University of Edinburgh

Department of Computer Science

The Structure and Uses of The Edinburgh Remote Terminal Emulator

by

J.C. Adams, W.S. Currie,
B.A.C. Gilmore

Internal Report CSR - 12 - 77

James Clerk Maxwell Building
The King's Buildings
Mayfield Road
Edinburgh
EH9 3JZ

September 1977

The Structure and Uses of The Edinburgh Remote Terminal Emulator

J. C. ADAMS(*)
Department of Computer Science,
University of Edinburgh,
James Clerk Maxwell Building,
The Kings Buildings,
Mayfield Road,
Edinburgh EH9 3JZ

W. S. CURRIE(**) & B. A. C. GILMORE
Edinburgh Regional Computing Centre
The Kings Buildings,
Mayfield Road,
Edinburgh EH9 3JZ

(*) Work supported by Science Research Council Grant BRG-93960
(**) Past work supported by Science Research Council Grant BRG-96084

ABSTRACT

Remote Terminal Emulation is an approach to the testing and evaluation of multi-access computer systems in which a workload driver is implemented external to and independent of the system being tested.

Remote Terminal Emulators may be used in many stages of system development eg. initial checkout, acceptance tests and tuning.

The Edinburgh Remote Terminal Emulator (ERTE) is a system designed to exert a specified interactive workload on a multi-access system such that it appears to the system that it is connected to live terminal devices. The emulator runs on a PDP 11/40 computer under the control of the DEIMOS operating system, which like the emulator is written in the high-level language IMP.

We describe the structure and development of ERTE with particular emphasis on its modularity, the use of a message based operating system, and the IMP language, which provide a flexible and easily modified tool.

Experiments, both current and projected, using ERTE are discussed and our experience of developing and using a terminal emulator is reviewed.

KEYWORDS

performance evaluation, benchmarks, interactive systems, measurement, communiations, teleprocessing

INTRODUCTION

Measurement and evaluation techniques used on computer systems in the past have largely been machine oriented [3]. Traditionally "scientific" systems have been evaluated in terms of raw instruction speed,a suitable instruction mix for the desired environment being set up,mostly of arithmetic instructions,and run off on various configurations. On the other hand,"commercial" systems have been compared more on I/O rates than on processor speeds.

With the advent of batch systems, throughput, the number of jobs executed in unit time,became a common measure. In addition both hardware and software components of a system were individually monitored to measure utilisation or other desired characteristics.

Whereas in a batch system the performance in processing individual jobs is of relatively little importance, in interactive multi-access systems the on-line user's view of the system's performance is of paramount concern. Several user oriented measures have been introduced such as response times,system availability,reliability and accessability,which in turn require new evaluation techniques.

One such technique is Remote Terminal Emulation.

REMOTE TERMINAL EMULATION

Remote Terminal Emulation [14] is an approach to the testing and evaluation of multi-access computer systems in which a representation of user workload and behaviour is presented to the system being tested (the Target system) by a driver external to and independent of the target system.

The driver is connected to the target system either directly or through a communications network so that the target system is unable to detect any difference between the driver and real users at real terminals.

An important part of the driver is a monitor which records data pertaining to the driver/target system interaction. Performance characteristics can then be determined by a later analysis of this log.

Two of the most important features of this technique are its repeatability and flexibility. The RTE is able to present any workload time after time to the target system, so any detected performance differences are due to the target system alone. To be flexible the RTE should be as machine independent as possible and must be able to cope with the unexpected,since interactive systems are non-deterministic; for example, operator messages to users may arrive at any time.

Real users working from scripts can be used to present workloads to the system. However this is a very tedious process, prone to error. A further alternative is to implement the driver

within the target system, but this could cause gross interference and would be highly system dependent. [12]

There are four principal areas of use for an RTE :-

1) Tuning. If the workload presented to the target system is a reasonably accurate reflection of the real user workload then by running a series of experiments on the system with changes in the scheduling parameters the RTE may be used as a method of tuning the system. This can assist in the improvement of overall performance. The data can also show how changes in the scheduling parameters discriminate in favour of or against classes of work.

