6-1995

DQDBsim User's Manual

Lewis Barnett III
University of Richmond, lbarnett@richmond.edu

DQDBsim User's Manual

Lewis Barnett
University of Richmond
Department of Mathematics and Computer Science
TR-95-01, June 1995

v0.2b

Contents

Chapter 1

Introduction

DQDBsim is a simulation program for the study of the performance of the Distributed Queue Dual Bus (DQDB) Metropolitan Area Network. The simulation is based on the description of DQDB found in [Ins90]. Only the Queued Arbitrated (QA) service is modeled by DQDBsim; the inclusion of Prearbitrated (PA) service is under consideration for inclusion in a future release. This document contains a description of the program, instructions for its use, details of the formats of input and data files, and information on other programs that are used cooperatively with DQDBsim.

1.1 Purpose

DQDBsim is intended as a tool for investigating the effects of a wide range of parameters on the performance of Media Access Control layer protocols for the DQDB Metropolitan Area Network. DQDBsim allows the user to investigate the effects of the message arrival process, the size of messages transmitted, the speed of the network, the layout of the network and the sequencing of transmission initiation among hosts on performance. The number of hosts, their positions on the network, and the overall length of the network are user configurable. The simulator provides results on a number of different performance metrics, such as overall throughput, average message delay, variance of message delay, and number and distribution of collisions. Many of these statistics are also tracked for each simulated station. In particular, the simulator tracks network delay from the transmitting station's perspective rather than only keeping track of message interarrival time from the network's perspective. Much of the concern surrounding the performance of the DQDB protocol has been focused on the fairness of the protocol and the effects of the Bandwidth Balancing mechanism which was added to the protocol to address this issue. DQDBsim models this aspect of the protocol and also supports data gathering modes which facilitate the investigation of the time dependent behavior of the protocol in addition to the average performance under a given traffic load.

1.2 Structure

The DQDBsim simulator is a single process discrete event simulator. The program simulates a finite population network where the number and position of nodes is user configurable. The program does not model message buffering at individual stations; each Distributed Queue State Machine (DQSM) at a station is assumed to generate a new arrival only after its previous message has been successfully transmitted. (Each station can have up to six DQSMs running concurrently; one for each of three priority levels on each of the two buses that each station is connected to.) Currently, only the Queued Arbitrated (QA) service is included in the simulation.

The parameters for a given execution of DQDBsim are read in from an experiment description file (described in Section 3). The file consists of a list of keywords and values. The values are used to initialize the simulator's variables. When the file has been completely parsed, one or more preliminary experiments are run to determine the offered load for the experiment, if appropriate. This is accomplished by running an experiment with one station generating traffic according to the default station parameters for arrival rate and message lengths. If all stations specified in the experiment description file follow the defaults, the offered load is the load generated by a single station multiplied by the number of stations specified for the experiment. If some stations have individually specified traffic processes, further offered load experiments will be run to determine the contribution to the overall load of such stations.

1.3 Features

1.3.1 Flexible Output Process Specification

The output process or sequence of intermessage times, message lengths, and recipients produced by each simulated station can be specified in several ways. These quantities can be generated randomly from probability distributions such as the exponential and uniform distribution. It is also possible to specify a custom tailored discrete distribution by specifying a set of probability/message length or probability/intermessage time pairs from which lengths or times can be randomly generated. These quantities can also be deterministic, i.e. can always be the same value. Finally, in the case of intermessage times, it is possible to simulate a continuously queued station. This type of station is always ready to transmit the next message immediately upon completing transmission of the current one.

All stations participating in a simulation can use the same distributions for these two quantities, or individual hosts can be configured differently. For example, it would be possible to configure one station with distributions which simulate the behavior of a file server while the other stations were configured to simulate the behavior of clients. In addition, it is possible to assign different distributions to each priority level for each bus in a given station. (In terms of the 802.6 standard, this means that each Distributed

Queue State Machine or DQSM has its own output process description.) This flexibility allows a wide variety of network traffic loads to be recreated.

1.3.2 User Oriented Delay Statistics

The simulator keeps track of the queueing delay experienced by transmitted messages. This is defined as the difference between the time the message was originally submitted for transmission and the time at which it completes successful transmission. This metric includes the time during which the message was "queued" at the station waiting for access to the channel and thus illuminates the delay characteristics of the protocol being simulated. Since messages may be broken down into segments to accommodate the fixed slot size of DQDB, this delay information is also recorded on a per-segment basis for some types of experiments. These quantities are more useful in determining the network response time from the point of view of the user than the network-oriented metric of overall interarrival time.

1.3.3 Automatic Offered Load Calculation

DQDBsim is able to automatically calculate the offered load for an experiment from the descriptions of the individual stations to be simulated. This is accomplished by automatically conducting several preliminary experiments, one for the default station distributions and one each for any hosts which use a different output process than the default. In these preliminary experiments, a single host transmits on an otherwise idle network with the default (or individual station) output process to determine the maximum offered load with the given output process.

1.3.4 Automatic Experiment Repetition

It is often desirable to repeat a simulation experiment several times starting with different random number generator seeds to diminish any effects of any particular sequence of events, then to use the averages of the collected statistics for data analysis. DQDBsim automatically performs this function for the experimenter for types of experiments where it is appropriate.

1.3.5 Varying Parameters

Simulation studies usually involve simulating the behavior of a system a number of different times while varying a single parameter to determine the effect that the parameter has on the performance of the system. DQDBsim is able to run such a sequence of experiments without human intervention by the use of the *varying parameter* mechanism. The parameter to vary and the way in which to vary it are specified in the experiment

description. The variation can be either by fixed sized steps up to a specified maximum value, or can be specified as an explicit sequence of values for the varying parameter.

1.4 Experiment Styles

DQDBsim supports three styles of experiment: summary experiments, interval experiments, and trace experiments. Summary experiments produce one report per varying parameter level which indicates the average performance of the protocol for that parameter set. Interval experiments divide a run up into a series of fixed length intervals and report statistics on the protocol's performance at the end of each interval. In combination with the capability to specify a "fuse" for each node (a length of time before the node will begin attempting to transmit at the beginning of an experiment) this allows the investigation of the effects of new stations becoming active on an already busy network. This type of information is important in the evaluation of the fairness of the protocol. Trace experiments keep track of the individual events generated by each DQSM as specified in the IEEE 802.6 standard document.

