# MaRS (Maryland Routing Simulator) - Version 1.0 User's Manual*

Cengiz Alaettinoğlu, Klaudia Dussa-Zieger, Ibrahim Matta
A. Udaya Shankar

Systems Design and Analysis Group
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

UMIACS-TR-91-80
CS-TR-2687

June 1, 1991

**Categories and Subject Descriptors:** C.4 [Performance of Systems]: *measurement techniques; modeling techniques; performance attributes;* C.2.2 [Computer-Communication Networks]: Network Protocols—*protocol architecture*; I.6.3 [Simulation and Modeling]: Applications.

*This work is supported in part by RADC and DARPA under contract F30602-90-C-0010 to UMIACS at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, RADC, or the U.S. Government.
Computer facilities were provided in part by NSF grant CCR-8811954.

# Contents

**1 Introduction** **1**

**2 Overview** **1**

**3 Obtaining MaRS** **5**

**4 Starting MaRS** **5**

**5 Components of MaRS** **6**
5.1 Link . . . 6
5.2 Node . . . 7
5.3 Routing . . . 9
5.3.1 SPF . . . 10
5.3.2 SEGALL . . . 11
5.3.3 ExBF . . . 11
5.4 Link-cost function . . . 11
5.5 Workload . . . 13
5.5.1 FTP_Source and FTP_Sink . . . 13
5.5.2 TELNET_Source and TELNET_Sink . . . 16
5.5.3 Simple_Traffic_Source and Simple_Traffic_Sink . . . 18
5.6 Performance monitor . . . 19
5.7 Stopper . . . 22

**6 User Interface** **22**
6.1 Control section . . . 22
6.2 Modifying the network . . . 26
6.3 Meter information windows . . . 27
6.3.1 Modifying a meter's parameter . . . 30
6.4 Manipulation of windows . . . 30

**7 Example** **30**

**A Customizing Colors, Fonts, and Keyclicks** **31**

# 1 Introduction

MaRS (Maryland Routing Simulator) is designed to provide a flexible platform for the evaluation and comparison of network routing algorithms. MaRS allows the user to define a network configuration, control its simulation, log the values of selected parameters, and save, load, and modify network configurations.

MaRS is a discrete-event simulator. In this manual, we use the term **target system** to refer to the network to be simulated. The target system is defined by a set of state variables and a set of events. Associated with each event is a routine that specifies how the state variables are updated when the event occurs. We use the term **simulated time** to refer to the time up to which the target system has been simulated.

This manual assumes background information about computer networks [Tanenbaum, 1988], X Window System [Jones, 1989], and UNIX [Ritchie, 1984].

**Acknowledgment** We gratefully acknowledge the use of some of the event handling and graphic interface routines of the MIT Network Simulator [Heybey, 1989]. Also, the user and programmer manuals of the MIT Network Simulator helped us in writing manuals for MaRS.

# 2 Overview

MaRS considers the target system as consisting of a physical network, a routing algorithm, and a workload, as illustrated in Figure 1. We next describe each of these in more detail. For full details, see sections 5 and 6.

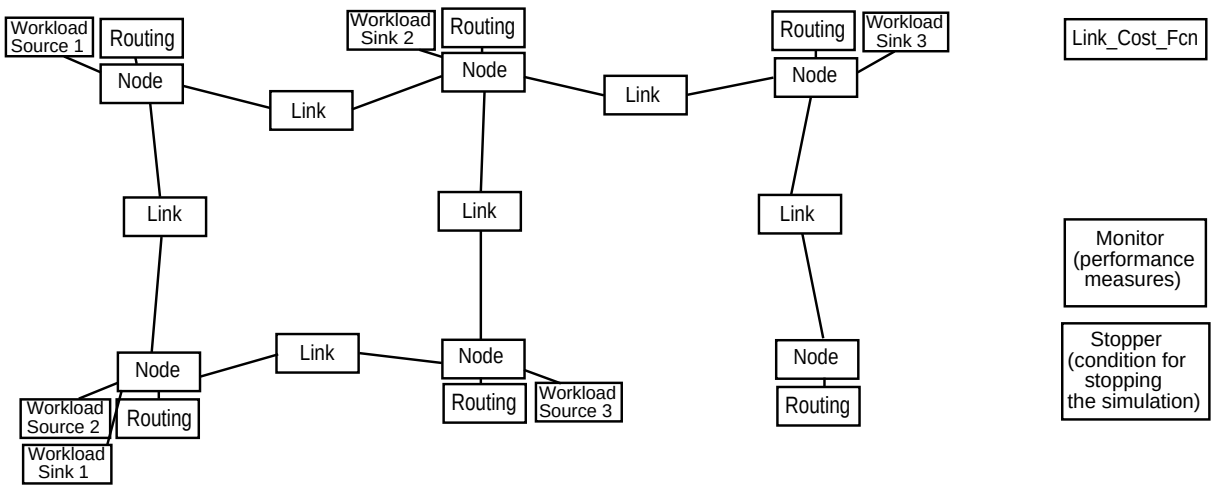


Figure 1: An example target system with performance monitor and stopper

The **physical network** consists of link components and node components. A **node component** models the "physical" aspects of a store-and-forward entity. In the current version, it is characterized by parameters such as buffer space, processing speed, packet queueing discipline, and failure and repair distributions. A **link component** models a transmission channel between two nodes. In the current version, a link component connecting node A and node B represents two one-way channels, one from node A to node B and one from node B to node A. Each one-way channel is modeled by a queue of packets. A link component is characterized by parameters such as bandwidth, propagation delay, and failure and repair distributions. By connecting link components and node components, the user can specify a network of arbitrary topology.

The **routing algorithm** maintains at each node routing information that allows the node to route packets to their destination. To realize a routing algorithm, a **routing component** is connected to every node component. The routing component contains the data structure representing the routing information (e.g. routing table) and the functions to update these data structures and propagate their information. In the current version, three types of routing algorithms are implemented: SPF (Shortest Path First) [McQuillan *et al.*, 1980], SEGALL [Merlin and Segall, 1979] and ExBF [Cheng *et al.*, 1989]. SPF is a link state routing algorithm, while SEGALL and ExBF are distance vector routing algorithms. Each algorithm uses next-hop routing. The user can choose any one of the three routing algorithms; once the choice is made, all the routing components have to be of that type.

In order to calculate paths, routing algorithms require a **link cost** to be associated with every link in the network. In the current version, four types of link cost functions are provided, based on hop count, utilization, delay, and hop-normalized delay. The user can choose one of the four functions by appropriately specifying a **link-cost component**. There is exactly one link-cost component, and it is not connected to any other component.

The **workload** is defined in terms of **source-sink** pairs. Each source-sink pair consists of a **source component** and a **sink component**. These components are connected to node components (normally, the source's node component is different from the sink's node component). The source component produces packets and passes them to its node component. The packets are then forwarded by node components and link components until they reach their destination node, where they are consumed by the sink component. A node component can have several source/sink components connected to it.

