Construction of a validated simulator for performance prediction of DECnet-based computer networks

Bernd Wolfinger
Fachbereich Informatik, Universität Hamburg

Max Mühlhäuser
Institut für Informatik III, Universität Karlsruhe

Abstract
Predicting important performance parameters of computer networks, recognizing potential bottlenecks, comparing design alternatives are factors of decisive importance in building complex computer networks. In this respect, computer aided simulation has proved to be a very effective design tool in a number of practical applications. It is shown by the example of the MOSAIC modeling system how a simulator applicable on a broad basis could be adapted to the specific characteristics of a class of existing computer networks.
The class of computer networks chosen for modeling is based on DECnet communication software. Modeling concentrates mainly on the DECnet protocols and their hierarchies. The paper indicates the adaptations necessary to adjust the MOSAIC kernel system adequately to DECnet computer networks and summarizes the results of a rather extensive validation for the modeling system, comprising calibration and accuracy establishment, which has been carried out successfully.

Keywords: Simulation, computer network architectures, communication protocols, protocol hierarchies, performance evaluation, modeling systems, validation

1.0 Introduction

The increasing importance of computer networks increases the demand for tools allowing their performance to be studied. An approach repeatedly employed successfully in performance prediction of computer networks is modeling with simulative model evaluation.
A large number of manufacturers already offer communication software permitting the build-up of homogeneous computer networks consisting of computers of that manufacturer (e.g. SNA /SUND80/ by IBM, DNA /WECK80/ by DEC, TRANSDATA /PAWL82/ by SIEMENS, DCA /BERG78/ by Sperry Univac, etc.). In that case, it is particularly desirable to have a modeling tool available which, among others, comprises a set of 'submodels' representing the most relevant computer network components, such as the communication software offered (e.g. submodels for the different communication protocols), the computers used as network nodes, the transmission channels etc. In particular, the modeling tool should permit a complete model of a computer network to be configurated flexibly out of these submodels. Such considerations have been the motivation underlying the development of a correspondingly structured simulator (MOSAIC) at the Institute for Data Processing in Technology of the Karlsruhe Nuclear Research Center in the period 1977 to 1979.
In establishing MOSAIC, common features of computer networks were worked out and transferred into the implementation of a simulator (MOSAIC 'kernel system'). In applying MOSAIC to an existing computer network, the submodels of the kernel system must be matched to the characteristics of the computer network investigated by means of suitable choice of parameters and/or by model extensions. Such adaptations of MOSAIC exist for rather different classes of computer networks (namely for the X.25 based BERNET /WOLF80/,/WOLF82/, for local minicomputer networks based on KINET communication software /DIDI82/, and for computer networks based on DECnet /MUEH80/).

This paper describes the application of MOSAIC to computer networks based on DECnet communication software (referred to as DECnet computer networks below). Besides outlining the characteristics of the MOSAIC kernel system (section 2) and the class of computer networks modeled (section 3), some of the adaptations are indicated which have been necessary to build up a simulator out of the MOSAIC kernel system that can be used to analyze DECnet computer networks (cf. section 4). In addition, the reduction in expenditure resulting from the approach described - i.e. usage of a kernel system - is to be shown in contrast to the usual procedure of directly using a programming/simulation language and starting from scratch. The results of validation indicated (section 5) will underline that, at least for some classes of computer networks, one is able, at comparatively low expenditure, to turn the MOSAIC kernel system into a modeling system behaving very realistically, provided there has been careful calibration.

2.0 The MOSAIC Modeling System

First, the MOSAIC kernel system will be introduced, the description concentrating mainly on the models elaborated to describe computer networks.

2.1 Underlying Concept of a Computer Network

The MOSAIC kernel system is based on the following view of a computer network: A computer network consists of a set of hosts and (possibly intelligent) terminals making data processing services available to users and being interconnected for mutual communication by a communication subsystem. The communication subsystem may consist of point-to-point connections, multi-point connections, or a network of interconnected communication processors. In the latter case, the data are transmitted either directly (line switching) or they are temporarily buffered within the communication processors (store and foreward). The store and foreward technique will be of primary concern for the following descriptions. Hosts, communication processors and terminals are called the computer network nodes. Correspondingly, a computer network can be represented as a set of computer network nodes directly interconnected by transmission lines.