1.5 Availability

DQDBsim can be retrieved by anonymous FTP from `urvax.urich.edu` in the `pub` directory. More information on how to download the package can be found via the World Wide Web at `http://www.urich.edu/~barnett/dqdbsim/`. The program is known to compile on the following systems:

Hardware Platform	**Operating System**
Sparc 1+	SunOS 4.1.3
DECstation 3100	Ultrix 4.3
DECpc XL560 (Pentium 60)	Solaris x86 2.4

Chapter 2

Using DQDBsim

The command line interface to DQDBsim is very simple. Options on the command line have been kept to a minimum in favor of storing as much information as possible about the actual network being simulated in the experiment description file where it can serve as documentation on the parameters of the experiment. Therefore, the options to the DQDBsim command deal with input/output and with restarting experiments that are interrupted before completion. The formal specification for the DQDBsim command is as follows:

`dqdbsim [-i` *input_file*`] [-o` *output_file*`] [-r` *expt_num*`] [(+|-)d` *debug_key*`] [-v] [-C]`

Square brackets indicate that the enclosed text is optional. The `-i` option allows the user to specify a filename from which the experiment description should be read. If this option is not present, DQDBsim takes input from the standard input. Similarly, the `-o` option specifies a file for DQDBsim's output. If this option is not present, output goes to the standard output. The `-v` option specifies verbose output. When this flag is present, the program produces status messages about the progress of the experiment. The `-C` option prints out the copyright information for the package and exits.

The `-r` option allows a summary experiment which was interrupted to be restarted. The varying parameter mechanism allows an entire sequence of experiments to be specified by a single experiment description file. Each of these experiments writes out an output file of summary data as it completes, so experiments which are interrupted by system failures or other unforeseen circumstances need not be started again from scratch; rather, they can be restarted from the point at which they were interrupted. Each experiment in the sequence is given an index number, which becomes part of the file name of the summary file generated by the experiment. The original value of the varying parameter has index 0, the first incremented value has index 1, and so on. It is possible to determine the index of the last completed experiment by looking at the filenames of the experiment summary files, which will have the extensions ".nav" (node average summary), ".bav" (bus average summary), or ".pav" (priority average summary), depending

on the report types requested in the experiment description. *Expt_num*, specified with the `-r` option, is the index of the next varying parameter step that should be executed.

If debugging support has been compiled in, the `+d` option can be used to request that several different type of debugging activity be carried out during a run. The `-d` option can be used to suppress specific debugging information. Multiple instances of this option can appear on a command line to specify that combinations of the available debugging modes be used simultaneously. This will be primarily of interest to users who are interested in modifying the program. It should be noted that many of these options slow the simulation process considerably or generate huge volumes of output. The following values can appear after the `(+|-)d` option:

`DUMPEXPT`: Print out the experiment description after it has been completely read in. Useful for checking that the file was parsed correctly.

`DUMPNET`: Print out the description of the nodes on the network generated by the initialization process.

`DUMPQTX`: Print out the event queue after each transmission event.

`DUMPQEV`: Print out the event queue after each dequeue operation.

`DUMPNLOCS`: Print out the node locations assigned during the initialization process.

`DUMP_SPLIT_DIST`: Since the output of a node at a given priority is actually divided between the DQSMs which manage interaction with the two buses, the output distributions specified in the description file must be "split" into two distributions, one for each DQSM. Further, the way the distributions are split (at least for intermessage time distribution and recipient distributions) will vary from node to node. This value requests that the resulting distributions be printed out.

`CLK_CONSISTENCY`: Sanity checking; prints an error message if an event is ever dequeued with a time earlier than events that have already been processed.

`RNG_CHECK`: Print out some information about the generation of random numbers.

`Q_CONSISTENCY`: Check for reasonable event queue contents.

`CHK_DQSM`: Trace the activities of a particular DQSM. The DQSM traced must be specified at compile time using the defined constants `DEBUG_NODE`, `DEBUG_BUS`, and `DEBUG_PRI`.

`DEBUGALL`: Turns on all debugging flags. Must be specified with the `+d` option.

`DEBUGOFF`: Turns all debugging flags off. Must be specified with the `-d` option. Useful if specific debugging operations have been compiled in as the default.

Chapter 3

The Experiment Description File

The command line options to DQDBsim are simple; the data used to determine the nature of a simulation are stored in an *experiment description file*. This file contains all of the information necessary to carry out a simulation, including the layout and operating characteristics of the network to be simulated, the string used as the basis for the names of the output files, etc. The features of DQDBsim, including the automatically varying parameter facility, are best understood from the explanations of the various entries in the description file.

3.1 General Format

The experiment description file (a file whose name typically ends in ".edf") is loosely patterned on the structure of X Window System resource files [SG86]. Each entry in the file consists of a keyword and a value separated by a colon and at least one space or tab. (NOTE: It is **very** important that the colon be followed by at least one space or tab.) This structure has several advantages. The presence within the description file of a label for each entry means the entries are self-identifying, so a rigid arrangement or ordering is not necessary. There are a few cases where DQDBsim expects to see certain entries in a certain order; these are clearly explained in the next section. Further, if the names are chosen carefully, this structure allows the description file to act as documentation for the experiment. An effort has been made to choose meaningful names for the entries in the description file for this reason. This structure also makes future modifications easier; the addition or deletion of keywords can be easily handled and will not introduce incompatibilities between data files defined for different versions of the program.

The experiment description file uses the C Shell convention for indicating comments. Lines that begin with "`#`" are considered to be comments and are ignored by the program. Likewise, any characters on a line that follow a "`#`" are ignored. Blank lines are not significant, and are also ignored.

Message length, interarrival times, and recipient lists are generated from distribu-

tions specified in the description file. Each distribution is given a name in addition to the values that define its behavior. Once a list of such distributions is specified, these names are used to choose a default distribution for message length, interarrival time, and message recipients. It is also possible to configure individual stations with distributions other than the defaults so that computational activities such as file service can be modeled.

3.2 Explanation of Entries

The name of each entry is followed with an indication of whether the entry is required to appear in all experiment description files or is optional. In situations where it is possible to use alternative units in specifying the value associated with a keyword, the choices and default units are indicated. When a unit specifier is supplied with a value, there should be a space between the value and the unit specifier. A sample experiment description file is shown in Appendix A.