In the current version, the user can choose between three different types of source-sink pairs: file transfer (FTP), remote login (TELNET), and a simple workload (Simple_Traffic). In FTP and TELNET, the source produces data packets according to a packet-

train model [Jain and Routhier, 1986]. Intuitively, each train corresponds to a connection. These workload models incorporate a flow control mechanism, and an acknowledgment with retransmission mechanism using roundtrip time estimates (see section 5.5 for details). The simple workload does not use the packet-train model and has no acknowledgment or retransmission mechanism.

We distinguish between the packets generated by the routing components and by the workload components as follows. The term **routing packets** refers to packets that are generated by routing components. The term **workload packets** refers to packets generated by workload components. In the case of FTP and TELNET, workload packets can be further subdivided into **data packets**, **ack packets**, and **token packets**. Data packets carry data, ack packets carry acknowledgements to data, and token packets are used by the workload components for control purposes (e.g. roundtrip time estimate).

In addition to the above components for the target system, MaRS provides a performance monitor component and a stopper component.

The **performance monitor component** provides performance measures to evaluate and compare different routing algorithms. The performance measures in MaRS can be divided into two classes: periodically-updated and event-updated. **Periodically-updated** performance measures are updated periodically at time instants separated by a fixed length interval called the **update period**. **Event-updated** performance measures are updated whenever a related event occurs. This classification is concerned with how performance measures are computed in our simulator, and not with intrinsic properties of the measures (for example, a measure such as average delay per packet can be either periodically-updated or event-updated). In the current version, examples of periodically-updated performance measures include the average network throughput, the time variance of the instantaneous network throughput, the instantaneous routing load, and average delay per packet (see Section 5.6 for definitions of these measures). Examples of event-updated performance measures include the average number of packets in an FTP train, the number of currently active connections, and number of currently failed links.

**Instantaneous performance measures** are based only on statistics collected during the last update period. **Average performance measures** are based on statistics collected after a specified period of time, called the **startup interval**, from the beginning of the simulation. Ignoring the startup interval reduces the transient effect in the beginning of the simulation due to the fact that the network is initially empty.

The **stopper component** allows the user to define a termination condition for the simulation, based on the statistics collected by the performance monitor component during the simulation. Intuitively, the termination condition being true should indicate that the simulation has reached steady state for the performance measures of interest. In the current version, the stopper component is characterized by parameters such as a minimum number of elapsed intervals before termination is possible, length of the interval, and an error bound relating successively computed averages.

All the components mentioned above are implemented in MaRS Version 1.0. If these components are inadequate for some purpose, the user can define new components, e.g. a new workload component or a new routing component. Detailed information about how to program new components can be found in [Alaettinoğlu *et al.*, 1991].

Normally, the scheduling of events involves a random number generator. However, a special class of events, referred to as **record type events**, can be scheduled using either a random number generator or a trace file. In the current version, the record type events consist of failure and repair events for nodes and links, and connection_on events[1]. The need for a trace file arises when the same sequence of record type events has to be used in two different simulation runs, in order to have a valid comparison of the results.

MaRS can be invoked either with or without the X Window System. With the X Window System MaRS provides a graphical mouse-based user interface that is convenient for building network configurations and for stepping through simulations. Running MaRS without the X Window System increases the execution speed of the simulator, because there are no updates to the screen. This is the mode typically used to carry out a planned series of simulations. When MaRS is called without the X Window System, the user has to specify a networkfile containing the network configuration and the parameter values for all components. Currently, there is no software support for automatically converting a graph, generated by a network generator such as Cabri [Bordier, 1990], to a network format understood by MaRS.

Section 3 explains how to install MaRS. Section 4 explains the command line options of MaRS. Section 5 describes the different types of components. Section 6 gives detailed information about the MaRS user interface, including a description of how a network configuration is built. Section 7 shows a small example.

[1]The class of record type events is defined in the code of the simulator.

# 3 Obtaining MaRS

MaRS is public domain software that was developed at University of Maryland at College Park and is available via `anonymous` ftp from `ftp.cs.umd.edu` in the directory called `MaRS`. The file "MaRS/README" contains instructions on how to copy relevant MaRS files and release notes. In addition this file contains installation guidelines. No warranty is provided or implied with this software.

The MaRS developers can be contacted via electronic mail at `mars@cs.umd.edu`. The mailing list `mars-users@cs.umd.edu` is for users to discuss MaRS related topics, comments and experiences with MaRS.

Additions/deletions/updates to the mailing list `mars-users@cs.umd.edu` can be requested by sending mail to `mars-users-request@cs.umd.edu`

Send mail to the developers of MaRS if you have problems/questions. Please send us any changes/improvements to MaRS and its tools to include in future releases.

# 4 Starting MaRS

After MaRS is installed, it can be started by typing

```
mars [options] [networkfile [stoptime]]
```

where `networkfile`, if present, contains the description of the target system. The `stoptime` value, if present, is used to stop the simulation when the simulated time exceeds this value. In the current version, the options are as follows:

`-x`
: start the simulation of the target system immediately, without the X window system interface. This option should not be used if the networkfile is not present.

`-s seed`
: initialize the random number generator using the value of `seed`.

`-i`
: when the simulated time exceeds the stoptime value specified, the user is asked whether or not to continue the simulation.

`-r`
: use reverse video.

`-record`
: write the occurrence of record type events[2] to the file `sim_event.process_id`.

`-play filename`
: read and schedule events from the file `filename`, which should be a file that was created using the `-record` option.

`-perfdt update_period`
: for periodically-updated performance measures use the value of `update_period` as the update period length in microseconds.

`-skipt startup_interval`
: average performance measures are based on statistics collected after the simulated time exceeds `startup_interval` in microseconds.

The `-x` option is useful for long simulation runs; with this option the simulator runs faster. The `-record/-play` options allows the user to repeat a sequence of record type event occurrences across different simulation runs, for example, two simulation runs with different routing algorithms subject to the same failure/repair event occurrences. The `-perfdt` option can be used to control the size of the log file, since most of the performance measures are periodically-updated performance measures.

# 5 Components of MaRS

**Components** are the basic building blocks of the target system. The target system is composed by defining a number of components and interconnecting them. In the X Window System interface, a component is represented by a rectangular box containing the component's name. If two components are interconnected, this appears as a line connecting the two boxes. Every component has input parameters and output parameters. The value of an input parameter can be changed by the user. Most output parameters can be displayed by meters (described in section 6.3); they can also be logged.

We next describe the types of components defined in MaRS.

## 5.1 Link

A **link component** models a bidirectional transmission channel between two nodes. It contains two queues of packets, one for each direction. A packet in a queue can be viewed as

[2] In the current version, the record type events consist of failure and repair events for nodes and links, and connection_on events. See the MaRS programmer's manual for details [Alaettinoğlu *et al.*, 1991].

being in transit in the link. A link can fail and be repaired according to specified parameters.