2) Experimentation. One of the major difficulties in research into any complex computing system is the lack of a body of consistent data. Such data may be used for empirical evaluation and for the validation and calibration of models which may be used for prediction of the system's performance. One of the major drawbacks to obtaining such a set of data is the variability in user workloads. An RTE provides a method of keeping the workload fixed over a set of experiments in which other system parameters (hardware components, scheduling algorithms) may be changed and a consistent set of data obtained.

3) Development. The effect of a system failure in a large time-sharing system in terms of lost work, frustration and user discontent is very high. An RTE gives a convenient way of testing major software/hardware changes in the system before

they are released to the general user population. Also during the development stages of a completely new system, an RTE provides a way of presenting the system with a repeatable workload which may be used in the tracing of errors or possible performance bottlenecks.

4) Procurement. RTE'S provide a method of obtaining objective performance data on several systems when a new system is being selected (Benchmarking). Also when a system has been selected an RTE may be used to check that the system does in fact fulfil its performance goals before the system is put into service.

WORKLOAD REPRESENTATION

Remote terminal emulators require a workload to present to the target system. This is normally held in the form of scripts which give the user commands,pauses etc. to describe the behaviour of one or more users.

The commands in the scripts are in the final instance machine dependent,but all think times,typing rates etc. can be specified in an independent form and translated for actual use, a method preferable for multi-machine use of the RTE.

It may also be possible to provide in the scripts specific responses which the RTE is to check against actual target system responses,or similarly times within which a response has to occur.

The Edinburgh Remote Terminal Emulator (ERTE) is an implementation of a general purpose workload driver suitable for the above-mentioned applications.

THE LOGICAL STRUCTURE OF ERTE.

The logical structure of the Edinburgh Remote Terminal Emulator is given in fig. 1. The data to simulate user behaviour is held in scripts, and consists of the commands for the target system, the user's typing efficiency, the thinking time between getting a target system response and typing the next command, and the terminal line speed.

The next level is emulation of a terminal. This level receives user commands and sends them towards the target machine, taking into account the terminal line speed. It also receives output from the target system and simulates printing this on the terminal (virtual printing). At this level the monitor log,consisting of the contents and timing of all commands to and responses from the target system, is recorded. This log is then available for both emulator validation and measurement of system characteristics,such as response time.

Below the virtual terminal level are two levels of protocol which are communication and target system dependent. The terminal protocol creates packets for individual terminals and handles the

end-to-end protocol between the RTE and the target mainframe. The line protocol ensures the correct transmission and reception of packets at the physical link level between the RTE and the next part of the communications network, which need not necessarily be the target mainframe.

PHYSICAL IMPLEMENTATION OF ERTE

THE HARDWARE

The emulator runs on a Digital Equipment Corporation PDP 11/40 computer with 32K words of core store, and one RK05 cartridge disc unit (1.2 Mbytes) which holds the system software. Mass storage is provided by a 66 Mbyte Ampex DM980 disc drive which currently holds both the script files and the monitor log data.

Communication with the target,currently the Edinburgh Multi Access System [15], is through a DQS11-E Synchronous Line Interface at a line speed of 9.6 Kb. To EMAS the emulator is simply another Terminal Control Processor on a single line into the Front End Processor (a PDP 11/45) [5].

THE SOFTWARE

The emulator software consists of several cooperating tasks (fig. 2) running under the DEIMOS operating system,which was developed locally for a number of communications applications.

DEIMOS is a general purpose multi-tasking system which will support a number of general user programs [6]. Each program in the system, which includes device handlers, is run as a separate task in its own 32K word virtual memory. Tasks may co-operate by sending messages, in particular all I/O is performed in this way.

DEIMOS is based on a small, fixed Kernel which performs four main functions.

a) It allocates the CPU to user programs and device tasks according to their respective priorities.

b) It stores and forwards messages.

c) It enables tasks to map onto other task's virtual memories to facilitate the passing of information.

d) It receives interrupts from all devices attached to the system forming them into messages for interpretation by the respective handler task.

All other system functions, including file system handling etc, are carried out by standard tasks. This enables systems to be easily tailored for specific machines with very little overhead.