3.2.1 Physical characteristics of the network

Several of the entries in the file deal with the physical characteristics of the network to be simulated. The user can specify the length of the network, the number of stations or nodes on the network, the bit transmission time (and thus the data rate), the maximum allowed message size, the method by which hosts are positioned on the network, the Bandwidth Balancing Modulus and any framing structure imposed by the underlying physical transport medium. If any of these entries are used, they should appear at the top of the description file, since their values may have an impact on units conversions of other parameters.

Keyword: `cable_length` (optional)
Data type expected: real
Allowed values: Any real greater than 0.0
Description: Specifies the actual length of the simulated network, and is used in determining propagation time for signals. The default units are seconds, in which case the specified value is the number of seconds it takes for a signal to traverse the maximum length of the network (propagation delay). An optional unit specifier can be added to change the units accepted. Possible units are "ft" for specifying the cable length in feet, or "m" for specifying it in meters. All values are converted into seconds for use in the program. The unit specifier "sec" can also be used for documentation purposes, but it is not necessary for proper interpretation of the experiment description. If `cable_length` is not specified, the default value is 436.8 microseconds, equivalent to a 50km network, assuming that the propagation rate is approximately 0.67c (a rate typical of optical fiber).

Keyword: `num_nodes` (required)
Data type expected: integer

Allowed values: $1 \leq$ `num_nodes` ≤ 1000
Description: Number of stations on the simulated network. Note that the upper limit is arbitrary and may be adjusted by changing the value of the `NODE_MAX` constant in `dqdb.h`.

Keyword: `bit_time` (optional)
Data type expected: real
Allowed values: any real number > 0.0
Description: Number of seconds it takes to transmit a bit. This quantity allows the data rate to be adjusted. Default is 22.35336195 nanoseconds, corresponding to the DS3 data rate of 44.736 Mbps.

Keyword: `max_msg` (optional)
Data type expected: real
Allowed values: Any value greater than 0.0
Description: The maximum sized message allowed. Default units are seconds, in which case the values specifies the amount of time it takes to transmit the largest allowed message. Optional units specifiers "bytes" and "bits" allow the maximum message to be specified in those quantities. If `max_msg` does not appear, the IEEE 802.6 default of 9188 bytes is used. If `max_msg` and `bit_time` both appear, and `max_msg` is specified in units other than seconds, the `max_msg` entry must occur after the `bit_time` entry, since the value of the `bit_time` entry affects the conversion from bits or bytes to seconds.

Keyword: `protocol` (optional)
Allowed values: DQDB
Description: Specifies what protocol the stations on the network will run. DQDB is the only choice and is the default if this keyword does not appear. This keyword is included to support future extensions of DQDBsim.

Keyword: `node_loc_spec` (required)
Allowed values: One of `H_REGULAR`, `H_UNIFORM`, `H_USERDEF`
Description: Describes how stations are located on the network. If the value is `H_REGULAR`, stations are evenly spaced on the cable. If the value is `H_UNIFORM`, stations are placed randomly on the cable according to a uniform distribution. If the value is `H_USERDEF`, the locations of stations are specified explicitly using the `location` keyword in individual station descriptions. (See Section 3.2.4.)

Keyword: `net_bwb_mod`
Data type expected: integer
Allowed values: must be ≥ 0
Description: Bandwidth Balancing modulus for the simulated network. The default value is 8. A value of 0 turns off the bandwidth balancing mechanism.

Keyword: `slot_overhead`
Data type expected: real

Allowed values: any value ≥ 0.0
Description: Per-slot overhead generated by the underlying physical transport medium, specified in seconds. Default if this keyword does not appear is 0.0.

Keyword: `frame_overhead`
Data type expected: real
Allowed values: any value ≥ 0.0
Description: If the underlying physical transport medium uses a frame structure which carries multiple slots, this parameter specifies the size of the frame header in seconds. If this keyword does not appear, the default value is 0.0.

Keyword: `slot_size`
Data type expected: integer
Allowed values: $0 <$ `slot_size` $\leq$ `max_msg`
Description: The size of a DQDB slot in bytes. If this keyword does not appear, the IEEE 802.6 default of 53 bytes is used.

Keyword: `header_size`
Data type expected: integer
Allowed values: $0 \leq$ `header_size` $\leq$ `slot_size`
Description: The size of the slot header. If this keyword does not appear, the IEEE 802.6 default of 5 bytes is used.

3.2.2 General Experiment Setup

There are several entries in the experiment description file dealing with how the specified experiment will be conducted. These entries are the character string that will be used to generate the names of various data files produced by the simulator, the number of repetitions to conduct for each run of the experiment, various entries dealing with the varying parameter mechanism, and entries specifying the types of reports to generate.

Keyword: `experiment_name` (required)
Data type expected: character string
Description: This entry should coincide with the name of the description file with the ".edf" extension stripped off. It is used to generate the names of the data files created during the simulation as explained in Section 4.4.

Keyword: `expt_type` (required)
Data type expected: character string
Allowed values: one of `SUMMARY`, `INTERVAL` or `TRACE`
Description: Three types of experiment are supported. `SUMMARY` experiments allow the use of the varying parameter mechanism to run a series of experiments based on one experiment description file. Statistics are collected for the entire duration of each

experiment run, and various reports on the overall behavior of the network can be produced. `INTERVAL` experiments collected statistics over a given interval of time (see the `interval` keyword) during a single experiment run and produces various reports which contain a series of records describing the behavior of the network over each interval. `TRACE` experiments do not collect statistics, but rather produce a record of the events that occur during the execution of the protocol. These events are keyed to the event descriptions in Figure 8-1 (p. 189) of the IEEE 802.6 standard [Ins90].

Keyword: `repetitions` (required)
Data type expected: integer
Allowed values: Must be > 0
Description: If the value is larger than one, DQDBsim will automatically conduct the specified number of repetitions of the experiment and present the averages of reported quantities in the output files. This entry is only meaningful for experiments of type `SUMMARY`.

Keyword: `interval` (required for experiments of type INTERVAL)
Data type expected: integer or real
Allowed values: Must be greater than 0
Description: Specifies the length of time over which statistics will be summarized for experiments of type `INTERVAL`. Can be given in seconds (specifier "sec") or slots (specifier "slots"), with the default units being slots.