Input parameters:

- Propagation delay [microseconds]
- Bandwidth [bytes/second]
- Distribution of interfailure times

  The interfailure time represents the time difference between starting the simulation and the occurrence of the first failure, as well as the time difference between the end of any repair time and the next failure occurrence. In the current version, the distribution can be chosen to be either exponential or uniform. If the exponential distribution is chosen, the mean of the distribution (in seconds) should be entered. If the uniform distribution is chosen, both the mean and the standard deviation (in seconds) of the distribution should be entered. (Note that a constant interfailure time can be specified by choosing a uniform distribution with a zero standard deviation.)

- Distribution of repair times

  The repair time represents the time taken to repair a link after its failure. The same distribution types are provided for the repair times as for the interfailure times.

Output parameters:

- Link status (up/down)
- Instantaneous workload utilization, for each direction

  The fraction of the link capacity that has been used for workload packets in the last update period, where link capacity is defined as the bandwidth multiplied by the update period.

- Instantaneous routing utilization, for each direction

  The fraction of the link capacity that has been used for routing packets in the last update period.

## 5.2 Node

A **node component** models the "physical" aspects of a store-and-forward entity such as a host or a switch. A node has a separate queue of packets for each outgoing link, and a single common queue for all routing packets received on its incoming links. The total size of all

queues cannot exceed a specified maximum number of bytes. In the outgoing link, routing packets have priority.

When a node receives a packet while the node is in a failed state, the packet is dropped. If the packet is a workload packet, it will be later retransmitted by the workload component that produced it.

When a node receives a workload packet while it is up, the node either passes the packet to a workload component connected to the node (i.e. the packet's destination), or appends it to the queue of an outgoing link (i.e. the next hop towards the packet's destination as specified by the routing component). If the next hop for the packet is not defined by the routing component, the packet is dropped. The packet may also be dropped because of buffer space limitation.

When a node receives a routing packet from the routing component connected to it and there is enough buffer space, the routing packet is inserted in the appropriate outgoing queue ahead of all workload packets in that queue (excluding the packet currently being transmitted, if any). If there is not enough buffer space, an attempt is first made to make room for the routing packet by dropping up to a predetermined number[3] of arbitrarily chosen workload packets, whose total size is greater than or equal to the size of the routing packet. In case of a successful attempt, the routing packet is inserted as described above. In case of an unsuccessful attempt, the routing packet is dropped.

When a node receives a routing packet on an incoming link and there is enough buffer space, the routing packet is appended to the single common routing packet queue in order to be processed by the routing component connected to the node. If there is not enough buffer space, the same policy described above in order to try making room for the routing packet is followed.

A node can fail and be repaired according to the parameters specified.

Input parameters:

- Delay to process a packet [microseconds]
- Total available buffer space [bytes]

  Bytes from this space are dynamically allocated to the various queues in the node on a FCFS basis. The user can specify an unbounded buffer space by entering -1 as the parameter's value.
- Distribution of interfailure times

[3]This predetermined number is currently set to 10 and can be changed. See the MaRS programmer's manual for details [Alaettinoğlu *et al.*, 1991].

The interfailure time represents the time difference between starting the simulation and the occurrence of the first failure, as well as the time difference between the end of any repair time and the next failure occurrence. In the current version, the distribution can be chosen to be either exponential or uniform. If the exponential distribution is chosen, the mean of the distribution (in seconds) should be entered. If the uniform distribution is chosen, both the mean and the standard deviation (in seconds) of the distribution should be entered. (Note that a constant repair time can be specified by choosing a uniform distribution with a zero standard deviation.)

- Distribution of repair times

  The repair time represents the time taken to repair a node after its failure. The same distribution types are provided for repair times as for interfailure times.

Output parameters:

- Node status (up/down)
- Amount of buffer space currently used [bytes]
- Maximum amount of occupied node buffer space so far in the simulation
- Number of workload and routing packets dropped by the node in the simulation
- Instantaneous packet drop rate

  The total number of workload and routing packets dropped by the node during the last update period divided by the length of the period.

- Current buffer utilization

  The currently used buffer space divided by the total available buffer space. It is equal to the currently used buffer space, if unbounded buffer space is specified by the user.

- Queue lengths [packets]

## 5.3 Routing

To realize a particular type of routing algorithm, a routing component of that type is connected to every node component of the target system. A routing component builds a "routing table", which specifies, for each destination, the next hop and the cost of the path towards that destination. This routing table is used by the attached node component to forward workload packets towards their destination. Currently, only next hop routing is used.

In the current version, we have implemented three types of routing components, corresponding to three routing algorithms: SPF, which is very much like the Arpanet routing algorithm [McQuillan *et al.*, 1980]; SEGALL [Merlin and Segall, 1979, Segal, 1981]; and ExBF [Cheng *et al.*, 1989].

SPF (Shortest Path First) is a link-state routing algorithm. In link-state algorithms, each node attempts to maintain a database describing the network topology and the link costs. Using this database, each node independently calculates shortest paths, and uses them to determine the next hop for any destination. Periodically, each node broadcasts the costs of all its outgoing links to all other nodes in the network.

SEGALL and ExBF are distance-vector algorithms. In distance-vector algorithms, each node periodically sends "summary" information to its neighbors regarding its current best cost to one or more destinations. Thus, each node can maintain for each destination the cost via each of its neighbors to that destination. For a given destination, the neighbor node through which the minimum cost is attained is chosen as the next hop.

In SEGALL type routing algorithm, the destination starts the next-hop-updates for that destination at all nodes that can reach it. We say that the destination has started a computation and we refer to the time between two successive computations as **intercomputation time**.

In ExBF type routing algorithm, each node checks if there is a change in the cost of its outgoing links. If so, the node starts a computation in which its costs to destinations are updated to reflect that change. Again we refer to the time between two successive computations as the intercomputation time.

We next describe the input and output parameters for each type of routing component. These components are not responsible for updating the link costs (mentioned above) of individual links; that is done by another component (the link-cost function component) described later in section 5.4. The description below assumes familiarity with the routing algorithms.

### 5.3.1 SPF

Input parameters:

- Mean and standard deviation of topology broadcast period [milliseconds]

  This period[4] is uniformly distributed.

Output parameters:

- Global topology table (adjacency matrix)

  Indicates the cost of each link in the network.

[4]Note that this period is different from the update period used to compute performance measures.

- Local topology table

  Indicates the cost and status of outgoing links.

- Routing table

### 5.3.2 SEGALL

Input parameters:

- Mean and standard deviation of intercomputation times [milliseconds]

  This intercomputation time is uniformly distributed.

Output parameters:

- Distance information table, for each destination

  Indicates the next hop and the cost via all neighbors, and other information needed by the routing algorithm.

- Local topology table

  Indicates the cost and status of outgoing links.