With the exception of a small part of DEIMOS, namely hardware register loading and compiler run time support, all the emulator software is written in the high level language IMP ([11],[10],[13]). The use of the high level language has, in our opinion, greatly speeded up the development of DEIMOS and ERTE. The initial software was written several times faster than a similar, but significantly different, system which was written almost entirely in assembler [7]. Similar benefits were also discernible in the ease with which errors were identified and corrected. The ability to adapt and extend the software to cope with different environments is proven. [7] Great care has been taken in the definition of modules and their interfaces, enabling sections to be replaced easily.

The principal tasks are as follows [1] :-

1)THE SCRIPT TASKS

Each script task emulates a set of virtual users, currently up to 16 per task, there normally being several script tasks in an ERTE configuration (see current uses section).

A script task is controlled by a parameter file, read in as part of the script task initialisation phase. The file contains the number of users for this task, a typing efficiency for those users and a delay factor between the initial user logon times. The parameter file also associates a script file name and terminal line speed with each virtual user.

The script task maintains pointers to the script files on disc for each user and reads items as required. The first line of a script file entry is the user input to the target system. Using the length of this input and the user's typing efficiency, a typing delay is calculated and effected before the input is passed to the terminal protocol handler and thence to the target system.

When the target system responds with a request for more user input the script task executes a user think time delay previously read from the script file before commencing with the next input.

Currently output from the target system is not returned to the script tasks, although provision has been made to allow the checking of target system responses against entries in the script files.

The script task continues to read from each script file until all the files are exhausted or the emulator is aborted by operator intervention. Note that one script file may contain several consecutive user sessions at a virtual terminal with different users.

It is possible for the operator to interrogate the script tasks to get the status of each user, cause a script file to be repeated, or abort the script task.

The script tasks are entirely target system independent and can easily be reconfigured for various numbers of users.

2)THE TERMINAL PROTOCOL HANDLER

The terminal protocol handler takes care of the following four functions:-

a) The handling of all buffering for messages between virtual users and the target, by maintaining a central buffer pool from which the SCRIPT tasks and Line Protocol Handler task request space as necessary.

b) Terminal Protocol Handling. The appropriate teminal protocol [8] is added to or removed from each message and control is exercised over the flow of messages to and from the target.

c) Calculation of Typing Delays. The delays involved in the typing of output are taken care of here and a message is sent to an appropriate SCRIPT task only when a reply to a user command has been 'typed' and the system has indicated that it is ready to receive more input.

d) Maintaining the Performance Log. During a run all messages whether generated by the users or the target system are recorded with a time stamp in a log file held on disc. This log may be analysed later to obtain performance data for the run. The fact that all input and output is recorded is a validation procedure which ensures that the target system has executed the specified scripts. During a run the operator commands available [1] enable checking to be made that the scripts are being processed.

3)THE LINE PROTOCOL AND DEVICE HANDLER

The Line Protocol Handler is passed messages by the Terminal Protocol Handler and ensures their correct transmission to the next stage in the communication network. The current implementation is run under the HDLC protocol [8].

The use of HDLC has several benefits, including the fact that correct transmission and reception of messages can be guaranteed. The distances involved are short enough to enable links without error correction to be used. However the overall use of ERTE in this manner would be limited by restricting its physical position with regard to any target mainframe. In addition, without error correction there would always be a slight risk of losing or corrupting messages.

HDLC with its facilities for transmitting and receiving several messages in advance of an acknowledgement leads to significantly better throughput than simpler protocols.

The protocol handler passes the messages to a device handler which controls the communications hardware.

4)THE TIMER AND INITIALISATION TASKS

The timer task is used by the script and terminal protocol tasks to return messages after a specified interval. For example, the script task, as one of its activities, sends a message to the timer which includes a script number and a time in seconds for a user think delay. After that time, the timer replies with a message including the script number. In this way think delays, typing

delays and printing delays may easily be handled. The timer also provides the time stamping facility for the monitor log.

The initialisation task loads the other tasks, handles all fatal error messages and the controlled close down of the RTE.