Keyword: `report_type` (optional)
Data type expected: character string
Allowed values: One of `EXPT_SERIES`, `NODE_AV_SUMMARY`, `BUS_AV_SUMMARY`, `PRI_AV_SUMMARY`, `NET_INTERVAL`, `NODE_INTERVAL`, `BUS_INTERVAL`, `PRI_INTERVAL`, `SUMMARY_REPORTS`, `INTERVAL_RPTS`, `NET_TRACE`
Description: DQDBsim can produce data files which support several different report types. A given report type is available only for a particular experiment type. Recall that the default experiment type is SUMMARY. In this case, the default value for the `report_type` keyword would be `EXPT_SERIES`. If the experiment type is changed, at least one instance of the `report_type` keyword should appear with a value appropriate to the new experiment type. Multiple `report_type` entries may appear in an experiment description. Data files necessary to create all of the requested reports will be created.

> `EXPT_SERIES`: This type of report is primarily of interest for experiments which use the varying parameter feature. Produces a data file of average quantities for each level of the varying parameter which can be used to create various types of plots. Valid only for experiments of type `SUMMARY`.
>
> `NODE_AV_SUMMARY`: A report containing summary information for the individual nodes that make up the network. Valid only for experiments of type `SUMMARY`.
>
> `BUS_AV_SUMMARY`: A report containing summary information for each of the two buses. Valid only for experiments of type `SUMMARY`.
>
> `PRI_AV_SUMMARY`: A report containing summary information for each priority level.

Valid only for experiments of type SUMMARY.

SUMMARY_REPORTS: Creates the data files needed to generate all of the summary reports. Valid only for experiments of type SUMMARY.

NET_INTERVAL: A report containing interval information for the network as a whole. Valid only for experiments of type INTERVAL.

NODE_INTERVAL: A report containing interval information for each of the nodes that make up the network. Valid only for experiments of type INTERVAL.

BUS_INTERVAL: A report containing interval information for each of the two buses. Valid only for experiments of type INTERVAL.

PRI_INTERVAL: A report containing interval information for each of the three priority levels. Valid only for experiments of type INTERVAL.

INTERVAL_RPTS: Creates the data files needed to generate all of the interval reports. Valid only for experiments of type INTERVAL.

NET_TRACE: Print out an identifier for each event as it occurs. Valid only for experiments of type TRACE.

Keyword: prefix_events (optional)
Data type expected: integer
Allowed values: must be greater than or equal to 0
Description: Specifies a number of events which should be processed before statistics gathering begins.

Keyword: varying_param (optional)
Allowed values: one of NONE CABLE_LENGTH, BIT_TIME, NUM_NODES, DEF_ARR_RATE, DEF_ARR_RATE_0, DEF_ARR_RATE_1, DEF_ARR_RATE_2, DEF_MSG_SIZE, DEF_MSG_SIZE_0, DEF_MSG_SIZE_1, DEF_MSG_SIZE_2
Description: This keyword is valid only for experiments of type SUMMARY If the value is NONE (the default), a single experiment run is conducted subject to the value of the repetitions entry, and the results are reported. However, if one of the other values is present, a series of experiments will be run, with the corresponding parameter being adjusted between runs. The summary data files requested via the report_type keyword are produced for each value of the varying parameter. If the EXPT_SERIES value is specified for the report_type keyword, a file is also produced containing the results for the entire series of runs. If CABLE_LENGTH, BIT_TIME, or NUM_NODES is the value, the program variable associated with the respective quantity is adjusted between runs. If DEF_ARR_RATE is the value, then the appropriate parameter of the inter-message arrival time distribution is adjusted between runs. This value assumes that all priority levels use the same default arrival rate distribution. If this is not the case, then the DEF_ARR_RATE_0, DEF_ARR_RATE_1, or DEF_ARR_RATE_2 keyword can be used to specify that the arrival rate for a particular priority level be adjusted. Similarly, if DEF_MSG_SIZE is the value, the appropriate parameter of the message size distribution

is adjusted, again, assuming that all priorities are using the same message size distribution. DEF_MSG_SIZE_0, DEF_MSG_SIZE_1, and DEF_MSG_SIZE_2 can be used to vary the message size for a single priority level. Note that the varying_param keyword can appear only once in an experiment description. (This restriction may be relaxed in future versions of the program.) WARNING: When CABLE_LENGTH is the varying parameter, the locations of the stations on the cable is recomputed every time the cable length is changed. If the original experiment contained any specific positioning information for individual stations using the node_index and location keywords, this information will be ignored in all but the first experiment.
Other required entries: Either vp_step and vp_max or or one or more occurrences of vp_pt must be present.

Keyword: vp_step (optional)
Data type expected: integer or real
Allowed values: determined by the varying parameter; must be positive
Description: If the varying_param entry is present and is not NONE, vp_step specifies the amount by which the varying parameter is to be adjusted between runs. Ignored otherwise.
Other required entries: vp_max

Keyword: vp_max (optional unless vp_step is present)
Data type expected: integer or real
Allowed values: determined by the varying parameter and vp_step; must be greater than the initial value of the varying parameter
Description: If the varying_param entry is present and has a value other than NONE, and vp_step is present, vp_max specifies the maximum value of the varying parameter for which an experiment should be run.
Other required entries: vp_step

Keyword: vp_pt (optional)
Data type expected: integer or real
Allowed values: determined by the varying parameter
Description: If the varying_param entry is present and has a value other than NONE, the vp_pt entry allows the user to specify the series of values that the varying parameter should take on in successive runs. This facility is useful when not all possible values of the varying parameter are of equal interest. For example, it allows runs to be more closely spaced around the saturation point of a network to study the behavior of protocols during the transition from normal conditions to overloaded conditions.

Keyword: num_events (required)
Data type expected: integer
Allowed values: any integer between 1 and MAXINT
Description: Specifies the number of events to be simulated before termination. An event in DQDBsim corresponds to the actions taken by one node in processing a slot. So, for example, running an experiment for 100 events on a network with five stations would cause each station to process 10 slots. (Remember that each station is connected

to two buses.)