The software used to implement computer network services, such as remote job entry, file transfer, or services to transmit data between two adjacent computer network nodes, is assumed to be structured in layers (cf. ISO reference model for 'Open Systems Interconnection' /ISO 81/). Within the hosts and within the terminals there exist both, layers for data transport oriented services and layers for application oriented services; usually, the communication processors only contain the layers of the data transport oriented services (in the ISO reference model: physical layer, data link layer, and network layer).

In a model concept, each layer within a computer network node - if offering precisely one computer network service - can be viewed to be realized by a protocol unit (PU), which makes the service of this layer available to a protocol unit of the next higher layer ('vertical' communication between so-called neighbours, see fig. 1). The latter may demand the service by means of so-called interface requests. The protocol units of the same layer ('horizontal' communication between so-called correspondents) cooperate in making the service available by exchanging data units (i.e. specific control information of the layer, possibly combined with data units of the next higher layer) in accordance with a given (communication) protocol /POUZ78/. The protocol defines the type and format of the data units and the actions of the protocol units. A protocol unit may have its own buffer to store data units in order to initiate the repetition of a transmission in case of an error or to be able to manipulate data units. Protocol units, if implemented in software, are subject to resource allocation, e.g. with respect to the CPU, by the local operating system (see fig. 1). Within the Application Layer, e.g. in host computers, user tasks exist, which can be network oriented, if they use computer network services, or which have only local character otherwise.

2.2 Purposes of Performance Analyses by MOSAIC

Quantitative performance analyses, in general terms, serve to give insight about the dependencies between values of performance characteristics and design variables and, on that basis, take measures in order to achieve 'optimum' (with regard to a chosen performance criterion) system behaviour for different load profiles. Important performance characteristics of computer networks and their components (transmission lines, computer network nodes, protocol layers, protocol units, etc.), the study of which was to be made possible by MOSAIC, are throughput of data units or user tasks per unit of time, delays of data units or user tasks, and utilization of computer network components.

The design variables should refer to the following aspects, among others:
- topological aspects: number of communication processors, structure of the meshed communication subsystem,
- mapping of logical transmission links to physical transmission links,
- characteristics of transmission lines,
- characteristics of computer network nodes and their operating systems (in particular task management),
- methods of flow control and error control,
- size of the buffers assigned to the protocol units,
- parameters of protocols (e.g., window size, maximum length of data units).

Unlike the modeling tools with comparable objectives made available so far, MOSAIC in particular had to support precise studies of protocols and their interactions in protocol hierarchies (for this extremely important requirement, see, for instance, /GERL80/).

2.3 Underlying (simulative) Models

In (computer aided) simulation, the computer network to be modeled is mapped onto a corresponding program, the simulator. The execution of the program, the experiment, reflects the dynamics of the real system. The program must contain facilities for measuring performance characteristics in the course of an experiment. In general, each experiment furnishes only one specific random sample for each performance characteristic considered. As a consequence, extensive series of experiments are usually necessary to derive a formal relationship between performance characteristics and design variables.

The first step in the development of a simulator for studying the performance of computer networks consists of the formulation of an overall model for the computer network to be examined. To formulate the overall model, suitable submodels must be elaborated for the components of a computer network listed in 2.1. In order to be able to access by simulation the relationships between performance characteristics and design variables (as demanded in 2.2), especially detailed modeling of the protocol hierarchy and of resource allocation within a computer network is required (see fig. 1). In this paper, only a rough description of the models can be provided; for more details the reader may consult /WOLF79/, where, in particular, the queueing models used to describe the behaviour of communicating computers (including their communication software) can be found.

