission event INTERFACE option or the JOIN reception event INTERFACE option will override the default specification given by this global command.

Example:

```
#Specify "eth1" as the default network interface

#for multicast transmission and group joins

INTERFACE eth1
```

6.13. INPUT

Script syntax:

```
INPUT <scriptFile>
```

The INPUT command cause MGEN to load and parse the given <scriptFile>. (Circular references are not detected by mgen and should be avoided_). This allows scripts to "include" other scripts. The parsing occurs in the order that the INPUT commands are encountered on the command-line and within the script files themselves.

Example:

```
#Load and parse the MGEN script file "script2.mgn"

INPUT script2.mgn
```

6.14. OUTPUT

Script syntax:

```
OUTPUT <logFile>
```

The OUTPUT command cause mgen to direct its log output to the indicated <logFile>. The last occurring OUTPUT command determines the log file to be used and the command-line takes precedence over any scripts provided as input to mgen. The file named by <logFile> will be overwritten if it already exists.

Example:

```
#Use the file "logFile.drc" for logging
```

```
OUTPUT logFile.drc
```

6.15. LOG

Script syntax:

```
LOG <logFile>
```

The LOG command cause mgen to direct its log output to the indicated <logFile>. The last occurring LOG command determines the log file to be used and the command-line takes precedence over any scripts provided as input to mgen. The file named by <logFile> will be appended if it already exists.

Example:

```
#Append the file "logFile.drc"
```

```
LOG logFile.drc
```

6.16. LOGDATA

Script syntax:

```
LOGDATA {on|off}}
```

Controls whether MGEN will log the optional data attribute field at MGEN receivers (including within MGEN binary log files). It does not affect whether MGEN senders send the requested data attribute. By default LOGDATA is set to ON.

Example:

```
#Don't log optional data attribute at receiver
```

```
LOGDATA off
```

6.17. SAVE

Script syntax:

```
SAVE <saveFile>
```

The SAVE command causes mgen to write a short MGEN script upon exit which includes current sequence number state for pending and active transmission flows as well as the current relative script offset time. If the <saveFile> created is given as an additional input script (with the *same* input script(s) given for the mgen instance which created <saveFile>), on a subsequent launch of mgen, mgen will return to the same state as it was when it previously exited.

The SAVE command can be used when it is desired to conduct separate runs of mgen, but preserve a continuous sequence number space across the multiple runs. An example script performing this function is given below. Note the same behavior can also be achieved via the mgen command-line using:

```
"mgen input <scriptFile> input <saveFile> save <saveFile> log <logFile>"
```

if an empty <saveFile> is provided on the first launch of mgen.

Example:

```
# This script executes another MGEN script, using
# "saveFile.mgn" for state recovery upon subsequent

# restarts after mgen is exited.

# (If "saveFile.mgn" is empty or non-existent upon

# initial startup, the "scriptFile.mgn" is run

# from the beginning.)

# (The "log" command is used to repeatedly append

# the "logFile.drc" file upon stop and restart)

INPUT scriptFile.mgn

INPUT saveFile.mgn

SAVE saveFile.mgn

LOG logFile.drc
```

6.18. SEED

Script syntax:

SEED <int>

Seeds the random number generator with the provided int value. If no seed is provided, the current time is used.

6.19. IPv6

Script syntax:

```
IPv6
```

Forces mgen to open sockets for IPv6 operation (i.e. AF_INET6 domain sockets) only. The default behavior for mgen is to open sockets with the domain based on environment and the type of IP addresses used in the script file used.

Example:

```
# Global IPv6 option
```

IPV6

6.20. IPv4

Script syntax:

```
IPv4
```

Forces mgen to open sockets for IPv4 operation (i.e. AF_INET domain sockets) only. The default behavior for mgen is to open sockets with the domain based on environment and the type of IP addresses used in the script file used.

Example:

```
# Gloval IPv4 option

IPv4
```

6.21. DF

Script syntax:

```
DF {ON|OFF}
```

Controls whether the DF fragmentation bit is set.

Example:

```
# Set DF fragmentation bit
```

```
DF on
```

6.22. ANALYTICS

Script syntax:

```
ANALYTICS
```

Enables build-in, per-flow analysis of received flows and outputs "REPORT" events to the mgen log file. The analysis (similar to NRL's TRPR) product includes:

1. average received goodput rate for the given REPORT "window" time
2. loss fraction measured for the given REPORT "window" time
3. average, minimum, and maximum latency for the given REPORT "window" time

For example,

```
mgen event "listen udp 5000" analytics window 10
```

would measure and log received message statistics once per 10 seconds for each flow detected.

The "window" command can be used to set the analysis measurement window. A window of about 1 second is the default value. The window SHOULD be set large enough to have multiple messages per window time for useful measurement. The format of the REPORT lines are similar to other MGEN events. For example, locally generated report events have a an offset value of 0.0:

```
01:17:01.983235 REPORT proto>UDP flow>3 src>127.0.0.1/63684 dst>127.0.0.1/5002 offset>0.000000 window>1.9
```

REPORT events from remote MGEN instances include a sent> field:

```
01:17:01.983235 REPORT proto>UDP flow>3 src>127.0.0.1/63684 dst>127.0.0.1/5002 sent>01:16.58.675309 offse
```

The first timestamp field minus the value of the "offset" field is when the reported measurement was computed (i.e. the time of the *end* of the given averaging window being reported.) The "offset" is useful when receiving report content from the remote MGEN senders and not a locally generated report. (Locally generated REPORT events the "offset" will be zero.) The "window" field indicates what averaging window was used for the report (e.g., it may be different for remote MGEN senders depending on how their "window" was set.

