Introducing MaRS, a Routing Testbed*

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

Department of Computer Science
University of Maryland
College Park, MD 20742

1 Introduction

We have recently developed a discrete-event simulator, referred to as MaRS (Maryland Routing Simulator), for studying routing systems. MaRS provides a flexible platform for the evaluation and comparison of network routing algorithms. It has an optional graphical (X Window System) interface. It is implemented in C on a UNIX platform. (The event handling and user interface routines come from a network simulator developed at MIT [5].)

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. The scheduling of events can be done using random number generators, trace files, or both. MaRS has a performance monitor that provides various average and instantaneous performance measures.

MaRS Version 1.0 can be obtained, with its user and programmer manuals, by anonymous ftp from directory `pub/MaRS` on `ftp.cs.umd.edu`. The MaRS developers can be contacted via electronic mail at `mars@cs.umd.edu`. A mailing list for users of MaRS is also provided, `mars-users@cs.umd.edu`. To subscribe to this list, send mail to `mars-users-request@cs.umd.edu`.

In Section 2, we describe briefly the structure of MaRS. In Section 3, we summarize some interesting results obtained using MaRS. For more information, readers are referred to references [1, 2, 3, 4].

2 Components of MaRS

The **network configuration**, i.e. the routing system being simulated consists of a physical network, a routing algorithm, and a workload.

The **physical network** consists of node and link components. A **node component** models the physical aspects of a store-and-forward entity, and is characterized by parameters such as buffer space, processing speed, packet queueing discipline, and failure and repair distributions. A **link component** models a bidirectional transmission channel between two nodes, and is characterized by parameters such as bandwidth, propagation delay, and failure and repair distributions. An operational link, i.e. one that is not in a failed state, behaves as an error-free FIFO channel. By connecting node and link components, a physical network of arbitrary topology can be specified.

The **routing algorithm** is specified by routing components (which are connected to node components) and a single link cost component. Each **routing component** maintains routing information, e.g. next hops, distances, link costs, etc. MaRS currently provides components

*This work is supported in part by RADC and DARPA under contract F30602-90-C-0010 to UMIACS at the University of Maryland, and by National Science Foundation Grant No. NCR 89-04590. 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.

†Also with Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

for three routing algorithms, SPF, Merlin-Segall, and Extended Bellman-Ford. The **link cost component** maintains and regularly updates the cost of each link. MaRS currently provides link cost functions based on hop count, utilization, delay, and hop-normalized delay.

The **workload** is defined in terms of **source-sink component pairs** which are attached to node components. The source produces packets destined for sink and passes them to its node component. The packets are then forwarded across the network and consumed by the sink. MaRS currently implements three types of source-sink pairs: file transfer (FTP), remote login (TELNET), and a simple workload (i.e. Poisson or uniform).

In FTP and TELNET, the source produces data packets according to a packet-train model; intuitively, each train corresponds to a connection. These workload models incorporate a static window-based flow control mechanism and an acknowledgment–with–retransmission mechanism using roundtrip time estimates. Retransmissions can be due to node or link failures or buffer space limitation. A workload pair is characterized by parameters such as inter-packet generation time, packet size, window size, etc.

In addition to the above components defining the network configuration, MaRS has a **performance monitor** component which provides various instantaneous and average performance measures, and a **stopper component** which allows the user to define a termination condition for the simulation.

3 Some Results obtained using MaRS

We have been using MaRS to compare two basic approaches to routing: namely **link–state** and **distance–vector**. We considered three algorithms: SPF, Merlin-Segall, and Extended Bellman-Ford, of which the last two are distance–vector algorithms. We considered the NSFNET-backbone network with today's hardware as well as a future high-speed hardware. We considered varying uniform and skewed workloads and failure distributions. Simulation results so far (explained in [4]) indicate that Extended Bellman-Ford is as good as SPF for good link cost functions. Furthermore, Extended Bellman-Ford is far less sensitive to changes in the link cost function. Merlin-Segall performs significantly worse than SPF and Extended Bellman-Ford except at very high and very low loads, and it is very sensitive to changes in link cost function [4].

References

[1] 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.

[2] C. Alaettinoğlu, K. Dussa-Zieger, I. Matta, and A. U. Shankar. MaRS User's Manual. Technical Report UMIACS-TR-91-80, CS-TR-2687, Department of Computer Science, University of Maryland, College Park, MD 20742, June 1991.

[3] C. Alaettinoğlu, K. Dussa-Zieger, I. Matta, Ó. Gudmundsson, and A. U. Shankar. MaRS Programmer's Manual. Technical Report UMIACS-TR-91-107, CS-TR-2723, Department of Computer Science, University of Maryland, College Park, MD 20742, July 1991.

[4] A. U. Shankar, C. Alaettinoğlu, I. Matta, and K. Dussa-Zieger. Performance Comparison of Routing Protocols Using MaRS: Distance-Vector versus Link-State. Technical Report UMIACS-TR-91-130, CS-TR-2762, Institute for Advanced Computer Studies and Department of Computer Science, University of Maryland, College Park, MD 20742, September 1991.

[5] A. Heybey. The Network Simulator. Laboratory of Computer Science, Massachusetts Institute of Technology, October 1989.