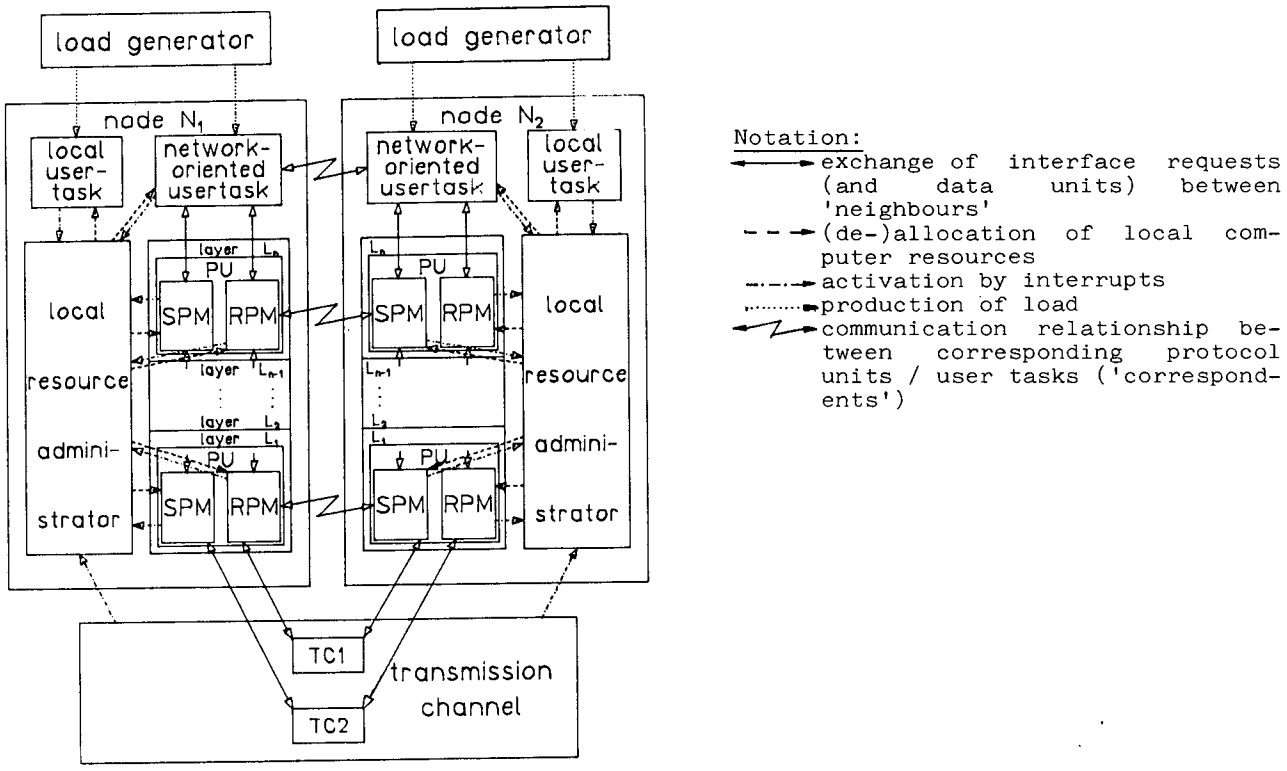

Fig. 1: Interactions between computer network components exemplified by two communicating network nodes N_1 and N_2.

One of the most important aspects of modeling a protocol hierarchy are the models of protocol units. The main criterion we had to satisfy in developing models of the protocol units was elaborating common features of protocol units and the need to introduce a basic model with properties common to several or even all protocol units of a protocol hierarchy. Depending on the scope of service-specific properties of a protocol unit, the basic model must be expanded in a concrete case of application. The basic model of a protocol unit includes introduction of a layer-independent queueing system for protocol units, definition of typical interface requests which can be used for a large number of communication protocols, and establishment of certain algorithms for handling typical interface requests by a protocol unit.

The queueing system for modeling a protocol unit is based on two service stations, the sending protocol module, and the receiving protocol module (see fig. 1). The sending protocol module is responsible for all actions of a protocol unit which are concerned with the sending of data units; the receiving protocol module is responsible for all actions connected to the receipt of data units. The flow of requests (by exchange of interface requests) and the data flow (by exchange of data units) are separated from each other. The actions of the sending and receiving protocol modules consist in analyzing and handling interface requests; the handling of an interface request may entail manipulation of data units. The pattern of a protocol unit's actions are predefined by means of sequential automata associated with the two service stations (sending and receiving protocol module).