3.2.3 Output process specification

The output process for a simulation consists, at the minimum, of definitions for a distribution for message lengths to be generated, a distribution for generating the intermessage delays, and a distribution for message recipients. At least these three distributions must be specified in the experiment description file, and they must be associated by name with the default message length distributions, the default intermessage distributions, and the default recipient distributions for each priority level. If no other provisions are made, all stations will use these distributions to generate traffic, resulting in a homogeneous traffic generation process. Other distributions may also be specified in the file, and these distributions can be associated with the output processes of specific stations or specific DQSMs, as described in Section 3.2.4. This allows the inclusion of stations with differing transmission behaviors on the same simulated network.

Keyword: `dist_name` (Required)
Data type expected: character string
Description: The name of a probability distribution to be used for generating either message lengths or interarrival times. The name will be later referenced to make the associated distribution the default for lengths, interarrival times, or recipients, or to associate the distribution with the lengths, interarrival times, or recipients of a specific station. All other keyword/value pairs associated with the distribution must appear before another instance of the `dist_name` keyword.
Other required entries: `dist_type, parameter1`

Keyword: `dist_type` (one required for each `dist_name`)
Allowed values: One of `EXPON`, `UNIFORM`, `FIXED`, `DISCRETE`, `CONTQUEUE`, `NULL_DIST`, `UNIFORM_RCP`, `USERDEF_RCP`
Description: If the value is `EXPON` the distribution is exponential with mean given by the value of `parameter1`. If the value is `UNIFORM` the distribution is uniform on the interval from the value of `parameter1` to the value of `parameter2`, inclusive. If the value is `FIXED`, the distribution is deterministic with the value of `parameter1` as the constant value of the distribution.* If the value is `DISCRETE` the value of the distribution is determined from a set of (value, probability) pairs specified with the `x_val` and `y_val` keywords. The points specify a cumulative distribution function; i.e. the y values increase toward 1.0. For example, to specify a distribution where messages of size 10, 20, 30, and 40 bytes occur with equal frequency, the following points would be specified:

```
x_val:  10 bytes
y_val:  0.25
```

*To be absolutely clear: a `FIXED` distribution always returns the same value, and this value is the value of the keyword `parameter1` associated with the distribution.

```
x_val:  20 bytes
y_val:  0.50
x_val:  30 bytes
y_val:  0.75
x_val:  40 bytes
y_val:  1.0
```

If the value is `CONTQUEUED` (valid only for distributions that will be used for generation of interarrival times) the generated interarrival time will always be 0, resulting in a station that continuously has messages to send. The value `NULL_DIST` is specified to handle situations where only a single priority level will be generating traffic. In this case, a distribution with type `NULL_DIST` can be created and associated with the default distributions for the other priority levels.

The two remaining types are useful only for distributions which will be used for specifying the recipients that a DQSM will communicate with. `UNIFORM_RCP` specifies that the recipients of the messages generated by the DQSM should be randomly chosen from a uniform distribution consisting of all other stations on the network. `USERDEF_RCP` indicates that a list of recipients and probabilities with the same format as the list for a distribution of type `DISCRETE` will be specified. Currently, no action is taken by a receiving node in the simulation, so this distribution is only used during the setup of the experiment to determine what percentage of the messages generated by a station will be transmitted on each bus.
Other required entries: For `EXPON` and `FIXED`, `parameter1` is required. For `UNIFORM`, `parameter1` and `parameter2` are required. For `DISCRETE` and `USERDEF_RCP`, `point_cnt` and at least one pair of `x_val` and `y_val` keywords are required.

Keyword: `parameter1` (one required for each distribution of type `EXPON`, `UNIFORM`, or `FIXED`)
Data type expected: real
Allowed values: For distributions to be used as message lengths, the value must be less than the maximum allowed message length. For interarrival time distributions, the only requirement is that the value be larger than 0.
Description: For distributions with type `EXPON`, `parameter1` is the mean of the distribution. For distributions with type `UNIFORM`, `parameter1` is the lesser endpoint of the interval from which the distribution is drawn. For distributions with type `FIXED`, `parameter1` is the fixed value of the distribution. For other types, `parameter1` is ignored. If the distribution with which `parameter1` is associated is used to generate message lengths, the default units are seconds. Other available units specifiers are "bits" and "bytes." If the distribution is used to generate intermessage intervals, the units are messages per second.

Keyword: `parameter2` (required for each distribution of type `UNIFORM`)
Data type expected: real

Allowed values: Must be larger than the value of `parameter1`. For distributions to be used as message lengths, the value must be less than the maximum allowed message length.
Description: The `parameter2` entry is used as the greater endpoint of the interval from which the uniform distribution is drawn.

Keyword: `point_cnt` (required for each distribution of type `DISCRETE` or `USERDEF_RCP`)
Data type expected: integer
Allowed values: $0 <$ `point_cnt` ≤ 100
Description: The number of points that will be specified for a user-defined distribution of type `DISCRETE` or `USERDEF_RCP`. (The upper limit on the number of points can be adjusted by changing the definition of the constant MAXPTS in the file ds_distrib.h)
Other required entries: The correct number of `x_val`/ `y_val` entries must follow the `point_cnt` entry.

Keyword: `x_val` (optional)
Data type expected: real
Allowed values: For distributions used for message length, the value must be less than the maximum allowed message size. For distributions used for intermessage times, the value must be larger than zero.
Description: For discrete distributions, a list of {`x_val`, `y_val`} pairs appear in the description file. The `x_val` values are the values that will be returned by the the distribution for message lengths, intermessage times, or recipient IDs. Note that for discrete distributions which will be used as intermessage time distributions, the x values should be the actual length of the interval to be generated in seconds, **NOT** the rate of message generation in messages per second! For recipient list distributions, the `x_val` values should be the node number of a station in the simulated network.
Other required entries: Each `x_val` must be followed by a `y_val`.

Keyword: `y_val` (optional)
Data type expected: real
Allowed values: Must be between 0.0 and 1.0.
Description: The cumulative probability that the corresponding `x_val` value will be returned by a discrete distribution. The `y_val` values for a distribution should increase to a maximum of 1.0, as described in the explanation for the `dist_type` keyword.
Other required entries: Should be directly preceded by the corresponding `x_val`.