- Routing table

### 5.3.3 ExBF

The input and output parameters for an ExBF component are the same as for the SEGALL component above.

## 5.4 Link-cost function

Recall that routing algorithms assume that a cost is assigned to each link. The **link-cost function** is responsible for computing the link costs. This function is invoked at time instances $t_0, t_1, \ldots, t_{i-1}, t_i, \ldots$ by the routing component, where each $[t_{i-1}, t_i)$ is a **link-cost update period** defined by the routing component (usually, this is the same as the topology broadcast period or intercomputation period). The link-cost value is calculated at time $t_i$ by applying a transformation to a statistic referred to as **raw cost** which is observed on the link during the period $[t_{i-1}, t_i)$. The raw cost usually reflects the load on the link. The link cost calculated at time $t_i$ is used in the next interval, that is $[t_i, t_{i+1})$.

In the current version, the following raw link cost functions are supported:

- Hop-count

  1 for an operational link, $\infty$ otherwise.

- Utilization

  Fraction of the time that the outgoing link queue (including the packet being transmitted) was not empty in the last link-cost update period.

- Delay

  Represents the average delay suffered by a packet on the outgoing link queueing and in transmission (including the processing time at the node and the propagation delay of the link), over all packets transmitted in the last link-cost update period.

- Hop-normalized delay [Khanna and Zinky, 1989]

  A utilization calculated as follows. The average queueing delay per packet and the average transmission delay per packet are measured over the last link-cost update period. The utilization is calculated from these delays, assuming an $M/M/1$ queueing model.

The following transformation is applied to the raw cost (except in the case of hop-count), in order to reduce the degree of oscillations [Khanna and Zinky, 1989]:

< calculate and exponentially average raw cost (i.e. utilization, delay, or hop-normalized delay) >;
$NormalizedRawCost := max(min(AverageRawCost \times Slope + Offset,\ MaxCost),\ MinCost)$;
if $|NormalizedRawCost - Cost| < MovementLimit$ then
  $Cost := NormalizedRawCost$;
else
  $Cost := Cost + MovementLimit \times sign(NormalizedRawCost - Cost)$;

where the (1) $Slope$, $Offset$, $MaxCost$, $MinCost$ and $MovementLimit$ are parameters of the transformation, (2) $AverageRawCost$ and $Cost$ are variables updated at every link-cost update period, and (3) $NormalizedRawCost$ is a temporary variable.

Input parameters of this component are:

- Choice of the cost function

- Slope, Offset, MovementLimit, MaxCost, MinCost
  Used in the transformation.

- Max delay and min delay per packet observed on any link
  Used to calculate Slope in case the delay cost function is used (for convenience).

## 5.5 Workload

The **workload** is defined in terms of **source-sink** pairs. Each source-sink pair consists of a **source component** and a **sink component**. These components are connected to node components (normally, the source's node component is different from the sink's node component). The source component produces packets and passes them to its node component. The packets are then forwarded by node components and link components until they reach their destination node, where they are consumed by the sink component. A node component can have several source/sink components connected to it.

In the current version, the user can choose between three different types of source-sink pairs: file transfer (FTP), remote login (TELNET), and a simple workload (Simple_Traffic). In FTP and TELNET, the source produces data packets according to a packet-train model [Jain and Routhier, 1986]. Intuitively, each train corresponds to a connection. These workload models incorporate a flow control mechanism, and an acknowledgment with retransmission mechanism using roundtrip time estimates. The simple workload does not use the packet-train model and has no acknowledgment or retransmission mechanism.

We next describe the components for the three types of connections.

### 5.5.1 FTP_Source and FTP_Sink

The source and sink components for FTP are referred to as FTP_Source and FTP_Sink. They model a workload representation of file transfer. FTP_Source produces and sends data packets according to the input parameters. FTP_Sink consumes these packets and sends ack packets back to FTP_Source. The source and sink also exchange token packets. Token packets are used for roundtrip time estimates.

FTP provides a static window-based flow control scheme consisting of two windows: produce window and send window. The **produce window** size limits the number of data packets that have been produced but not yet sent into the network. If FTP_Source tries to produce a data packet and the produce window is full, FTP_Source will repeat its attempt after the specified delay between data packets. The **send window** size limits the number of data packets that have been sent but not yet acknowledged. If send window is full and FTP_Source wants to send a produced data packet, it will be delayed until the next ack packet is received.

In case of a workload packet loss (due to a link failure, node failure, or buffer space limitations), the FTP_Source or FTP_Sink component that produced the packet schedules a retransmission after a **time-out period**.

The time-out period is determined by the **roundtrip time estimate** which is calculated with the help of token packets. The roundtrip time of the $i$th token packet is the time difference between sending the packet and receiving it back. The roundtrip time estimate is

calculated by exponentially averaging the round trip time of the *i*th and *i-1*st token packet with a factor of 0.5.

Input parameters:

- Initial source-sink status (on/off)

  The status of the first connection between FTP_Source and FTP_Sink at the start of the simulation. The first connection can be on at the beginning of the simulation (i.e. start), or it can be off at the beginning and activated later after the specified delay between connections (i.e. trains).

- Number of connections between the specified FTP_Source and FTP_Sink

- Average number of packets per connection

  Geometrically distributed.

- Packet length [bytes]

- Average delay between connections [milliseconds]

  Exponentially distributed. This is the time difference between the production of the last packet of a connection and the production of the first packet of the next connection.

- Delay between packets [microseconds]

- Produce window size [packets]

- Send window size [packets]

Output parameters:

- Source-sink status (on/off)

  It displays the status of the current connection. The status of a connection is **on** if there are packets of the connection to be sent or to be acknowledged. The status of a connection is **off** if all packets of the connection have been acknowledged.

- Number of the current connection

  First connection is numbered one.

- Number of packets in the current connection

- Number of the current packet in the current connection

  First packet in a connection is numbered one.

- Number of packets produced

  Total number of data packets produced by the source since the beginning of the simulation.

- Number of packets sent

  Total number of data packets sent by the source since the beginning of the simulation.

- Number of packets acknowledged

  Total number of ack packets received by the source since the beginning of the simulation.

- Number of packets received

  Total number of data packets received by the sink since the beginning of the simulation.

- Number of packets retransmitted

  In the source component, it represents the total number of data and token packets retransmitted by the source since the beginning of the simulation. In the sink component, it represents the total number of ack packets retransmitted by the sink since the beginning of the simulation.

- Roundtrip time estimate [microseconds]

- Instantaneous sent rate

  The number of data bytes carried by data packets sent into the network by the source in the last update period divided by the length of the period.

- Instantaneous acked rate

  The number of data bytes carried by packets that have been acknowledged by ack packets at the source in the last update period divided by the length of the period.

- Instantaneous retransmission rate

  The number of data bytes carried by data packets that have been retransmitted by the source in the last update period divided by the length of the period.