The structure of the data units is defined recursively, a (L_i-) data unit at the interface between layers L_i and L_{i-1} being composed of control information of layer L_i possibly combined with data units of higher layers. Control information, among other purposes, serves for addressing and mutually exchanging information among correspondents to realize a communication protocol.

For modeling the local resource allocation to protocol modules and user tasks within a computer network node there is the local resource administrator. Its main function is to organize the allocation of the resources CPU and main memory. The local resource administrator thus consists of a service station and several queues, the elements of which are protocol modules and user tasks. Protocol modules, especially those of the lower protocol layers, generally are permanently resident in the main memory and therefore have to compete only for the CPU.

As a model of the transmission line (transmission channel) we select a queueing system which, see fig. 1, consists of two service stations, TC1 and TC2. In order to be able to take into account the mode of operation of the transmission channel (simplex, half duplex, full duplex), interactions between the two service stations must be considered. If the mode of operation is full duplex, TC1 and TC2 work independently of each other, whereas, in the half duplex mode, TC1 and TC2 must be synchronized, because the two service stations cannot be active at the same time; in the simplex mode, either TC1 or TC2 will be omitted (with the corresponding queues for interface requests and data units, not represented by fig. 1).

The MOSAIC simulator was implemented based on the submodels of a computer network, elaborated above.

2.4 The Kernel System Implemented

In the following the implemented kernel system of MOSAIC is described, which allows a complete model of a computer network to be built out of the set of submodels introduced and the behaviour of the complete model to be studied under various boundary conditions, e.g. under the impact of a given load.

A number of important requirements were taken into account in establishing the MOSAIC kernel system:
(1) Applicability of the simulator under various boundary conditions (characteristics of computer network, load profiles, etc).
(2) Possibility to use the simulator for studies involving different degrees of detail.
(3) Easy expandibility of the simulator.
(4) User oriented representation of the experimental results by means of a graphic display.
(5) Efficient experimentation by supporting interactive simulation experiments.
(6) High portability of the programming language(s) used for implementation.
Especially to satisfy requirements (1) - (4), the concept of a modeling system has been elaborated for MOSAIC, which is based on the idea of making available, as building blocks, the submodels required to simulate communication flows in computer networks, and configu-

rate an executable simulation program out of these building blocks at the beginning of an experiment. This method permits mutually independent definition and problem-free exchanges of submodels. The following main components of the MOSAIC modeling system can be distinguished:
- The building blocks representing a conversion of the submodels presented into the programming language used.
- A configurator which, at the beginning of a simulation experiment, establishes the complete model as an executable program out of the building blocks, taking into account the input data.
- The load generator generating requests of the computer network environment and acting upon the computer network model during a simulation experiment.
- The measurement facilities, which serve to measure model variables in the course of a simulation experiment (e.g., performance characteristics) and store the data collected for subsequent evaluation.
- Facilities for data evaluation, which include dedicated components both for statistical evaluation of the measured data and for graphic processing of the experimental results.

Aside from the 'facilities for graphic processing (of results)', which were implemented in FORTRAN, the language selected in implementing the modeling system was the high level language SIMULA (cf. /DAHL68/). SIMULA is characterized by a modern language concept, excellent portability and a significantly higher efficiency than languages such as GPSS (if the CPU time required for executing equivalent programs is used as a criterion of comparison).
The MOSIN user interface /MUEL80/ - being independent of MOSAIC - serves to establish the input data files for simulation experiments required by the configurator, which, in general, are rather extensive. MOSIN is based on a command language and permits experiments using graphic storage or raster displays to have graphic definitions of the protocol units and, in particular, to establish all communication links between the communication partners in a graphic format.