DATA ACQUISITION AND REDUCTION

A log of all messages and the time at which they appeared is recorded by the Terminal Protocol Handler. This file is analysed to obtain appropriate performance data. A suite of analysis programs produces counts, distributions, means, variences etc. for a variety of measures. These measures may be obtained over all users or any specified subclass of users and include:

a) Response Times :- either averaged over all commands given or for specific subclasses of commands. The definition of response time will vary according to the target system and the application, however two response times are currently considered:

   1) First Response :- the time from giving a command to the system until the first character of the reply is received (no typing or subsequent delays considered).

   2) Final Response :- the time from giving a command to the

system until all of of the reply is typed and the target has indicated that it is ready to receive a further command.

b) Command Types :- this is a frequency count of the command subclasses which gives a measure of how changes in the configuration or scheduling algorithms in the target discriminate for or against certain forms of interaction.

c) I/O Rates Achieved :- this is a count of the number of characters input and output from each virtual terminal. This may be used as a measure of the throughput capacity of the software/hardware handling interactive communications.

CURRENT AND PROPOSED USES

ERTE has resulted from a number of year's work in Edinburgh on interactive benchmarks. The first version was used to provide an interactive benchmark for the Edinburgh Multi-Access System [15] in 1973 and to produce a workload measure to match against the performance of an ICL 2980 that the Regional Computing Organisation was buying. This benchmark was run on a PDP 11/45 using the first version of DEIMOS.

The next version has been used by Southhampton University since August 1976, and latterly by Kent University to benchmark their ICL 2900s. This version runs on a PDP 11 (with segmentation) using DEIMOS and software similar to ERTE. The significant difference between ERTE and the Southampton benchmark is that instead of a synchronous line, Southampton use two asynchronous multiplexers, each capable of supporting 16 lines, each of which appears to the 2970 as a teletype. Consequently, instead of a Line Protocol Handler, there is a Multiplexer Handler which fans out the messages from the Terminal Protocol Task to the discrete lines.

At present the major use of ERTE is in the running of a set of controlled experiments on EMAS. Within these, ERTE is used both as a method of providing a range of repeatable workloads and as a measurement device. EMAS is measured running under realistic loads on a variety of configurations and the data so obtained is used to validate and calibrate a range of models of this class of system. The measurements are collected by ERTE (see Section 5) and within

EMAS [2].

The ability exists to vary three major hardware parameters within the target- the size of main memory (up to 1Mbyte), the size of secondary memory (up to 12Mbytes) and the channel bandwidth to secondary memory (1 or 2 channels). As EMAS is also written in the high level language IMP and is relatively easy to modify for a system of this size and complexity, almost any part of the software system could be considered for changes. However experiments have already been carried out and will continue to be carried out on the scheduling algorithms and working set [4] calculation mechanisms employed.

The workload currently used is one defined from measurements taken on EMAS some time ago [2]. This mimics real users in that a set of base program and data files are available in the target machine for use during an ERTE run. These are not altered but any other files created during a run are automatically destroyed before the next run. Research is also underway into the automatic production of new synthetic workloads built from a set of basic units and validated against measurements taken on the system in normal use. Using these synthetic workloads it is intended to further investigate the effects of load and balance within load upon EMAS'S performance and to carry out a similar exercise on different time-shared systems eg. PDP-10 and ICL 2900.

ERTE may also be used to test system changes and hopefully eliminate errors before the system is released to normal users. The development of ERTE has been in parallel with many major changes in the communications network software. ERTE has shown its usefulness

validating models of time-sharing systems. It is hoped, however, that ERTE will provide a vehicle for further research into more rigorous measurement and empirical evaluation techniques. The work currently being carried out is on large time-sharing systems, an area which has seen considerable research effort in the last 15 years. However, having developed techniques on this class of system it is hoped that these will be easily extendable to any form of teleprocessing system or indeed communications network.

FIGURE 1 : THE LOGICAL STRUCTURE OF ERTE.