6.23. WINDOW

Script syntax:

```
WINDOW <secs>
```

Sets the report analysis measurement window. A window of about 1 second is the default value. The window should be set large enough to have multiple messages per window time for useful measurement. The "window" field in the report line, e.g.

```
01:17:01.983235 REPORT proto>UDP flow>3 src>127.0.0.1/63684 dst>127.0.0.1/5002 offset>0.000000 window>1.9
```

indicates what averaging window was used for the report (e.g., the window in reports transmitted by remote MGEN senders may be different than locally generated MGEN reports depending on how their "window" command was set")

6.24. REPORT

Script syntax:

```
REPORT
```

Enables analytics reporting in all transmitting flows, meaning each flow will include in their transmitted MGEN message payload content per-flow analytics reports. MGEN nodes that receive these "report" payloads will also log those as REPORT event in their log files in addition to the usual RECV events. Note that the receiving mgen must have the analytics global set in order to log the REPORT messages. Thought should be given to what reporting is required when considering the global REPORT option versus the per flow REPORT option as a single flow can report for all received flows and multiple reporting flows will be redundant (even though this reporting is "in-band" of scripted message generation.) The metrics reported as sufficiently quantized to enable compact reports and that enables reporting potentially on many received flows in a single transmitted MGEN message. See the flow REPORT option for more detailed information on this capability.

6.25. SUSPEND

Script syntax:

```
SUSPEND <flowId|range|all>
```

Suspends a local flow (i.e. as opposed to the control of remote flows that inserting this options into a flow event provides). The main use for this would be to suspend flows upon startup so that they may be activated later upon receipt of remote flow command "resume" or "reset" messages. For example:

```
mgen event "on 1 udp dst 127.0.0.1/5001 per [1 1024]" suspend 1
```

would create a flow with flowid "1" that is immediately suspended (e.g. not transmitting) upon startup. Then later, if a 'resume' or 'reset' remote flow command is received, the flow would be activated and begin transmitting upon command. They keyword "all" can also be used with the suspend command (but not with the corresponding flow event option) to take action on all flows. For example:

```
mgen event "on 1 udp dst 127.0.0.1/5001 per [1 1024]" event "on 2 udp dsdt 127.0.0.1/5002 per [1 1024]" s
```

would create two flows that are suspended upon MGEN startup. Note that the suspend global command can be send to an mgen instance.

6.26. RESUME

Script syntax:

```
SUSPEND <flowId|range|all>
```

Resumes a local flow (i.e. as opposed to the control of remote flows that inserting this options into a flow event provides). The main use for this would be to resume flows that have been previously suspended in a local mgen instance. For example:

```
mgen event "on 1 udp dst 127.0.0.1/5001 per [1 1024]" suspend 1
```

would create a flow with flowid "1" that is immediately suspended (e.g. not transmitting) upon startup. Then later, the global 'resume' command can be used to activate the suspended flow.

```
mgen instance mgen1 resume 1
```

They keyword "all" can also be used with the resume command (but not with the corresponding flow event option) to take action on all flows. For example:

```
mgen instance mgen1 resume all
```

would resume all flow(s) suspended in mgen instance mgen1

6.27. RESET

Script syntax:

```
RESET <flowId|range|all>
```

Resets a local flow (i.e. as opposed to the control of remote flows that inserting these options into a flow event provides). The main use for this would be to reset flows managed by local mgen instances. When a "reset" command is received, the sender flow message counter is reset and the flow will continue transmitting count messages. For example:

```
mgen instance mgen1 event "on 1 udp dst 127.0.0.1/5001 per [1 1024] count 10"
```

would create a flow with flowid "1" that is scheduled to send 10 messages upon startup. Then later, when a 'reset' remote flow command is received, the flow would be activated and begin retransmitting 10 messages. For example: flows. For example:

```
mgen instance mgen1 reset 1
```

would cause the counter for the flow to be reset and an additional 10 messages would be sent. If the counter has already expired the reset message will have no effect.

6.28. RETRY

Script syntax:

```
RETRY <count>[/<delay>]
```

Sets the default retry behavior for all flows. If the underlying TCP socket for the flow fails to connect, or disconnects, attempt to reconnect the socket <count> times after <delay> seconds. The default retry interval is 5 seconds. The transport will attempt to reconnect over the life of the flow if the retry count is set to -1. If multiple flows are sharing the transport a common retry pattern should be used by all the flows or the last attribute set "wins".

For example:

Attempt to connect every five seconds:

```
ON 1 TCP DST 127.0.0.1/5000 PERIODIC [1 1024] RETRY -1/5
```

Attempt to connect twice after initial failure every 3 seconds

```
ON 1 TCP DST 127.0.0.1/5000 PERIODIC [1 1024] RETRY 2/3
```

RECONNECT events are logged at each reconnect attempt:

```
17:36:12.762619 RECONNECT flow>1 srcPort>35859 dst>127.0.0.1/5000
17:36:17.763483 RECONNECT flow>1 srcPort>46864 dst>127.0.0.1/5000
17:36:22.764783 RECONNECT flow>1 srcPort>46866 dst>127.0.0.1/5000
```

7. MGEN Log File Format

The MGEN message format contains information to facilitate network performance measurements through post-analysis of MGEN log files. Some of the types of performance statistics which can be determined include:

Message Throughput	The time or arrival and size of received messages are logged by MGEN receivers. Network throughput can be assessed with this information.
Message Delivery Latency	MGEN messages contain a timestamp reflecting when they were sent and the time of reception is logged by

	MGEN receivers. Delivery latency statistics (jitter or absolute if clock synchronization (e.g. NTP) is possible) can be determined with this information.
Message Loss Rate	MGEN messages are sequence numbered. Message loss can be accounted for via analysis of logged sequence number information.
Message Re-ordering	The logged MGEN sequence number information can also be used to determine message re-ordering statistics.
Multicast JOIN/LEAVE Latency	The occurrence and time of JOIN/LEAVE events are logged by MGEN receivers. The JOIN latency can be determined by comparing the arrival time of the first message associated with a particular multicast group to the time the group was joined. The LEAVE latency can be determined by comparing the time of the _last_ packet arrival time to the time the receiver left that multicast group (Note that you need a packet sniffing program like tcpdump to see packets after you leave the group).