The configurator needs information concerning the
- basic description of the computer network model, such as the number of different types of computer network nodes; for each type: the number of nodes for this type existing in the computer network and the number of protocol layers included in each node;
- description of the general structure of the protocol hierarchy, e.g., for each type of computer network node: number of services offered in parallel for each protocol layer;
- description of each computer network node, such as processing speed of the CPU; times used for interrupt processing; buffer available for communication purposes; characterization of computer specific load (alternatively user tasks and/or interface requests can be generated);
- description of each protocol layer, such as maximum length of data units, probability of a (detected) error in transmission of a data unit within this layer;
- description of all
  a) protocol units, such as
     number characterizing the communication protocol realized by the protocol unit; indication whether multiplexing is necessary, whether, in case of an asymmetrical protocol, the protocol unit is a 'master' or a 'slave', whether, for instance in case of a protocol such as X.25, the protocol only knows data exchange phase or also an initialization and termination phase; buffer size assigned to store data units;
  b) sending protocol modules, such as
     characterization of neighbour and correspondent relations; execution times of communication software (depending on the type of interface requests to be handled); the need to carry out time monitoring, block formation or fragmentation; maximum size of the 'sending window', if a window-mechanism determines flow control;
  c) receiving protocol modules, such as
     characterization of neighbour relations, of execution times of communication software (depending on request type, cf. sending protocol module); maximum size of the 'receiving window', which can be used to control the acknowledgement traffic;
- characterization of the computer network topology by indicating all computer network nodes directly interconnected;
- description of all transmission channels, such as mode of operation; transmission capacity; propagation delay; maximum length of data units accepted; probability of transmission errors occuring;
- description of parameters to control the simulation experiment, such as run-length; for batch-experiments: degree of detail for the trace demanded and control of data acquisition and data evaluation.

The measurement facilities of MOSAIC, among other features, cover
- the utilization of all computer network components (protocol units / protocol modules, computer network nodes, transmission channels, main memory, communication-oriented buffers,...);
- the delays suffered by data units within computer network components (layers, protocol

units / protocol modules);
- the data throughput (for all layers, for the entire computer network);
- the response times measured on the Application Layer;
- the number of user tasks, data units generated within the different components of the computer network;
- the queue-sizes of the protocol units, the local resource administrators, the transmission channels;
  etc.

The graphic aids /WOLF79/ integrated in MOSAIC relate to plots of the dynamic behaviour of performance characteristics (throughput, delays, etc.) and to representations of queue-sizes within the protocol hierarchy; in interactive execution of an experiment, all these graphic results can be gained at each of the interrupting points selected.

3.0 DECnet Computer Networks

The class of computer networks to be modeled is described based on DECnet Phase II version, which was available at Karlsruhe University and therefore served as a starting point for the first version of the MOSAIC(DECNET) simulator.

A course description of the DECnet communication software is given in the following. Digital Equipment Corp. supports the establishment of (homogeneous) computer networks by a software package sold under the trade name DECnet. Application of DECnet is supported for various operating systems on several computer families. The basis for implementation of DECnet was the Digital Network Architecture (DNA), in which the structure and the mode of operation of the communication protocols is described (details on DNA can be found in /WECK80/). In DNA, the communication software can be considered as logically sub-divided into six layers as represented in fig. 2.

Studies of DECnet indicated that implementation of this layered structure is characterized by the following pecularities:
The top layer (Application Layer) represents the set of network-oriented user tasks; in case of remote file access, user tasks pass on interface requests to the NAL; in inter-task communication, direct access is made to NSL.
The software of the NAL (executed as a set of subroutines) is not resident in main memory, as is customary with protocol modules, but bound to user tasks as required (sectionwise); thus, user tasks and DAP-components compete for resources as one logical unit.
For certain types of transmission channels, the DDCMP is left out, i.e., interface requests from NSP are transmitted directly to the channel driver (DDM).

The Transport Layer has not yet been a component of Phase II DECnet.