- Instantaneous delay per packet [milliseconds]

  The average time difference between the first time the data packet is sent by the source into the network and the receipt of an ack packet at the source, averaged over all data packets acknowledged in the last update period. Note that this delay includes retransmission delays.

### 5.5.2 TELNET_Source and TELNET_Sink

The source and sink components for TELNET are referred to as TELNET_Source and TELNET_Sink. They model interactive traffic. TELNET_Source produces and sends data packets. TELNET_Sink consumes these packets and acknowledges them by sending ack packets. In addition TELNET_Sink also produces and sends data packets, which are consumed and acknowledged by TELNET_Source. Thus, both TELNET_Source and TELNET_Sink produce data packets and ack packets. This exchange of packets can be viewed as requests from TELNET_Source to TELNET_Sink, and responses from TELNET_Sink to TELNET_Source. To distinguish between data packets produced by TELNET_Source and TELNET_Sink, we use the term **response packet** to refer to data packets produced by TELNET_Sink[5].

Concerning the flow control scheme, retransmission of workload packets, and roundtrip time estimates, TELNET uses token packets and follows the same strategies as FTP.

Input parameters:

- Initial source-sink status (on/off)

  Defined as for FTP above.

- Number of connections between the specified TELNET_Source and TELNET_Sink

- Average number of packets per connection

  Geometrically distributed.

- Packet length [bytes]

- Response packet length [bytes]

  Geometrically distributed.

- Delay between connections [milliseconds], delay between packets [microseconds], delay to response [microseconds]

  All can be chosen to be either exponentially or uniformly distributed.

- Produce window size [packets]

- Send window size [packets]

Output parameters:

- Source-sink status (on/off)

  Defined as for FTP above.

[5]The term response packet is only introduced for convenience. It is not a new packet type.

- Number of current connection
- Number of packets in the current connection
- Number of the current packet in the current connection

  First packet in a connection is numbered one.
- Number of packets produced

  In the source component, it represents the total number of data packets produced by the source since the beginning of the simulation. In the sink component, it represents the total number of response packets produced by the sink since the beginning of the simulation.
- Number of packets sent

  In the source component, it represents the total number of data packets sent by the source since the beginning of the simulation. In the sink component, it represents the total number of response packets sent by the sink since the beginning of the simulation.
- Number of packets acknowledged

  In the source component, it represents the total number of ack packets received by the source since the beginning of the simulation. In the sink component, it represents the total number of ack packets received by the sink since the beginning of the simulation.
- Number of packets received

  In the source component, it represents the total number of response packets received by the source since the beginning of the simulation. In the sink component, it represents the total number of data packets received by the sink since the beginning of the simulation.
- Number of packets retransmitted

  In the source component, it represents the total number of data , ack, and token packets retransmitted by the source since the beginning of the simulation. In the sink component, it represents the total number of response, ack, and token packets retransmitted by the sink since the beginning of the simulation.
- Roundtrip time estimate [microseconds]
- Instantaneous sent rate

  Defined as for FTP above.

- Instantaneous acked rate

  Defined as for FTP above.

- Instantaneous retransmission rate

  Defined as for FTP above.

- Instantaneous delay per packet [milliseconds]

  Defined as for FTP above.

### 5.5.3 Simple_Traffic_Source and Simple_Traffic_Sink

The source and sink components for Simple_Traffic are referred to as Simple_Traffic_Source and Simple_Traffic_Sink. They model a very simple workload. Simple_Traffic_Source produces and sends data packets according to the input parameters. Simple_Traffic_Sink consumes these packets. There are no ack or token packets.

Simple_Traffic provides a static window-based flow control scheme using a single window. The window size determines the maximum number of produced, but not sent data packets. If Simple_Traffic_Source tries to produce a data packet and the window is full, it will repeat its attempt after the specified delay between data packets.

Any data packet that is received by the sink is considered to be acknowledged at the source. Note that the packet-train model is not used and there are no retransmissions.

Input parameters:

- Packet length [bytes]
- Distribution of the interpacket time

  Exponential or uniform.

- Window size [packets]

Output parameters:

- Number of packets sent

  Total number of data packets sent by the source since the beginning of the simulation.

- Number of packets acknowledged

  Total number of data packets received by the sink since the beginning of the simulation.

- Number of packets received

  Total number of data packets received by the sink since the beginning of the simulation. (Note that this equals the number of packets acknowledged.)

- Number of packets lost

  Total number of data packets lost since the beginning of the simulation due to link failure, node failure, or buffer space limitations.

## 5.6 Performance monitor

The performance monitor component collects and evaluates statistics about the network. The performance measures provided by MaRS can be classified into two categories: periodically-updated and event-updated. A periodically-updated performance measure is updated periodically at time instants separated by a fixed length interval called **update period**. An event-updated performance measure is updated whenever a related event occurs. For example, the total network throughput is updated periodically, while the current number of failed links in the network is updated whenever the status of a link changes from *up* to *down* or vice versa.

Average and instantaneous values of performance measures are calculated. An **average value** is computed based on statistics collected after a specified period of time, called the **startup interval**[6], from the beginning of the simulation. We thus call the period from the end of the startup interval to the end of the simulation the **measurement interval**. Ignoring the startup interval reduces the transient effect in the beginning of the simulation due to the fact that the network is initially empty. An **instantaneous value** is computed based only on statistics collected during the last update period.

Periodically-updated performance measures are: average network throughput, instantaneous network throughput, time variance of instantaneous network throughput, average delay per packet, instantaneous delay per packet, time variance of instantaneous delay per packet, number of dropped workload packets, number of routing packets used, average data load, instantaneous data load, time variance of instantaneous data load, average routing load, instantaneous routing load, and time variance of instantaneous routing load.

Event-updated performance measures are: current number of connections, average (and time variance of) number of packets in an FTP/TELNET connection, average (and time variance of) life-time of an FTP/TELNET connection, and maximum packet delay, and current number of failed links.

[6] Currently, the default length of the startup interval is 100 seconds and can be changed using the command line option -skipt.

An exception to the above classification is the buffer space statistics, which are updated only at the end of the simulation.

Output parameters:

- Average network throughput

  The total number of data bytes that have been acknowledged during the measurement interval divided by the length of the measurement interval.

- Instantaneous network throughput

  The total number of data bytes that have been acknowledged in the last update period divided by the length of the update period.

- Average delay per packet

  The **delay** of a data packet is the time difference between the first time the data packet is sent by the source into the network and the receipt of its corresponding ack packet at the source. The average delay per packet is the total delay of all data packets that have been acknowledged during the measurement interval divided by number of data packets acknowledged in this interval.

- Instantaneous delay per packet

  The total delay of all data packets that have been acknowledged in the last update period divided by number of data packets acknowledged in this period.

- Maximum packet delay

  The maximum of the delays of data packets that have been acknowledged during the measurement interval.

- Current number of connections in the network

  Number of connections that are currently on.

- Average number of packets per FTP connection

  The total number of data packets in all FTP trains that finished during the measurement interval divided by the number of such trains.