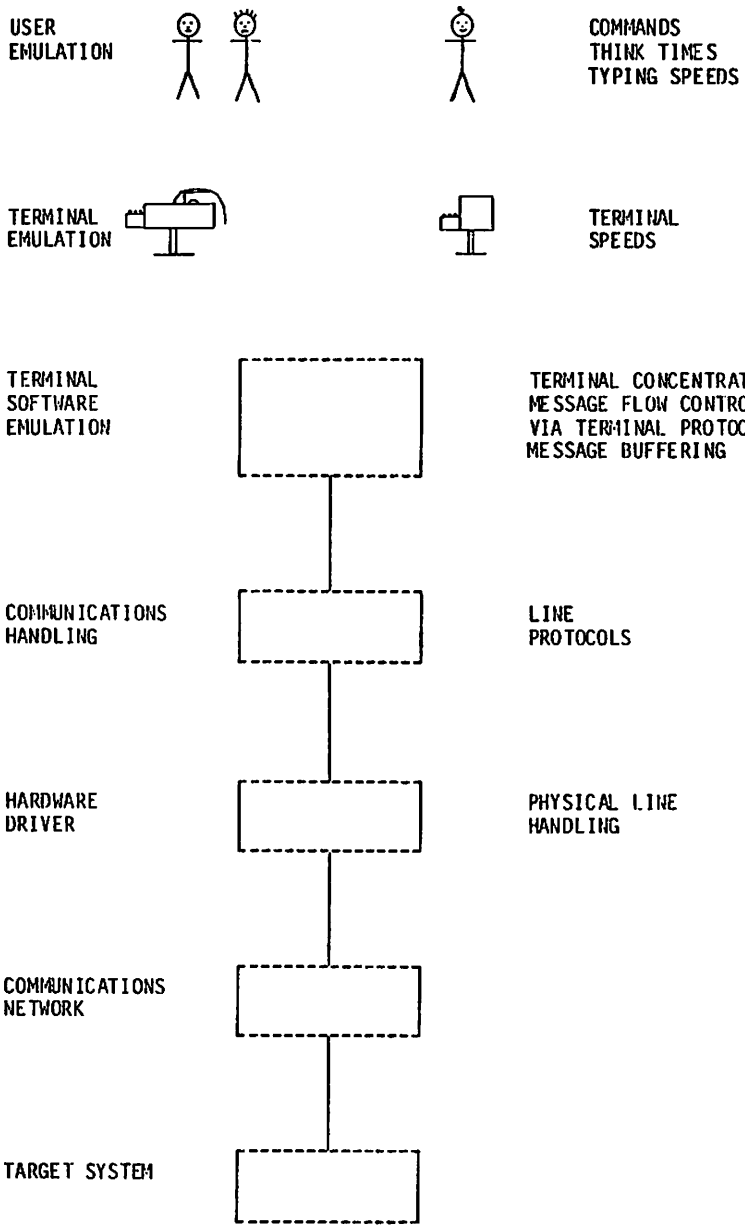
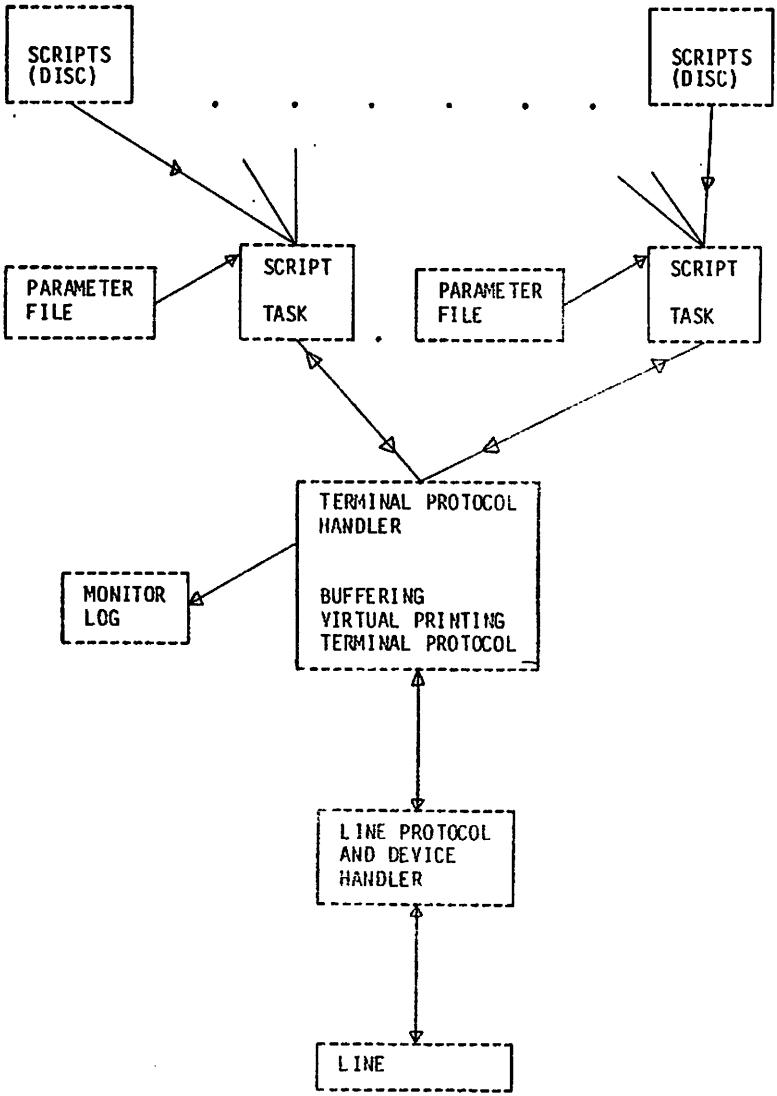