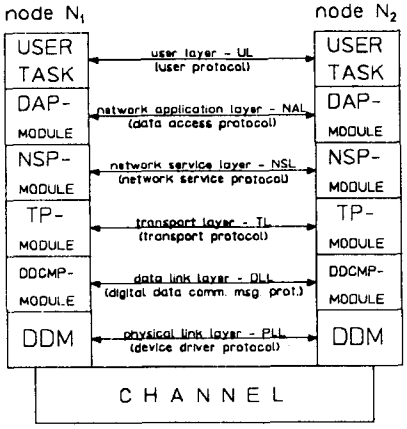

Fig. 2: Layered structure in a DNA two node network

To support the buffer management, three types of buffers are used to store data and requests:
Communication control block generally describe interface requests and may include pointers to the respective data units, but may also themselves contain data units or parts of data units (e.g., in DDCMP).
Small data buffers preferably are used to store control information exchanged between NSP

protocol units.
Large data buffers are used in the Network Service Layer and lower layers primarily to store user data (data units of the user layer). Since the length of large data buffers cannot be dynamically changed, user data which are too large must be fragmented.
On the receiver side, the large data buffers are used throughout for all types of data units (they are then called 'receive data buffers').

4.0 Adaptation to DECnet of the MOSAIC Modeling System

In adapting the MOSAIC kernel system to the characteristics of DECnet computer networks, the preparatory work done in implementing the kernel system was found to be extremely helpful. On one hand, the large number of types of requests and data units and functions (fragmentation, time control of critical events etc.) could directly be adopted or easily expanded; on the other hand, the flexibility of the system permitted adaptation to characteristics specific to DECnet (dedicated data buffers, 'reply' mechanism of DDCMP, etc.). As was to be expected, adaptations and modifications largely involved the submodels of the protocol units and, to a lesser extent, the local resource administrator. Since the other main components of the modeling system could be largely unchanged, compatibility with other MOSAIC versions has been preserved to a large extent. Therefore, for instance, it is possible, without excessive expenditure to integrate those protocol units implemented in other versions (e.g., HDLC, /WOLF80/).
A detailed description of the mapping of DECnet protocols on MOSAIC protocol units and information about the submodels implemented for MOSAIC(DECNET) can be found in /MUEH80/.
The input and output interface of MOSAIC had to be changed only slightly:
The input interface was expanded by optional 'DECnet data' reflecting characteristics specific to the DEC products. Their scope was limited to one additional input card per computer network node.
In the output interface it was attempted to bear in mind the intention to map, as precisely as possible, the main memory management function. Thus, the indications of buffer allocations in computer network nodes were expanded by detailed information about the type of buffers allocated.

The adaptation of the MOSAIC modeling system to computer networks equipped with the DECnet Phase II communication software has provided a tool with extensive possibilities of application. Simulation of communication flows is possible for a broad spectrum of user classes (demanding NSP or DAP services). The modeling system supports arbitrary types of topologies; limitations could arise only from the available memory of the computer on which the simulation experiments are being run. Different network characteristics and variations of protocol hierarchies are covered to the extent in which they are meaningful for DECnet computer networks. Detailed modeling of the buffer management significantly supports the accuracy of experimental results. This opens up a spectrum of applications, among which experiments for protocol studies, analyses of protocol hierarchies or optimizations of buffer management represents only a small part.

One major characteristic of the MOSAIC modeling system is its expandability. There are no basic difficulties to be expected in adapting MOSAIC to Phase III DECnet (presently in preparation), integrating protocols (X.25 etc.) from other MOSAIC versions into MOSAIC(DECNET), or integrating submodels of alternative communication systems (e.g., ETHERNET). Run times required on a IBM370/168 to simulate medium-sized DECnet configurations are in general shorter than the real time intervals modeled.

5.0 Validation

After implementation of a modeling system, model verification and validation have to take place (see /CHAT77/, /EFRO75/, /SCHW80/, /TEOR75/). Verification serves to indicate the correctness of implementation with respect to model specification. For the simulator to be able to match reality with sufficient precision and thus enable relevant information to be obtained about real systems' behaviour, a validation phase must follow the verification phase. Validation is organized in two steps:

(1) The calibration phase serves for more precise adaptation of the simulator (e.g.,submodels) to reality, with particular emphasis on the dynamic model behaviour. For this purpose, values must be assigned mainly to those model parameters which are still unknown (e.g., use of data obtained by measurements in an existing real system). At this point in time, changes in the simulator may again become necessary which possibly imply a restart of the verification phase.
(2) The phase of accuracy establishment serves for conclusive comparison between the behaviour of the simulator and that of the real system modeled. Quantitative estimates of the differences in behaviour of the simulator and the real system(s) define the applicability and the precision of the simulator.