Many of the above performance measures and statistics can be measured and optionally graphed using the NRL trpr (trace plot real-time) program. This program can parse the MGEN log file format and tcpdump traces.

7.1. General Log Format

Each line of the MGEN text log file format corresponds to a unique event and follows the convention:<eventTime> <eventType> <event attributes ...>The <eventTime> field is in the form hrs:min:sec and represents the computer's system Greenwich Mean Time (GMT) at the time of the event.

The <eventType> field is one of the following:

RECV	Denotes the arrival of a received MGEN message.
RERR	Indicates an invalid MGEN message was received.
SEND	Denotes the transmission of an MGEN message.
JOIN	Marks a join to an IP multicast group.
LEAVE	Marks the departure from an IP multicast group.
LISTEN	Indicates when mgen began monitoring a specific port
IGNORE	Indicates when mgen ended monitoring of a specific port
ON	Indicates when mgen initiated a TCP connection to the indicated destination ip address and port or when a UDP flow began transmitting.
CONNECT	Indicates when an mgen TCP "client" or "sender" has established a TCP connection from the indicated source port or to the destination ip address and port.
RECONNECT	Indicates when an mgen TCP "client" as attempted to reestablished a TCP connection from the indicated source port or to the destination ip address and port.
ACCEPT	Indicates when an mgen TCP "server" or "listener" has accepted a TCP connection from the indicated source ip address and port to the destination port.
SHUTDOWN	Indicates when an mgen TCP connection was shutdown after a scheduled >OFF< event on the client side or when an active connection is shutdown by the server on the server side. When TCP completes the shutdown

	procedure, an OFF or a DISCONNECT event will be logged as appropriate.
DISCONNECT	Indicates when an mgen TCP connection was disconnected prior to a scheduled >OFF< event on either the client or server side. This event indicates a TCP error has occurred on the connection. (NOTE: this indication of an unscheduled TCP shutdown is currently reliable only on Linux systems.)
OFF	Indicates that an mgen flow was stopped by a scheduled >OFF< event on the client or by a server IGNORE event on the sever. (Only available when transmission logging is enabled on the client side).
START	Indicates when mgen started processing Transmission and Reception events.
STOP	Indicates when mgen stopped processing Transmission and Reception events.

Different event types will have different event attribute sets. The <event attribute> fields are explicitly labeled so that log file parsing programs can seek specific attributes of interest for given event types.

7.2. Log File RECV Events

The format of the RECV event log file line is:

```
<eventTime> RECV proto><protocol> flow><flowId> seq><sequenceNumber> src><addr> / <port> dst><addr>/<port> sent><txTime> size><bytes> [host><addr>/<port>] [gps><status>,<lat>,<long>,<alt>] [data><len>:<data>] [flags><flag>]
```

The <eventTime> corresponds to when the message was received. The <protocol> specifies the protocol (udp,tcp,sink).The <flowId>, <sequenceNumber>, and <txTime>, are from the payload of the MGEN message. The <txTime> is in the same format as the <eventTime> (i.e. <hr:min:sec> GMT)

The "dst" <addr>/<port> is from the message payload and corresponds to the destination address to which the source addressed the MGEN message.

The "src" <addr>/<port> is the source address determined from the corresponding `recvfrom()` call for UDP transport or the address to which the TCP connection was made. (An optional "host" address will be embedded in the payload by the MGEN message source and made available as an attribute of the logged RECV event in the future).

The message "size" in <bytes> is also from the payload, but for UDP transport, should also correspond to the UDP packet payload size. Note that TCP mgen messages can be larger than the maximum UDP message size of 8192 bytes and can be of unlimited size. Therefore, mgen breaks large TCP messages into mgen message "fragments" of a maximum size of 65535 and sets a flag on the mgen message to indicate that it is a TCP message "fragment". Message fragments are flagged with 0x01 to indicate that the message is not complete. The last fragment in a TCP message is flagged 0x02 to indicate "end of message".

For example, a TCP mgen message of size 66559 will be received and logged by the receiving node as two messages as follows:

00:33:36.427143 RECV proto>TCP flow>1 seq>1 src>10.0.0.1/35056 dst>10.0.0.2/5000 sent>00:36:11.377105 size>65535 gps>INVALID,999.000000,999.000000,-999 flags>0x01

00:33:36.427499 RECV proto>TCP flow>1 seq>1 src>10.0.0.1/35056 dst>10.0.0.1/5000 sent>00:36:11.380137 size>1024 gps>INVALID,999.000000,999.000000,-999 flags>0x02

Also note that a single SEND message will be logged by the transmitting node with a size corresponding to the TCP message size, e.g.:

00:29:51.396962 SEND proto>TCP flow>1 seq>1 srcPort>0 dst>10.0.0.2/5000 size>66559

The "host" <addr>/<port> corresponds to the MGEN message source's "perceived" default local address. Note that this may be different from the source address contained in the MGEN log file due to firewalls, Network Address Translation (NAT) devices, multi-homed sources, etc. The accuracy of this information depends upon the source host's configuration with regards to domain name service (DNS), etc. Note this field is optional and may not be present if this information is not valid (The current initial MGEN release does not yet support this option).

The "src", "dst", and "host" <addr> fields are dotted decimal IPv4 addresses or colon-delimited IPv6 addresses.

The "flags" field is discussed above.