Keyword: `def_interpkt_dist_0` (required)
Data type expected: character string
Allowed values: Must be the `dist_name` of a previously defined distribution.
Description: The name of the default intermessage arrival time distribution for priority level 0 for the experiment. Each station uses this distribution to generate intermessage times for priority level 0 unless a station specific distribution is chosen.

Keyword: `def_interpkt_dist_1` (required)
Data type expected: character string
Allowed values: Must be the `dist_name` of a previously defined distribution.
Description: Similar to `def_interpkt_dist_0`.

Keyword: `def_interpkt_dist_2` (required)
Data type expected: character string
Allowed values: Must be the `dist_name` of a previously defined distribution.
Description: Similar to `def_interpkt_dist_0`.

Keyword: `def_length_dist_0` (required)
Data type expected: character string
Allowed values: Must be the `dist_name` of a previously defined distribution.
Description: The name of the default message length distribution for priority level 0 for the experiment. Each station uses this distribution to generate message lengths at priority level 0 unless a station specific distribution is chosen.

Keyword: `def_length_dist_1` (required)
Data type expected: character string
Allowed values: Must be the `dist_name` of a previously defined distribution.
Description: Similar to `def_length_dist_0`.

Keyword: `def_length_dist_2` (required)
Data type expected: character string
Allowed values: Must be the `dist_name` of a previously defined distribution.
Description: Similar to `def_length_dist_0`.

Keyword: `def_rcpt_dist_0` (required)
Data type expected: character string
Allowed values: Must be the `dist_name` of a previously defined distribution.
Description: The name of the default recipient distribution for priority level 0 for the experiment. Each station uses this distribution to decide which bus to transmit a given message on for priority level 0 unless a station specific distribution is chosen.

Keyword: `def_rcpt_dist_1` (required)
Data type expected: character string
Allowed values: Must be the `dist_name` of a previously defined distribution.
Description: Similar to `def_rcpt_dist_0`.

Keyword: `def_rcpt_dist_2` (required)
Data type expected: character string
Allowed values: Must be the `dist_name` of a previously defined distribution.
Description: Similar to `def_rcpt_dist_0`.

3.2.4 Individual node specifications

In the absence of instructions to the contrary, the placement and behavior of all nodes on the network are determined by the value of `node_loc_spec`, the default message length, intermessage, and recipient distributions, and the Bandwidth Balancing modulus for the network. If a different behavior or placement is desired for individual stations, the `node_index` keyword in combination with one or more of the other associated keywords can be used to change the characteristics for individual stations.

Keyword: `node_index` (optional)
Data type expected: integer
Allowed values: $0 \leq$ `node_index` $<$ `num_nodes`
Description: The index of one of the simulated stations. This entry indicates the beginning of a description of a particular station on the network which will differ in some way from stations following the experiment defaults.

Keyword: `location` (optional)
Data type expected: real
Allowed values: $0 \leq$ `location` $\leq$ `cable_length`
Description: `Location` allows stations to be explicitly arranged on the network. The associated value overrides the station location which would have been generated from the `node_loc_spec` value for the experiment. Associated with the station specified by the most recent `node_index` value. Note that it is important that the location value be such that all stations maintain their relative ordering.

Keyword: `length_dist_0` (optional)
Data type expected: character string
Allowed values: The `dist_name` of a previously defined distribution.
Description: Specifies the message length distribution that will be used by a particular station at priority level 0. Associated with the station specified by the most recent `node_index` value.

Keyword: `length_dist_1` (optional)
Data type expected: character string
Allowed values: The `dist_name` of a previously defined distribution.
Description: Specifies the message length distribution that will be used by a particular station at priority level 1. Associated with the station specified by the most recent

node_index value.

Keyword: length_dist_2 (optional)
Data type expected: character string
Allowed values: The dist_name of a previously defined distribution.
Description: Specifies the message length distribution that will be used by a particular station at priority level 2. Associated with the station specified by the most recent node_index value.

Keyword: ipi_dist_0 (optional)
Data type expected: character string
Allowed values: The dist_name of a previously defined distribution.
Description: Specifies the intermessage interval distribution that will be used by a particular station at priority level 0. Associated with the station specified by the most recent node_index value.

Keyword: ipi_dist_1 (optional)
Data type expected: character string
Allowed values: The dist_name of a previously defined distribution.
Description: Specifies the intermessage interval distribution that will be used by a particular station at priority level 1. Associated with the station specified by the most recent node_index value.

Keyword: ipi_dist_2 (optional)
Data type expected: character string
Allowed values: The dist_name of a previously defined distribution.
Description: Specifies the intermessage interval distribution that will be used by a particular station at priority level 2. Associated with the station specified by the most recent node_index value.

Keyword: rcpt_dist_0 (optional)
Data type expected: character string
Allowed values: The dist_name of a previously defined distribution.
Description: Specifies the recipient distribution that will be used by a particular station at priority level 0. Associated with the station specified by the most recent node_index value.

Keyword: rcpt_dist_1 (optional)
Data type expected: character string
Allowed values: The dist_name of a previously defined distribution.
Description: Specifies the recipient distribution that will be used by a particular station at priority level 1. Associated with the station specified by the most recent

node_index value.

Keyword: rcpt_dist_2 (optional)
Data type expected: character string
Allowed values: The dist_name of a previously defined distribution.
Description: Specifies the recipient distribution that will be used by a particular station at priority level 2. Associated with the station specified by the most recent node_index value.

Keyword: node_bwb_mod
Data type expected: integer
Allowed values: any integer ≥ 0
Description: Allows the Bandwidth Balancing Modulus for the node to be set. Note that in the 802.6 standard, BWB_MOD is a network-wide parameter; if your intent is to simulate a network which adheres to the standard, do not use this keyword.

Chapter 4

Data File Formats

Some types of DQDBsim experiments can produce great volumes of data, so a mechanism for specifying which data to record has been built in to the program. Depending on the `report_type` entries in an experiment description, a number of different data files may be produced by an experiment execution. This section discusses the purpose and format of the various files.

All data files are in binary format. This decision was made primarily to reduce the amount of disk space that data files take up. It has the disadvantage that data files are not portable across the architectures that the program runs on due to byte-ordering differences. Two utility programs, `ds_plot` and `ds_display`, have been provided to manipulate and display the data files. The `ds_plot` program can be used to combine data abstracted from multiple experiment series files into a form which is readable by two freely distributed graphing program for the X Window System, `xgraph` and the ACE/GR plotting package. The `ds_display` program prints out human readable versions of the data files. (See Sections 5.1, 5.2, 5.3, and 5.4.)