- Average life-time of an FTP connection

  The **active duration** of an FTP train is the time interval from producing the first packet in the train to receiving ack packets for all packets of the train at the source. The average life-time of an FTP connection is the total of the active durations of all

the FTP trains that finished during the measurement interval divided by the number of such trains.

- Average number of packets per TELNET connection

  Defined similar to the FTP one.

- Average life-time of a TELNET connection

  Defined similar to the FTP one.

- Current number of failed links

  Number of link components that currently have status *Down*.

- Total number of dropped workload packets

  Number of workload packets that are dropped in the simulation.

- Total number of routing packets

  Number of routing packets used in the simulation.

- Average data load

  The fraction of the network capacity (i.e. sum of all link capacities) used by workload packets during the measurement interval.

- Instantaneous data load

  The fraction of the network capacity used by workload packets in the last update period.

- Average routing load

  The fraction of the network capacity used by routing packets during the measurement interval.

- Instantaneous routing load

  The fraction of the network capacity used by routing packets in the last update period.

- Buffer usage statistics

  At each node the maximum buffer space used is maintained. At the end of the simulation, the maximum, the minimum and the average of those are provided.

### 5.7 Stopper

The stopper component enables the user to define a termination condition for the simulation, based on the input parameters and the statistics collected by performance monitor component. Intuitively, the simulation should be in "steady state" when the termination condition is satisfied. In the current version, we characterize the stopper component by three parameters: minimum number of elapsed intervals before termination is possible, length of the interval, and a percentage error bound relating successively computed statistics. We define the network throughput during a time interval to be the total number of data bytes acknowledged from the beginning of the interval to its end divided by the interval length. The termination condition is defined to hold when during a specified number of consecutive intervals of specified length, the throughput of the network in each interval stays within a specified percentage error bound relative to the average network throughput over all the intervals.

Input parameters:

- Number of intervals
- Interval length [microseconds]
- Error bound [percentage]

## 6 User Interface

MaRS has a graphic interface, which is almost entirely mouse-driven. The display is divided into three parts: the **control section**, the **network display**, and the **text window**. The control section is located on the right side of the screen. It has a clock indicating the simulated time, and the control buttons for operations such as starting the simulation, loading network configurations, etc. The network display is located on the left side of the screen. It displays the configuration of the network being simulated. The text window is a narrow window at the bottom of the screen used for text input and output. This text window appears only when it is being used.

### 6.1 Control section

#### Clock

The clock is located in the upper right hand corner of the screen. The clock has two parts. The first part is an analog clock indicating the simulated time. Each tick of the big hand represents 1 millisecond. Each tick of the small hand

represents one revolution of the big hand. A tick is a movement of 6 degrees around the circle. The purpose of the analog clock is not to display the exact amount of simulated time which has passed but rather to indicate how busy the simulator is. The second part is a digital clock, located below the analog clock. The digital clock displays the current simulated time. It is exact to the nearest microsecond.

**START**

Clicking the middle mouse button on the START control button will start the simulation with simulated time initialized to zero. The simulation can be started as many times as the user wishes.

**CONTINUOUS / EVENT STEP / TIME STEP**

When the user clicks the middle mouse button on the CONTINUOUS control button, the name in the button changes from "CONTINUOUS" to "EVENT STEP". Clicking the middle mouse button again changes the name from "EVENT STEP" to "TIME STEP". Another click of the middle mouse button returns the user to "CONTINUOUS", the normal operating mode. Both TIME STEP and EVENT STEP modes allow the user to step through events. In the EVENT STEP mode, each time the user clicks the middle mouse button on the clock, one of the events scheduled to occur at the current simulated time takes place. In the TIME STEP mode, each time the user clicks the middle mouse button on the clock, all the events scheduled to occur at the current simulated time take place.

**DELAY**

The DELAY control button allows the user to slow down the execution speed of the simulation by setting a real-time delay between the firing of each event. Note that this does not affect the simulation with respect to the simulated time. By default, this delay is zero. If the user wishes to have a delay between the firing of each event, he can click the middle mouse button on the DELAY control button. The text window will appear asking for the desired delay (in microseconds).

**UPDATE**

Clicking the middle mouse button on the UPDATE control button will toggle screen updating on and off. If screen updating is off, the simulator will not

update the screen, and hence the simulator will run faster. The clock, however, will still run.

**DSP: ALL / DSP: NODES / DSP: SORTED**

This control button allows the user to control the way the network is displayed. Clicking the middle mouse button on DSP: ALL changes the name in the button from "DSP: ALL" to "DSP: NODES". Clicking the middle mouse button again changes the name from "DSP: NODES" to "DSP: SORTED". Another click of the middle mouse button returns the user to the DSP: ALL display mode. When the display mode is DSP: ALL, the components of the network are displayed at the locations where the user has located them. When the display mode is DSP: NODES, only node and link components are displayed. When the display mode is DSP: SORTED, the node and link components are displayed with each routing component placed to the left of the node component attached to it, and other components (if any) placed to the right of the node component to which they are attached.

**KILL**

The KILL control button allows the user to destroy components. After the user clicks the middle mouse button on the KILL control button, each component that the user clicks on, with any of the mouse buttons, is destroyed. When the user is finished deleting components, clicking any of the mouse buttons on the QUIT control button (described below) ends this mode. Note that clicking on QUIT does not stop the simulator.

**LOAD**

The LOAD control button allows the user to load a network configuration. When the user clicks the middle mouse button on the LOAD control button, the text window appears asking for the name of the file to be loaded. Note that this erases whatever configuration was being displayed on the screen at that time (if any).

**SAVE**

The SAVE control button allows the user to save the present configuration. Clicking the middle mouse button on the SAVE control button causes the text window to appear asking for a filename under which to save the configuration.

### SNAP

The SNAP control button allows the user to do the same function as SAVE, in addition it saves the present arrangement of meters and information windows on the display. Neither SNAP nor SAVE saves the present values of the components' parameters.

### RECORD / STOP_RECORD

This control button allows the user to record event occurrences. Clicking the middle mouse button on the RECORD control button changes the name in the button from "RECORD" to "STOP_RECORD". Clicking the middle mouse button again changes the name from "STOP_RECORD" to "RECORD". The "RECORD" mode causes event occurrences to be recorded to the file sim_events.PID[7].

### PLAY

The PLAY control button enables the user to repeat event occurrences that have been previously recorded. When the middle mouse button is clicked on PLAY, the text window appears asking for the file name to be played. If the user is running MaRS with X window system, then he/she should click on the START control button to cause the event occurrences specified in this file to be scheduled.

### PRINT

Clicking the middle mouse button on the PRINT control button allows the user to print out the current network topology into a postscript file. The user must have the file basic_stuff.ps in his current directory.

### QUIT

Clicking the middle mouse button on the QUIT control button will stop the simulation and end the simulator. If the user has the line "mars.quitwarn: on" in his .Xdefaults file, the simulator will confirm the quit before ending the simulation.