Fig. 3 describes these relationships.

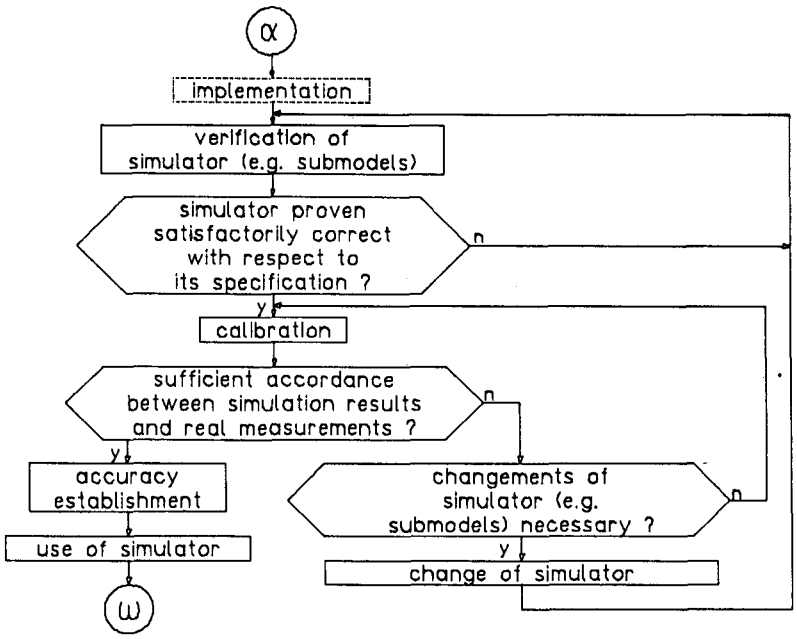

Fig. 3: Simplified strategy for simulator verification and validation

Verification was supported greatly by the test aids made available in MOSAIC (e.g., trace procedures, procedures for specific testing of protocol states etc.) and by the modular program structure. Since the verification phase involved no fundamental problems, it will not be described here in more detail.

The proceeding adopted in validation corresponds to a common strategy:
During the calibration phase, measurements were obtained from an existing DECnet configuration so as to be able to define missing model parameters. Some first simplified intercomparison experiments helped in matching the simulator to the behaviour of the real systems considered. The phase of accuracy establishment was based on a set of new experiments for characterizing the behaviour of the DECnet configuration available, which experiments differed sufficiently from those used for calibration.

5.1 Calibration

The model parameters with values to be assigned in the calibration phase, as mentioned above, mainly relate to the model behaviour over time. The approach used in MOSAIC in describing the time behaviour of the communication software consists of providing processing times for interface requests; the approach pursued, first of all, permits different service (processing) times to be used for each computer and each layer. (The service time of an interface request results from the CPU time required to run that part of the communication software involved in handling the request). Moreover, MOSAIC distinguishes constant and length-dependent service times, the latter taking into account the lengths of data units possibly involved in request processing (e.g., necessary if data is moved in buffer). Since service times for sending and receiving activities are distinguished as well as handling of user data and control information, respectively, eight different service time parameters result per computer and protocol unit. Even this major differentiation was not sufficient for completely assessing DECnet service times because, for instance, the times occurring in connection with NSP control information must be differentiated further by the type of control information involved. However, it was found in calibration that, for instance, the service times needed to process different NSP control information shows linear dependencies, irrespective of the computer characteristics; accordingly, the further differentiation of service times necessary for MOSAIC(DECNET) could be taken into account in factors incorporated in the program, thus obviating the need for additional expansion of the input interface.