The global positioning system (GPS) information is available when the MGEN message source is used in conjunction with the NRL gpsLogger program. This program monitors an attached GPS receiver for position information and "publishes" it in shared memory. When mgen is run and detects that it can "subscribe" to GPS position information, it places it in the MGEN message payload. Note that gpsLogger can also be used with a pulse-per-second (PPS) capable GPS receiver to provide accurate time synchronization for hosts running the MGEN toolset. This may be useful for mobile network test environments. The MGEN log file "gps" attribute has the following comma-delimited fields:

<status>	This indicates the validity of the GPS information and may be either "INVALID", "CURRENT", or "STALE".
<lat>	This is the GPS latitude in degrees. A negative value denotes South while a positive value denotes North.
<long>	This is the GPS longitude in degrees. A negative value denotes West while a positive value denotes East.
<alt>	This is the GPS altitude in meters.

The optional "data" attribute is present only if the received MGEN message contains optional user-defined payload. If present, the <len> indicates the length (in bytes) of the user-defined payload and the <data> following the colon character':' is a hexadecimal representation of the user data where each pair of characters corresponds to one byte of user data. Thus, the number of characters in the <data> field will be 2 * <len>. (The "data" option was supported in MGEN 3.x via the MGEN Payload Manager (mpmgr) tool and is not yet supported in MGEN 4.x. The documentation will be updated when this option is supported).

Example RECV event log lines:

```
22:59:52.312721 RECV proto><protocol> flow>1 seq>1 src>10.0.0.1/5000 dst>10.0.0.2/5002
sent>22:59:52.310324 size>1024

23:59:53.312721 RECV proto><protocol> flow>1 seq>2 src>10.0.0.1/5000 dst>10.0.0.2/5002
sent>22:59:52.310324 size>1024 host>10.0.0.1/5000 gps>CURRENT,35.123,79.234,57

23:59:53.312721 RECV proto><protocol> flow>1 seq>2 src>10.0.0.1/5000 dst>10.0.0.1/5002
sent>22:59:52.310324 size>1024 host>10.0.0.1/5000 gps>CURRENT,35.123,79.234,57 da-
ta>10:01a97b34458cff0021e8
```

7.3. Log File RERR Events

The format of the RERR (Receive Error) event log file line is:

```
<eventTime> RERR proto><protocol> type><errorType> src><addr>/<port>
```

The <eventTime> corresponds to when the message in error was received. The <errorType> is one of "none", "version", "checksum", or "dstaddr". An receive error of type "version" indicates the MGEN sender is using an mgen executable with an incompatible version number. The "checksum" error indicates the received message failed checksum validation, and the "dstaddr" error indicates an invalid or unsupported destination address type in the MGEN message received. The <src> attribute indicates the source address of the message in error.

7.4. Log File SEND Events

The format of the SEND event log file line is:

```
<eventTime> SEND proto><protocol> flow><flowId> seq><sequenceNumber> src><srcPort> dst><ad-
dr>/<port> size><bytes> [host><addr>/<port>]
```

The <eventTime> corresponds to when the message was sent, and it should precisely match the <txTime> logged by the machine the packet is sent to, if the packet is received correctly.

All the data items are the same as those used in the Log File RECV Events.

7.5. Log File JOIN Events

The format of the JOIN log file event line is:

<eventTime> JOIN group> <groupAddress> [src> <srcAddress>] [interface> <interfaceName>]

The <groupAddress> is the IP multicast group address which was joined. The format of this field is either a dotted decimal IPv4 address or a colon-delimited IPv6 address. The <interfaceName> is given only when the executed MGEN script used the INTERFACE option in the corresponding JOIN script event.

Example JOIN event log lines:

```
22:59:50:234757 JOIN group>224.1.2.3
```

```
22:59:51:129574 JOIN group>224.1.2.4 interface>eth1
```

```
22:59:51:129574 JOIN group>224.1.2.4 src>25.25.25.1 interface>eth1
```

7.6. Log File LEAVE Events

The format of log file LEAVE event lines is:

<eventTime> LEAVE group><groupAddress> [src> <srcAddress>] [interface><interfaceName>]

The <groupAddress> is the IP multicast group address which was left. The format of this field is either a dotted decimal IPv4 address or a colon-delimited IPv6 address. The <interfaceName> is given only when the executed MGEN script used the INTERFACE option in the corresponding LEAVE script event.

Example LEAVE event log lines:

```
22:59:59:234757 LEAVE group>224.1.2.3
```

```
22:59:59:753683 LEAVE group>224.1.2.4 interface>eth1
```

```
22:59:59:753683 LEAVE group>224.1.2.4 src>25.25.25.1 interface>eth1
```

7.7. Log File LISTEN Events

The format of the LISTEN event log file line is:

```
<eventTime> LISTEN proto><protocol> port><portNumber>
```

The <protocol> field corresponds to the transport protocol type being used. Supported protocols include "UDP" and "TCP". The <portNumber> field is the host port number to be monitored.

Example LISTEN event log lines:

```
22:59:48:834205 LISTEN proto>UDP port>5000
```

```
22:59:49:328039 LISTEN proto>UDP port>5001
```

7.8. Log File IGNORE Events

The format of the IGNORE event log file line is:

```
<eventTime> IGNORE proto><protocol> port><portNumber>
```

The <protocol> field corresponds to the transport protocol type which was being used. Supported protocols include "UDP" and "TCP". The <portNumber> field is the host port number to be no longer monitored.

Example IGNORE event log lines:

```
23:00:00:723467 IGNORE proto>UDP port>5000
```

```
23:01:00:235629 IGNORE proto>UDP port>5001
```

7.9. Log File ON Events

The format of the ON event log file line is:

```
<eventTime> ON flow><flowID> srcPort><srcPort> dst><dst>/<portNumber>
```

This event indicates that mgen has attempted to establish a TCP connection with the target destination address and port or a UDP flow has begun transmitting. It does not indicate that the connection has been successfully established, only that a connection has been attempted. The <flowID> field corresponds to the TCP flow ID of the connection. The <srcPort> is either the OS provided or user specified src port for the flow. The <dst> field corresponds to the destination of the TCP connection. The <portNumber> field is the destination port number of the TCP connection.