[7]Currently, only failure, repair and connection_on occurrences are recorded.

## 6.2 Modifying the network

### Creating Components

To create a component, hold the shift key down and click the right mouse button on the network display. The initial location of the new component will be the location of the cursor when the right mouse button is released. After the right mouse button is released, a menu of component types will appear. After the user clicks any mouse button on the desired component type, a component information window will appear displaying the component's parameters and their default values. For most components, the user will be asked to enter values for some of the input parameters. Other input parameters will have their default values.

### Destroying Components

The KILL control button allows the user to destroy components (See the function of the KILL control button described above.)

### Opening/Closing a Component's Information Window

To display information about a component, click the middle mouse button on the component's box. A window will appear displaying each parameter of the component and its value. To close this information window click the middle mouse button on the component's box again.

### Connecting/Disconnecting Components

To connect and disconnect two components select a component by clicking the middle mouse button on it while the shift key is held down, then select the second component with the same method. The last two selected components are connected (are made neighbors) and a line connecting the two components will appear if they were not connected already. Otherwise they become disconnected and the line between them disappears. The last selected components stay selected till the user clicks the middle mouse button while the shift key is held down on something other than a component.

### Making Peers

To define a source-sink pair, after the workload source and sink components have been defined and connected to their respective nodes, the user must make them peers. To do this, click the left mouse button on each of the components while holding the shift key. The text window will confirm that the components are made peers.

### Modifying a Component's Input Parameter

To change the value of a component's input paramater, open the component's information window and click the middle mouse button on the parameter to be changed. The text window will appear, at which the new value of the parameter can be entered.

### Logging Data

Logging is a method of recording the values of a parameter. Logging for a parameter can be toggled on and off by clicking the middle mouse button on the right-hand box in the information window displaying that parameter. When logging for a particular parameter is turned on, its box in the information window changes its color, and every value of that parameter is saved in a log file. The log file is created in the current directory, and its name consists of two parts: the first part is "sim_log" and the second part is the process ID of the simulator.

### Creating a Meter

To display information about a parameter in a graphical form, a meter can be created for that parameter. To create a meter for a particular parameter of a component, click the middle mouse button on the left-hand box next to the parameter on the component's information window. This box will change color, and the meter will appear below the component's box. This meter will remain on the screen even if the component's information window is not displayed. Meters consist of rectangular boxes of resizable lengths and heights. There are different types of meters which display a parameter's values in different ways. To destroy a meter, click the middle mouse button on the left-hand box again. The meter will disappear, and the box will revert to its normal color.

## 6.3 Meter information windows

To display information about a meter, click the middle mouse button on the meter. This will cause a meter information window to appear. This window has the following information about the meter:

- Meter name: Each meter has a name. The default name is the first ten characters of the name of the parameter whose values the meter is displaying.
- Component name: This is the name of the parameter's component.

- Meter type: Each parameter has a default meter type, which is used when the meter is first created. The types are as follows:
  - BINARY METER: This meter is either completely black or white, depending on whether the parameter is off or on.
  - BAR GRAPH: This meter uses a wide horizontal line to display the parameter's value. The length of the line is dependent upon the paramter's value and is calculated in the following way. If the value of the parameter is between zero and one, the length of the line is equal to the value multiplied by the total length of the meter. If the value of the parameter is greater than one, the length of the line is calculated by multiplying the length of the meter by the value of the parameter and dividing by the value rounded up to the nearest integer. This has the effect of scaling the meter as the value continues to increase.
  - LOG: This meter is the same as the BAR GRAPH meter except that it uses the natural logarithm of the parameter's value instead of the value itself.
  - TIME HISTORY-A: This type of meter displays the values of the parameter versus time. Time is on the horizontal axis, and the values of the parameter are on the vertical axis. At certain intervals of simulated time, the average value of the parameter for that interval of time is computed and displayed as a line whose vertical height is proportional to this value. These lines are scaled in height in such a way that the largest value displayed on the meter, rounded up to the nearest integer, would have a line the same height as that of the meter. Each new line is displayed to the right of the previous one. When the rightmost edge of the meter is reached, all the lines are shifted to the left as a group about 1/4th the length of the meter. The sampling interval is about 30 milliseconds. This value can be changed if the user wishes as described below in section 6.3.1.
  - TIME HISTORY-D: This type of meter is exactly like the TIME HISTORY-A meter with the following exceptions: first, it updates the display each time its parameter changes, not at certain increments of simulated time. Second, it does not attempt to smooth out the values in any way, it simply displays the value of the parameter at the time the parameter is sampled.
  - DELTA METER: The DELTA meter is exactly like the TIME HISTORY-D meter with one exception. Instead of displaying the value of its parameter, it displays the difference between the present value of the parameter and the previous value of the parameter.
  - HISTOGRAM: This type of meter displays the frequency distribution of the parameter's values. The meter has four parameters. The first parameter is the

lower limit on tabulated values (Histogram Min). The second parameter is the upper limit on tabulated values (Histogram Max). The third parameter is the number of histogram cells (Histogram Cells). These three parameters specify the "interesting" range of values, (Histogram Min, Histogram Max), as well as the interval of each cell, which is equal to (Histogram Max - Histogram Min) divided by the number of Histogram Cells. If the sample value is out of the "interesting" range, the value is ignored. The fourth and last parameter for this type of meter is the number of retained histogram samples (Histogram Samples) for which the frequency distribution is displayed. Each of these four histogram parameters has a default value that can be changed by clicking the middle mouse button on the histogram parameter. A prompt will appear in the text window, and the new value can be entered.

- TEXT METER: This type of meter displays the value of the parameter in numeric or character format. It is mainly used to display routing tables, local topology tables, etc.

- Scale: Each meter has a scaling factor. It divides the values of the parameter by this scaling factor before displaying it.

- Sampling time increment: Some meters sample the value of their parameter at certain intervals of simulated time. The sampling time increment specifies this time interval in milliseconds.

- Display meter name: This attribute specifies whether or not the meter's name is displayed in the meter. Displaying the meter's name is toggled on and off by clicking the middle mouse button on this attribute.

- Display scale: At the moment, this attribute is not used. Eventually, it will specify whether or not lines indicating the meter's scale are displayed in the meter. Display scaling is toggled on and off by clicking the middle mouse button on this attribute.

- Histogram Min: This attribute specifies the lower limit on tabulated values for a meter of the HISTOGRAM type.

- Histogram Max: This attribute specifies the upper limit on tabulated values for a meter of the HISTOGRAM type.

- Histogram Cells: This attribute specifies the number of histogram cells for a meter of the HISTOGRAM type.

- Histogram Samples: This attribute specifies the number of retained histogram samples for a meter of the HISTOGRAM type.