4.1 The Report Header

All data files share a common header. This header contains the following information:

Name: the value of the `expt_name` entry from the experiment description file.

Version: the version of the simulator which produced the file.

Timestamp: the date and time the simulation run described by the file began.

Hostname: the name of the computer where the simulator was run.

User: the name of the user who ran the simulation. (May not be available on some platforms.)

This information is printed at the beginning of any report produced by the `ds_display` program.

4.2 Series Summary File

A series summary file is generated for experiments of type `SUMMARY` when the `EXPT_SERIES` or `SUMMARY_RPTS` report type has been requested. It holds a summary of an entire series of experiments run using the varying parameter facility of DQDBsim. Each line is a summary of the statistics for a single run or a set of repetitions run with the same parameters. The following quantities are present in each record for this type of data file:

Sequence number: The index number of the run that the record describes.

Offered load: The sum of the traffic load produced by each node specified in the experiment description file transmitting on an otherwise idle network.

Throughput: The ratio of time spent in actual transmission to the total time elapsed.

Data rate: The actual data rate achieved on average during the experiment. This quantity is calculated by multiplying the throughput by the maximum data rate for the network.

Av. message delay: The average queueing delay (as defined in Section 1.3.2) experienced by messages transmitted during the experiment. This quantity is calculated by dividing the total delay for the experiment by the number of successful message transmissions. This metric is the closest analogue for comparing DQDB to other protocols which support variable length messages directly.

Message delay variance: The variance of the average message queueing delay, calculated from the delay histogram.

Av. segment delay: The average queueing delay (as defined in Section 1.3.2) experienced by segments transmitted during the experiment. Because DQDB uses a fixed slot size for transmission, variable length messages must often be broken down into multiple segments of the appropriate length for transmission. This quantity is calculated by dividing the total delay for the experiment by the number of successful segment transmissions.

Segment delay variance: The variance of the average message queueing delay, calculated from the delay histogram.

Arrival rate: The average number of messages transmitted per second over the duration of the experiment.

Message size: The average size of messages transmitted during the experiment, in bytes. Calculated by dividing the total transmission time for the experiment by the number of successful transmissions (giving the average transmission time per message) then converting to bytes.

Message overhead: Length in seconds of the overhead generated by the protocol or underlying transmission medium characteristics for each message transmitted.

Node count: The number of stations (transmission sources) on the simulated network.

Cable length: The actual length of the cable to which the stations were attached in the experiment. Used in calculating signal propagation times.

Maximum Data Rate: The capacity of the channel being simulated. Calculated from the user-specified bit time.

Total messages: The total number of messages which were successfully transmitted over the duration of the experiment.

Total segments: The total number of segments which were successfully transmitted over the duration of the experiment.

Total slots: The total number of slots which traversed the network over the duration of the experiment. Not all of these slots were necessarily used.

Duration Simulated length of the run in seconds.

Slot time: Length of each slot in seconds.

Slot size: Length of each slot in bytes.

Header time: Length of the slot header in seconds.

Header size: Length of the slot header in seconds.

Some of these quantities are included in the file for the purpose of calculating derived quantities and do not appear directly in any reports.

Files generated in this format are set up to be processed by `ds_plot`, a data analysis utility described in Section 5.1. In addition to the statistics collected for each run, the values of all of the parameters that can vary from one run to the next by the varying parameter mechanism are included.

4.3 Interval data files

Most of the other data files produced by DQDBsim use the interval data format. Each interval data record contains the following information (fields without descriptions are the same as the corresponding field in the Experiment Series Summary format):

Sequence number: If produced by an experiment of type `INTERVAL`, this field specifies interval during which the data in the record was generated.

Throughput

Data rate

Av. message delay

Message delay variance

Av. segment delay

Segment delay variance

Arrival rate

Message size

Message overhead

Maximum Data Rate

Total messages

Total segments

Total slots

Slot time

4.4 Data File Generation

Data files are generated in response to the values for the `report_type` keyword included in the experiment description file. The following table shows the types of files generated:

Contents	Extension	Expt type	`report_type` flag
Experiment series summary	.esu	SUMMARY	EXPT_SERIES
Node average summary	.nav	SUMMARY	NODE_AV_SUMMARY
Bus average summary	.bav	SUMMARY	BUS_AV_SUMMARY
Priority average summary	.pav	SUMMARY	PRI_AV_SUMMARY
All summary reports		SUMMARY	SUMMARY_RPTS
Network interval report	.ivl	INTERVAL	NET_INTERVAL
Per node interval report	.niv	INTERVAL	NODE_INTERVAL
Per bus interval report	.biv	INTERVAL	BUS_INTERVAL
Per priority interval report	.piv	INTERVAL	PRI_INTERVAL
All interval reports		INTERVAL	INTERVAL_RPTS
Event trace	stdout	TRACE	NET_TRACE

For experiments of type `SUMMARY` which use the varying parameter mechanism, the names of the files generated will contain a component which indicated the parameter that varied and a two digit index which indicates which run of the experiment the file pertains to. The parameter identifiers are as follows:

Identifier	**Varying parameter**
cl	Cable length
bt	Bit time
nn	Number of nodes
ar	Overall arrival rate
ar0	Arrival rate for priority level 0
ar1	Arrival rate for priority level 1
ar2	Arrival rate for priority level 2
ms	Overall Message size
ms0	Message size for priority level 0
ms1	Message size for priority level 1
ms2	Message size for priority level 2

For example, a summary experiment called "test" which specifies arrival rate for priority level 0 as its varying parameter and requests node average reports would generate files with names like `testar0-00.nav`.

Interval experiments also have their own naming convention. Since interval reports can be requested per node, per bus or per priority level, the corresponding data files will append a single letter (n, b or p), followed by a four digit index to the name of the experiment. The index indicates either the node number, the bus number, or the priority number that the file is associated with. (This is overkill in most cases, but keeping the number of digits constant over the three varieties of report kept the code cleaner.) For example, one of the node interval reports for an interval experiment called "itest" would be called `itest_n0001.niv`.