Example ON event log lines:

```
23:00:00:723467 ON flow>1 srcPort>4000 dst>10.0.0.1/5000
```

7.10. Log File CONNECT Events

The <protocol> field corresponds to the transport protocol type which was being used. Supported protocols include "UDP" and "TCP".

The format of the CONNECT event log file line is:

<eventTime> CONNECT flow><flowId> srcPort><srcPort> dst><dst>/<portNumber>

The <src> and <dst> fields correspond to the local source port and destination ip address/port of the TCP connection. The <flowID> is the transmitting flow id. If multiple flows are sharing the connection, CONNECT events will be logged for each flow.

Example CONNECT event log line:

```
23:00:00:723467 CONNECT flow>1 srcPort>4000 dst>10.0.0.2/5000
```

7.11. Log File ACCEPT Events

The <protocol> field corresponds to the transport protocol type which was being used. Supported protocols include "UDP" and "TCP".

The format of the ACCEPT event log file line is:

<eventTime> ACCEPT srcPort><srcAddr><srcPort> dstPort><dstPort>

The <src> and <dst> fields correspond to the source ip address and port and local receiving(dst) port of the TCP connection.

Example ACCEPT event log line (server):

23:00:00:723467 ACCEPT srcPort>4007 dst>10.0.0.2/5000

7.12. Log File SHUTDOWN Events

The format of the SHUTDOWN event log file line is:

<eventTime> SHUTDOWN flow><flowID> srcPort><srcPort> dst><dstAddr><dstPort> (client)

<eventTime> SHUTDOWN src><srcAddr><srcPort> dstPort><dstPort> (server)

This event indicates that a TCP connection with the dst indicated address and port was shutdown after a client OFF event or by the server after a server IGNORE event has been processed and client connections remain active on the dst port being listened to. In the TCP client's log file, the dst address and port reflect the address and port of the node the client was attempting to connect to. In the TCP server's log file, the src address and port reflects the address and port of the connecting node. The <flowID> field corresponds to the TCP flow ID of the connection (this field is only available in the client's event log). Note that if a TCP connection was prematurely terminated, no SHUTDOWN event will be logged.

Example Client SHUTDOWN event log line (client):

```
23:00:00:723467 SHUTDOWN flow><flowId> src>10.0.0.2/5000 dstPort>6000
```

Example Server SHUTDOWN event log line (server):

```
23:00:00:723467 SHUTDOWN src>10.0.0.1/5000 dst>6000
```

If multiple flows are sharing the same connection, the shutdown event will be logged only when the last flow has stopped sending to the socket pair. An "OFF" event will be logged for any connections ending while other flows are still transmitting, e.g.:

```
22:35:52.458531 OFF flow>1 srcPort>4000 dst>10.0.0.1/5000

22:35:52.458542 SEND proto>TCP flow>2 seq>3 srcPort>4000 dst>10.0.0.1/5000 size>8192

22:35:52.458598 SHUTDOWN flow>2 srcPort>4000 dst>10.0.0.1/5000

22:35:52.460419 OFF flow>2 srcPort>4000 dst>10.0.0.1/5000
```

7.13. Log File DISCONNECT Events

The format of the DISCONNECT event log file line is:

<eventTime> DISCONNECT flow><flowID> srcPort><srcPort> dst><dstAddr><dstPort> (client)

<eventTime> DISCONNECT src><srcAddr</<srcPort> dstPort><dstPort> (server)

This event indicates that a TCP connection with the indicated address and port has disconnected either because the connection could not be established in the first place or because the connection was prematurely terminated. In the TCP client's log file, the dst address and port reflect the address and port of the node the client was attempting to connect to. In the TCP server's log file, the src address and port reflects the address and port of the connecting node. The <flowID> field corresponds to the TCP flow ID of the connection (this field is only available in the client's event log). Note that if a TCP connection was not prematurely terminated, no DISCONNECT event will be logged. (NOTE: Indication of unscheduled TCP disconnection is not available relieably under windows at this time.)

Example Client DISCONNECT event log line:

```
23:00:00:723467 DISCONNECT flow>1 srcPort>4000 dst>10.0.0.2/5000
```

Example Server DISCONNECT event log line :

```
23:00:00:723467 DISCONNECT src>10.0.0.1/4000 dstPort>5000
```

7.14. Log File RECONNECT Events

The format of the RECONNECT event log file line is:

<eventTime> RECONNECT flow><flowID> srcPort><srcPort> dst><dstAddr><dstPort> (client)

This event indicates an attempt to reconnect a TCP connection with the indicated address and port that has disconnected either because the connection could not be established in the first place or because the connection was prematurely terminated. In the TCP client's log file, the dst address and port reflect the address and port of the node the client was attempting to connect or reconnect to. The <flowID> field corresponds to the TCP flow ID of the connection. Note that if multiple flows are sharing a TCP socket reconnect events will be logged for each flow.

Example Client RECONNECT event log line:

```
23:00:00:723467 DISCONNECT flow>1 srcPort>4000 dst>10.0.0.2/5000
```

7.15. Log File OFF Events

The format of the OFF event log file line is:

<eventTime> OFF flow> <flowID> srcPort><srcPort> dst><dstAddr>/<dstPort> (client)

<eventTime> OFF src><srcAddr><srcPort> dstPort><dstPort> (server)

This event indicates that a TCP connection with the indicated address and port has been successfully shutdown (e.g. disconnected) after a scheduled mgen OFF event. In the TCP client's log file, the dst address and port reflect the address and port of the node the client was connected to. In the TCP server's log file, the src address and port reflects the address and port of the connecting node. The <flowID> field corresponds to the TCP flow ID of the connection (this field is only available in the client's event log). Note that if a TCP connection was prematurely terminated by either the client or the server, a DISCONNECT event will be logged instead of an OFF event. (NOTE: Under Windows operating systems, the DISCONNECT event is not reliable and the OFF event may be logged for both planned and unplanned socket disconnects.)