### 6.3.1 Modifying a meter's parameter

To change the value of a meter's parameter, open the meter's information window and click the middle mouse button on the parameter to be changed. For Meter name, Scale, Sampling time increment, Histogram Min, Histogram Max, Histogram Cells, and Histogram Samples, the text window will appear, at which the new value of the parameter can be entered. For Display meter name and Display Scale, clicking on them will toggle the value of the parameter on and off. For the Meter type parameter, clicking on it will cause a menu window to appear displaying all types of meter. The user can then select the desired type by clicking on it with any of the mouse buttons.

## 6.4 Manipulation of windows

**Raising/Lowering Windows** Clicking the right mouse button on a window will raise that window so that no other windows on the display will obscure it. Clicking the left mouse button on a window will lower that window so that it does not obscure any other windows on the display.

**Moving Windows** It is possible to move any of the windows in the network display, including the control buttons (but only as a group with the clock), component boxes, component information windows, etc. To do this, click the right mouse button on the window to be moved, drag the window to the new location, and then release the right mouse button.

**Resizing Windows** It is possible to resize meter windows. To do this, click the middle mouse button on one corner of the window and resize it by moving the mouse. When the window is in the desired shape and size, release the middle mouse button.

The user can resize and move the entire simulator window using the window manager.

# 7 Example

Together with the MaRS code, user's manual and programmer's manual, a small networkfile called "example-net" is provided. The user can start MaRS with this networkfile in order to gain his/her first experience with the simulator. The networkfile can be used to load, modify, and save the network configuration, log the values of selected parameters, control the simulation, etc. as described above. After getting familiar with MaRS, the user should try to build his/her own network following the description in Section 6.

# A Customizing Colors, Fonts, and Keyclicks

The simulator has a full set of default colors and fonts for the screen, so the user does not need to specify any colors or fonts. If the user wishes, he can customize the colors and fonts to suit his own preferences. Each control button, each type of component, the clock, the meter-type menu, the text window, and the network display can each have their own color.

The user can also change the key bindings to be different from the ones presented in this manual. In most cases, it is best to simply use the default key bindings unless your window manager, mouse type, or other factor beyond your control prevents you from using them. The user can specify either the mouse button or the key state associated with each command, or both the mouse button and the key state. The user cannot specify whether the function is selected with a click or a click-drag-release. The possible mouse buttons are right, left, and middle. The possible key states are shift, control, meta, and none (for no key pressed). The commands for which the key/mouse clicks can be changed are selecting windows, moving windows, resizing windows, creating components, toggling neighboring between components, raising windows, and lowering windows. One thing to avoid is binding a key/button combination to a new function but not binding a new key/button to the old function. In that case, the key/button combination will only work for one of the functions, and the other function will be lost.

The user can specify all of these options in his X resource database information file. A sample with every possible option specified follows:

```
/* Colors for the control buttons */
mars.START:red
mars.CONTINUOUS:green
mars.DELAY:CadetBlue
mars.UPDATE:SeaGreen
mars.DISPLAY:gold
mars.KILL:yellow
mars.LOAD:OrangeRed
mars.SAVE:PaleGreen
mars.SNAP:thistle
mars.RECORD:MediumViolet
mars.PLAY:MediumViolet
mars.PRINT:MediumViolet
mars.QUIT:blue

/* Colors for different types of components */
mars.LINK:SkyBlue
```

```
mars.NODE:cyan
mars.SPF:orange
mars.SEGAL:orange
mars.EXBF:orange
mars.FTP_SOURCE:green
mars.FTP_SINK:green
mars.TELNET_SOURCE:green
mars.TELNET_SINK:green
mars.SMPLTRFC_SOURCE:green
mars.SMPLTRFC_SINK:green
mars.PERF_MONITOR:pink
mars.STOPPER:tan
mars.LINK_COST_FUNC:red

/* Colors for clock, the meter menu, and the network display */
mars.clockcolor:aquamarine
mars.editcolor:tan
mars.textcolor:yellow
mars.backgroundcolor:SkyBlue

/* Fonts */
mars.infofont:6x13
mars.controlfont:vg-31
mars.compfont:6x13
mars.editfont:9x15
mars.clockfont:9x15
mars.meterfont:6x10
mars.textfont:9x15

/* Warning message for quitting */
mars.quitwarn:off

/* Key bindings */
mars.select.button:middle
mars.select.key:none
mars.move.button:right
mars.move.key:none
mars.resize.button:middle
```

```
mars.resize.key:none
mars.create.button:right
mars.create.key:shift
mars.connection.button:middle
mars.connection.key:shift
mars.raise.button:right
mars.raise.key:none
mars.lower.button:left
mars.lower.key:none
mars.peer.button:left
mars.peer.key:shift
```

# References

[Alaettinoğlu *et al.*, 1990] C. Alaettinoğlu, K. Dussa, A. U. Shankar, and J. Bolot. Routing testbed: initial design. Technical Report UMIACS-TR-90-71, CS-TR-2475, Department of Computer Science, University of Maryland, College Park, MD 20742, May 1990.

[Alaettinoğlu *et al.*, 1991] C. Alaettinoğlu, K. Dussa-Zieger, and I. Matta. Mars programmer's manual. Department of Computer Science, University of Maryland, College Park, MD 20742. In preparation, expected June 1991.

[Bordier, 1990] J. Bordier. Cabri version 3.0. IMAG Laboratoire de Structures Discretes et de Didactique, April 1990.

[Cheng *et al.*, 1989] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves. A loop-free Bellman-Ford routing protocol without bouncing effect. In *ACM SIGCOMM '89*, pages 224–237, September 1989.

[Heybey, 1989] A. Heybey. The network simulator. Laboratory of Computer Science, Massachusetts Institute of Technology, October 1989.

[Jain and Routhier, 1986] R. Jain and S.A. Routhier. Packet trains - measurements and a new model for computer network traffic. *IEEE Journal on Selected Areas in Communication*, SAC-4(6):986–995, September 1986.

[Jones, 1989] O. Jones. *Introduction to the X Window System.* Prentice-Hall, 1989.

[Khanna and Zinky, 1989] A. Khanna and J. Zinky. A revised ARPANET routing metric. In *ACM SIGCOMM '89*, pages 45–56, September 1989.

[McQuillan *et al.*, 1980] J. M. McQuillan, I. Richer, and E. C. Rosen. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[Merlin and Segall, 1979] P. M. Merlin and A. Segall. A failsafe distributed routing protocol. *IEEE Transactions on Communications*, COM-27(9):1280–1287, September 1979.

[Ritchie, 1984] D.M. Ritchie. The evolution of the unix time-sharing system. *AT&T Bell Laboratories Technical Journal*, 63(8):1595–1605, October 1984.

[Segal, 1981] A. Segal. Advances in verifiable fail-safe routing procedures. *IEEE Transactions on Communications*, COM-29(4):491–497, April 1981.

[Tanenbaum, 1988] A. S. Tanenbaum. *Computer Networks.* Prentice-Hall, 1988.