Chapter 5

Related Utilities

Two programs are used to process and display data generated by the simulator. `Ds_plot` is an interactive program which allows the user to choose quantities from one or more series or interval data files and produces output suitable for a plotting package to display. This program can produce output suitable for use by two plotting packages, the xgraph package and the ACE/GR family of plotting packages (`xvgr/xmgr`). The `ds_display` program can be used to display the (binary format) data files in human readable form.

5.1 The ds_plot program

`Ds_plot` transforms data files into a format which can be plotted using the either the `xgraph` program (see Section 5.3) or the `xvgr/xmgr` programs from the ACE/GR plotting package (see Section 5.4). `Ds_plot` allows the user to interactively choose which quantities to plot on the x and y axis of a two dimensional graph, and to specify a name for each data set. Multiple files can be plotted on the same graph. The following command line options are available:

`ds_plot [-o` *output_file*`] [-f` *output_format* `] [-u] [-V]`

where

output_file	is the name to use for the plot file
output_format	has the value `XGRAPH` for output readable by the xgraph plotting package, or `ACEGR` for output readable by the ACE/GR programs xvgr or xmgr. The default if the `-f` option is not specified is XGRAPH.
`-u`	prints the usage message for the program.
`-V`	prints the ds_plot version number.

The user of `ds_plot` is prompted for all other necessary information.

5.2 The ds_display program

Because the data files produced by DQDBsim are in binary format, it is not possible to simply user a text editor to look at their contents. The `ds_display` program is provided for that purpose. The command line usage of the program is as follows:

`ds_display -i` *input_file* `[-o` *output_file*`] [-c] [-V]`

where

input_file	is the (required) name of the file to display.
output_file	is the (optional) name of the file to put the output into.
`-c`	indicates that the output should be in "columnar" form rather than "tabular" form. Tabular form is the default and puts all information about an experiment run or interval on one long line. Columnar output breaks the information about an experiment run into several labeled lines and is sometimes easier to read.
`-V`	prints out the ds_display version number.

Each type of data file contains a code which indicates what type of file it is, so it is not necessary to tell `ds_display` what kind of file you are passing to it.

5.3 The xgraph program

Xgraph is an X11 based plotting program with many nice features, including the ability to zoom in on portions of a graph and facilities to produce laser printer output of graphs or data files that can be included as figures in LaTeX documents. It was written by David Harrison of the UC-Berkeley Electronics Research Lab. Data produced by DQDBsim can be selectively converted to the format read by xgraph using the ds_plot program. For further details, see *xgraph(1)*.

5.4 The ACE/GR programs

The ACE/gr package is a set of plotting with an X Window system interface called `xmgr`, which uses the Motif user interface conventions. (There is also and XView version called `xvgr`, but it is no longer supported.) The package was written by Paul Turner of the Oregon Graduate Institute of Science and Technology Center for Coastal and Land-Margin

Research. Among other features, it allows interactive adjustment of many graph formatting options, and can generate output in postscript or FrameMaker interchange format. It is available for anonymous ftp from ftp://ftp.teleport.com/pub/users/pturner/acegr. The `ds_plot` program can produce output which can be loaded and displayed by `xmgr` or `xvgr`.

Appendix A

Appendix: Sample experiment description file

```
# Duplicates the behavior of the experiment pictured in Figure 7 of
# the July 1992 Transactions on Communications article ‘‘DQDB Networks
# with and without Bandwidth Balancing’’ by Hahne, Choudhury, and
# Maxemchuk. [HCM92]
experiment_name: hcm92-2

# Network characteristics
num_events: 200000
bit_time: 6.666666667e-9 #150Mbps
cable_length: 48 km
num_nodes: 4
protocol: DQDB
net_bwb_mod: 9
node_loc_spec: N_REGULAR # Evenly spaced

# Experiment configuration options
expt_type: INTERVAL
interval: 112 slots # Approx. length of network
report_type: NET_INTERVAL
report_type: NODE_INTERVAL
report_type: BUS_INTERVAL
repetitions: 1
varying_param: NONE

# Descriptions of probability distributions to be used in the experiment.
dist_name: LongFile # Long file transfer
dist_type: FIXED
parameter1: 300000 bytes

dist_name: MediumFile # Medium file transfer
dist_type: FIXED
parameter1: 200000 bytes
```

```
dist_name: ShortFile # Short file transfer
dist_type: FIXED
parameter1: 100000 bytes

dist_name: primaryIpi
dist_type: FIXED
parameter1: 1

dist_name: RcpDist # All send to node #3
dist_type: USERDEF_RCP
point_cnt: 1
x_val: 3
y_val: 1.0

dist_name: null_dist
dist_type: NULL_DIST

# Default distributions
def_interpkt_dist_0: primaryIpi # Only priority 0 transmits
def_interpkt_dist_1: null_dist
def_interpkt_dist_2: null_dist
def_length_dist_0: LongFile
def_length_dist_1: null_dist
def_length_dist_2: null_dist
def_rcpt_dist_0: RcpDist
def_rcpt_dist_1: null_dist
def_rcpt_dist_2: null_dist

# Specific host descriptions:  these keywords are repeated for each
# host whose behavior should be different from the default in some way
node_index: 0
node_fuse: 112 slots # Let cables fill w/ slots

node_index: 1
length_dist_0: MediumFile
node_fuse: 5040 slots # 45 intervals

node_index: 2
length_dist_0: ShortFile
node_fuse: 1120 slots # 10 intervals

# Node 3 only participates as a destination, it does not produce traffic.
node_index: 3
length_dist_0: null_dist
ipi_dist_0: null_dist
rcpt_dist_0: null_dist
```

Bibliography

[HCM92] Ellen L. Hahne, Abhijit K. Choudhury, and Nicholas F. Maxemchuk. DQDB Networks with and without Bandwidth Balancing. *IEEE Transactions on Communications*, 40(7):1192 – 1204, July 1992.

[Ins90] Institute of Electrical and Electronics Engineers. *IEEE Standards for Local and Metropolitan Area Networks: Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN)*, 1990. IEEE Std 802.6-1990.

[SG86] Robert Scheifler and Jim Gettys. The X Window System. *ACM Transactions on Graphics*, 5(2):79 – 109, April 1986.