Example Client OFF event log line (client):

```
23:00:00:723467 OFF flow>1 srcPort>4000 dst>10.0.0.2/5000
```

Example Server OFF event log line (server):

```
23:00:00:723467 OFF src>10.0.0.1/4000 dstPort>5000
```

Note that the server will only log a single OFF event if multiple tcp flows are being recevied over the socket pair, although the client will log OFF events for each flow.

7.16. Log File START and STOP Events

The format of the START and STOP event log file line is:

```
<eventTime> STARTor<eventTime>STOP
```

These log file lines indicate the time at which MGEN began and ended its operation. The "START" time corresponds to the relative time zero for any executed scripts. This "START" time is when the mgen program was executed unless the global START command was invoked. The "STOP" command corresponds to when the mgen program was halted.

7.17. Log File REPORT Events

The format of locally source REPORT events is (note the offset will be 0.0 and there is no "sent>" field):

```
01:17:01.983235 REPORT proto>UDP flow>3 src>127.0.0.1/63684 dst>127.0.0.1/5002 offset>0.000000 window>1.970563 rate>10.000000 kbps loss>0.000000 latency ave>0.000119 min>0.000109 max>0.000129
```

The format of REPORT events from remote nodes includes the "sent>" field:

```
01:17:01.983235 REPORT proto>UDP flow>3 src>127.0.0.1/63684 dst>127.0.0.1/5002 sent>01:16.58.675309 offset>0.023000 window>1.970563 rate>10.000000 kbps loss>0.000000 latency ave>0.000119 min>0.000109 max>0.000129
```

8. Binary Log File Format

At the beginning of binary log files, there is a plain text line to make it easy to tell what kind of file it is. It has the mgen version number, as well as the type of file ("binary_log"). This line is terminated with a line feed and a NULL ('\0') character. Following the NULL, the file contains a series of binary formatted records. There are several different types of records in the binary log file format. Each record consists of a number of fields. The first single-byte field indicates the record type. A record type of 0 is considered invalid. All multiple-byte fields are in standard network byte order (i.e. most significant byte first). Each record in the binary log file corresponds to a single unique Mgen event, just as each line in the text-based log file does. Each binary log file record contains the same information that every line of the text-format log file has. The text-format log file can actually be recreated from a binary log file using the "convert" command of mgen.

8.1. Binary Log File RECV Events

The format of the RECV event binary log file record is:

```
The format of the RECV event binary log file record is:

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    type = 1   |   protocol    |       eventRecordLength       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        eventTimeSeconds                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      eventTimeMicroseconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            srcPort            |  srcAddrType  |  srcAddrLen   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           srcAddr ...                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          messageSize          |    version    |    flags      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           mgenFlowId                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         sequenceNumber                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          txTimeSeconds                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        txTimeMicroseconds                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            dstPort            |  dstAddrType  |  dstAddrLen   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           dstAddr ...                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           hostPort            | hostAddrType  | hostAddrLen   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          hostAddr ...                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            latitude                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           longitude                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            altitude                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   gpsStatus   |  reserved   |           payloadLen           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          payload ...                         |
```

The <type> field contains the record type. The record type for RECV events is 1. For backwards compatibility, unknown record types are skipped by the binary to text log file conversion function. The <protocol> field indicates the protocol of the message UDP <1>, TCP <2>, or SINK <3>. <0> indicates an invalid protocol.

The <eventRecordLength> field contains the length of this record, starting with the next byte. Thus, it contains the length of the entire record, less what will have already been read when this two-byte number is obtained.

The <srcPort> field contains the port number that the message was sent from. The <srcAddrType> field indicates the type of the source. Possible types and values include:

INVALID_ADDRESS	0
IPv4	1
IPv6	2

The <srcAddrLen> field indicates the length in bytes of source address field <srcAddr> to follow. The length should be 0 (zero) for the INVALID_ADDRESS type, 4 for IPv4 addresses, and 16 for IPv6 addresses.

The <srcAddr> contains the address to which the MGEN message was sent from. The address is in network byte order.

The rest of the packet is just a copy of the message payload from the original packet. For details on these fields, please look here.

8.2. Binary Log File TCP Connection Events

The format of the TCP connection event binary log file records is:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    type = 1   |   protocol    |       eventRecordLength       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        eventTimeSeconds                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      eventTimeMicroseconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       srcPort/dstPort         |   addrType    |     addrLen   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        srcAddr/dstAddr ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       srcPort/DstPort         |           flowID              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      (flowID cont.)           |   addrLen     |    hostAddr   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  (host addr cont)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The <type> field contains the record type. The record type for TCP corresponds to the tcp event type (ON, CONNECT, DISCONNECT, RECONNECT, OFF). The <protocol> field indicates the protocol of the message TCP <2>. <0> indicates an invalid protocol.

The <eventRecordLength> field contains the length of this record, starting with the next byte. Thus, it contains the length of the entire record, less what will have already been read when this two-byte number is obtained.

The <flowID> field indicates whether the event is a client or server event. A flowID indicates that the event is a TCP client event. A <0> flowID indicates that the event is a TCP server event. The <port> field contains the port number that the message was sent to/from. The <addrType> field indicates the type of the address. Possible types and values include:

INVALID_ADDRESS	0
IPv4	1
IPv6	2

The <addrLen> field indicates the length in bytes of source address field <addr> to follow. The length should be 0 (zero) for the INVALID_ADDRESS type, 4 for IPv4 addresses, and 16 for IPv6 addresses.