FIGURE 2 : THE SOFTWARE STRUCTURE OF ERTE



REFERENCES

[1] J.C. ADAMS, W.S. CURRIE, & B.A.C. GILMORE, "The ERTE User Manual" Edinburgh University, Computer Science Dept., internal report (1977).

[2] J.C. ADAMS, & G.E. MILLARD, "Performance Measurement on the Edinburgh Multi-Access System" Proceedings of the International Computing Symposium , Antibes, pp105 - 112 (1975).

[3] J.C. ADAMS, "performance measurement and evaluation of Time-Shared, Virtual Memory Systems" Phd Thesis, Edinburgh University - to be published

[4] P.J. DENNING, "The Working Set Model for Program Behaviour" Comm. ACM, 11, pp323 - 333 (1968)

[5] B.A.C. GILMORE, & S.T. HAYES, "The EMAS Front End Processor" ERCC internal report, (1977)

[6] B.A.C. GILMORE, "The DEIMOS User Manual" ERCC internal report (1976)

[7] B.A.C. GILMORE, "Imp as a tool for Small Systems Implementation" M.Phil. Thesis, Edinburgh University (1977)

[8] HDLC Protocol is fully defined in ISO reports ISO/TC97/SC6-731

and ISO/TC97/SC6-1005.

[9] ITP "Interactive Protocol for use in the RCO Ntework" Edinburgh Regional Computing Organisation report No. NP/37.

[10] P.S. ROBERTSON, "The Production of Optimised Code from Portable Compilers" Phd. Thesis, Edinburgh University - to be published

[11] P.S. ROBERTSON, "Experience with the Portable IMP Compiler" Edinburgh University Computer Science Dept. (1977)

[12] J. STASUIK, "Terminal Driver Monitor" University of Michigan, Ann Arbor, Computer Center (1976)

[13] P.D. STEPHENS, "The IMP Language and Compiler" Computer J, 17, pp216 - 223 (1974).

[14] United States National Bureau of Standards, "Survey of Remote Terminal Emulators" special publication 500-4 (1977)

[15] H. WHITFIELD, & A.S. WIGHT, "The Edinburgh Multi-Access System" Computer J, 16, pp331 - 346 (1973).

ACKNOWLEDGEMENTS

We would like to thank Dr. A.S.WIGHT, P.S.ROBERTSON (Edinburgh University Computer Science Department), Dr.

G.BURNS and Dr. A.MCKENDRICK (Edinburgh Regional Computing Centre) for their assistance and advice throughout the development of ERTE.