The <addr> contains the address to which the MGEN message was sent to/from. The address is in network byte order.

The <flowID> indicates the flow id of the tcp connection on the server side only. Host address fields are included if available.

8.3. Binary Log File RERR Events

The format of the RERR (receive error) event binary log file record is:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    type = 1   |   protocol    |       eventRecordLength       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       eventTimeSeconds                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     eventTimeMicroseconds                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            srcPort            |  srcAddrType  |  srcAddrLen   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          srcAddr ...                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          errorType                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The <type> field contains the record type. The record type for RECV events is 1. For backwards compatibility, unknown record types are skipped by the binary to text log file conversion function. The <protocol> field indicates the protocol of the message UDP <1>, TCP <2>, or SINK <3>. <0> indicates an invalid protocol.

The <eventRecordLength> field contains the length of this record, starting with the next byte. Thus, it contains the length of the entire record, less what will have already been read when this two-byte number is obtained.

The <srcPort> field contains the port number that the message was sent from. The <srcAddrType> field indicates the type of the source Possible types and values include:

INVALID_ADDRESS	0
IPv4	1
IPv6	2

The <srcAddrLen> field indicates the length in bytes of source address field <srcAddr> to follow. The length should be 0 (zero) for the INVALID_ADDRESS type, 4 for IPv4 addresses, and 16 for IPv6 addresses.

The <srcAddr> contains the address to which the MGEN message was sent from. The address is in network byte order.

The <errorType> indicates the type of message error detected and is one of the following possible error type values:

No error	0

Version number mismatch	1
Checksum validation failure	2
Message length error	3
Destination address invalid	4

8.4. Binary Log File SEND Events

SEND events are only logged if transmit logging has been turned on with the "txlog" option. The format of the SEND event binary log file record is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    type = 2   |   protocol    |       eventRecordLength       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        TCP message size                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          messageSize          |    version    |    flags      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          mgenFlowId                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        sequenceNumber                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         txTimeSeconds                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       txTimeMicroseconds                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            dstPort            |  dstAddrType  |  dstAddrLen   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          dstAddr ...                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           hostPort            | hostAddrType  | hostAddrLen   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         hostAddr ...                          |
```

These fields are the same ones that are used in the RECV events. The <type> for SEND events is 2. The <protocol> field indicates the protocol of the message UDP <1>, TCP <2>, or SINK <3>. <0> indicates an invalid protocol. Immediately following the <eventRecordLength> is a copy of the original message payload, without the GPS data and payload. The event time is left out, since it is the same as the transmit time. TCP message size will only exist for TCP protocol send events.

8.5. Binary Log File LISTEN/IGNORE Events

LISTEN and IGNORE events both have the same format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = 3/4   |   protocol    |       eventRecordLength       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       eventTimeSeconds                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     eventTimeMicroseconds                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   protocol    |   reserved    |          portNumber           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

LISTEN events have a <type> of 3, while IGNORE events have a <type> of 4. The <protocol> field contains the transport protocol type to LISTEN for, while the <portNumber> field tells us what port number to LISTEN on. Possible types and values for the <protocol> include:

INVALID_PROTOCOL	0

UDP	1
TCP	2

8.6. Binary Log File JOIN/LEAVE Events

JOIN and LEAVE events also have an identical format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = 5/6   |   reserved    |       eventRecordLength       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        eventTimeSeconds                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      eventTimeMicroseconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           groupPort           | groupAddrType | groupAddrLen  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          groupAddr ...                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ifaceNameLen  |             asciiInterfaceName ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      asciiInterfaceName ...                   |]
```

JOIN events are of <type> 5, and LEAVE events have a <type> of 6. The <groupPort> is the port which will be used to JOIN/LEAVE the group. The <groupAddrType>, <groupAddrLen>, and <groupAddr> are the type, length, and raw value of the group address, similar to those of addresses in other record types. The ifaceNameLen field contains the length in bytes of the <asciiInterfaceName> that follows it.

8.7. Binary Log File START/STOP Events

START and STOP events both have the same format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = 7/8   |   reserved    |       eventRecordLength       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        eventTimeSeconds                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      eventTimeMicroseconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

START events have a <type> 7, while STOP events are of <type> 8. They contain only the time the event occurred.

9. MGEN Message Payload

Note that the Version 5.0 message format is slightly different than that of MGEN Version 4.0. The MGEN message payload is in the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          messageSize          |    version    |    flags      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          mgenFlowId                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        sequenceNumber                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         txTimeSeconds                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      txTimeMicroseconds                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            dstPort            |  dstAddrType  |  dstAddrLen   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          dstAddr ...                          |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           hostPort            | hostAddrType  | hostAddrLen   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         hostAddr ...                          |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            latitude                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           longitude                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            altitude                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   gpsStatus   |  reserved   |           payloadLen            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          payload ...                          |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          padding ...                          |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          checksum                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

All multiple-byte fields are in standard network byte order (i.e. most significant byte first).

The <messageSize> field indicates the total size (including the <messageSize>, <version>, <flags>, etc fields) of the MGEN message in bytes. The current UDP-only transport limits this to a maximum of 8192 bytes. In the future, larger message sizes will be supported and in conjunction with the <flags> field, very large messages will be supported as a concatenation of MGEN messages to support emulation of large file transfers, etc.

The <version> field is the MGEN message protocol version number. This will enable future versions of MGEN to b backwards compatible and prevent older versions of MGEN from attempting to parse packets in unknown format.

Currently a single <flags> value (CHECKSUM = 0x01) is defined. When this flag is set, it indicates the presence of the <checksum> field at the end of the MGEN message. It is expected that additional flags will be useful as MGEN adds support for transport types besides UDP.

The <mgenFlowId> contains the flow/thread identification value associated with the MGEN flow in the corresponding script which created the flow. Note that each flow identified from an MGEN source has its own sequence number space.

The <sequenceNumber> contains the 32-bit sequence number which is incremented with each message generated for an MGEN flow. This will wrap to zero when the maximum is reached.

The <txTimeSeconds> and <txTimeMicroseconds> fields are used to mark the time of transmission of the MGEN message. The time is the source computer's system time in Greenwich Mean Time (GMT).

The <dstPort> is the destination port number to which the MGEN message addressed by the source.

The <dstAddrType> field indicates the type of destination address encapsulated in following fields. Possible types and values include:

INVALID_ADDRESS	0
IPv4	1
IPv6	2

The <dstAddrLen> field indicates the length in bytes of the destination address field <dstAddr> to follow. The length should be 0 (zero) for the INVALID_ADDRESS type, 4 for IPv4 addresses, and 16 for IPv6 addresses.

The <dstAddr> contains the destination address to which the source addressed the MGEN message. The address is in network byte order.

Note that the following fields are optional and the MGEN message length my be truncated at any point after here. Any incomplete optional fields are considered invalid.

The <hostPort> and <hostAddr> (if present and valid) contain the MGEN message source's default local address. Note that this may be different from the source address contained in the MGEN log file due to firewalls, Network Address Translation (NAT) devices, multi-homed sources, etc.

The <hostPort> is the destination port number to which the MGEN message was addressed by the source.

The <hostAddrType> field indicates the type of destination address encapsulated in following fields. The possible values are the same as for the <dstAddrType> described above.

The <hostAddrLen> field indicates the length in bytes of the destination address field <hostAddr> to follow.

The <hostAddr> contains the source's perception of its local default network address. In mgen, this is determined by a system call to `gethostname()`, followed by a call to name resolution. This address may be incorrect if the host is not correctly configured or domain name service (DNS) is unavailable.

The <latitude>, <longitude>, and <altitude> fields contain values corresponding to GPS information for the MGEN source if it was available. The <latitude> and <longitude> fields are encoded as follows:

<fieldValue> = (unsigned long)((<actualValue>+180.0)*60000.0)

The <altitude> field is the direct representation of the altitude value available from the source's GPS system.

The <gpsStatus> indicates the validity of the GPS information which was encoded. Possible status types and values currently include:

INVALID_GPS	0
STALE	1
CURRENT	2

In addition to the <gpsStatus> field, actual values of 999.0 for latitude and longitude, and ?999 for altitude also correspond to invalid values.

The <payloadLen> field, when of non-zero value, indicates the presence of optional user-defined content in the MGEN message. The <payloadLen> value indicates the quantity (in bytes) of user-defined content which follows.

The <payload> field contains the user-defined content and is of length <payloadLen> bytes. Note that a short MGEN <messageSize> could truncate this field. If the MGEN user provides the optional user-defined content, it is up to the user to ensure that the generated MGEN messages are of sufficient size as not to truncate the <payload> content.

The <padding> portion of MGEN messages contain undefined data content.

The <checksum> field is optional and is present when the CHECKSUM (0x01) flag is set in the <flags> field. Note that corrupted messages may result in MGEN messages with the <flags> field itself corrupted, so it may be useful for MGEN implementations to have an option to validate checksums even when the CHECKSUM flag is not set if it is known that the sender is providing checksum content.

Note: The total size of the MGEN message is defined by the <messageSize> field. The optional fields may be truncated if the <messageSize> is small. The minimum MGEN message size will depend upon the IP address types being used. For example, the minimum allowed MGEN message size using IPv4 addresses with no optional fields is 28 bytes (i.e. for UDP, the UDP payload size would be 28 bytes). If GPS information is to be included without truncation, the minimum message size becomes 52 bytes with the inclusion of the <hostAddr> and GPS information. For IPv6 destination addresses, the minimum allowed MGEN message size is 40 bytes with no optional fields. If GPS information is included the minimum message size with truncating information is 76 bytes.

10. Compile options

10.1. RANDOM_FILL

Adding the RANDOM_FILL compile time option will cause MGEN to fill the payload with random content. Otherwise, the payload will be zero filled. (Alternatively, the DATA option may be used).

10.2. HAVE_IPV6

The HAVE_IPV6 compile time option indicates that the system is IPV6 capable. IPV6 packets will be generated.

10.3. SIMULATE

The SIMULATE compile time option indicates that MGEN will be running in a simulation environment.

10.4. HAVE_GPS

The HAVE_GPS compile time option indicates that a GPS service is available. GPS information will be put into the MGEN message payload.

10.5. HAVE_PCAP

The HAVE_PCAP compile time option indicates that the system has PCAP installed. This option should be used when MGEN will be "cloning" a TCPDUMP file.

10.6. Other

Operating system compile time options include UNIX, _WIN32_WCE, WIN32, LINUX, MACOSX. These should be set as appropriate to the operation system type. UNICODE log file output may be specified for WIN32 operating systems. On Solaris (and possibly other Unix) operating systems, the IP_MAX_MEMBERSHIPS option will set the IP_MAX_MEMBERSHIP limit per socket to -1 as no pre-defined limit is set in the OS.

11. Known issues

11.1. Macosx Windows TCP Interaction

There is a negative interaction between Nagle's TCP Algorithm and Delayed ACK that causes a TCP performance delay between traffic flowing from some Windows OSs to macosx. This can be alleviated somewhat by setting the sysctl net.inet.tcp.delayed_ack value to 1 (always employ delayed ack, 6 packets can get 1 ack) although performance may still be still be impacted. Windows-7 is impacted more than Vista. The phenomenon varies by traffic pattern.

11.2. Windows Buffer Sizes

As windows default buffer size 8192K it is recommended that adequate rx/tx buffers be set for UDP traffic via the TXBUFFER and RXBUFFER global or flow commands so UDP performance is not degraded.