For collecting the measured data from the real system, a measuring program for calibration ('CAL') was developed, which performs measurements with an accuracy of approximately 10

µsec by means of a programmable realtime clock. After integration into the communication software of the computer to be measured, CAL is ready for execution without needing any modifications to be made in the operating system or the communication software. After a call of CAL, the boundary conditions of the measurements are defined interactively (type of request to be measured, lengths of data units possibly involved, transmission channel used, etc.). A task communicating with CAL must be installed on the target computer.
The results are output in a preprocessed format (converted, split up with respect to the protocol units, etc.), containing additional information, for instance, about local connections or omission of DDCMP.
Since the measurements should be carried out without major interference in the communication software, the measuring subroutines for direct measurements on the system had to be designed in such a way as to be loadable into the communication software at runtime. For this purpose, the following scheme was applied: let - as an example - the objective of a measurement be the generation of a service time value required in a layer L_i, to process a specific interface request. In that case, transferring an interface request from a layer L_{i+1} to layer L_i and processing it within L_i may be regarded, in a simplified view of the programming technique, as a subroutine call (in actual fact, the technique used is somewhat more complicated). In our example, the code of a measuring subroutine is brought into the code of the communication software pertaining to L_i. If measurements are to be carried out, the 'subroutine call' from L_{i+1} to L_i is deflected in such a way that L_{i+1} first activates the measuring subroutine. In the measuring subroutine, a time measurement is carried out, directly followed by a branching to L_i. The 'return branch' from L_i leads back to the measuring subroutine, where another time measurement is carried out before control is passed back to L_{i+1}. The measured data stored in the measuring subroutine are read by CAL in a time-uncritical way, their difference indicating the service time required by L_i. The additional delay produced by the measuring procedure proper is below the measuring accuracy of the real-time clock. Fig. 4 again summarizes the introduction of a measuring subroutine.

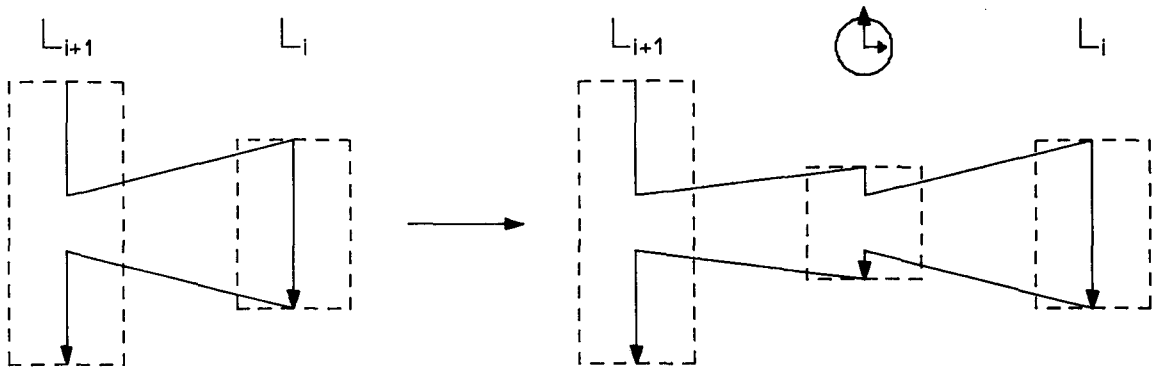

Fig. 4: Embedding of measuring programs into communication software (schematic representation)

The detailed description of the measurement facilities can be found in /MUEH80/. A small subset of the results obtained is shown in table 1. The results refer to the underlying DECnet configuration at Karlsruhe University. For the measurements, an 'unloaded' situation was established in both computers involved, i.e., no other load was assumed to exist in the computer network besides the one request to be processed (and its implications for the network), which was to be measured.
Because of the deterministic load, the results are of non-stochastic nature, and so a statistical evaluation of the measurements is not required.

Service times: NSP
(µsec)

PDP	11 / 45	11 / 40	11 / 34
CSVTSDU	840	1045	1100
BSVTSDU	0.1875	0.2250	0.2375
CSVTSCI	277	332	343
BSVTSCI	0.15	0.18	0.19

multipliers for different types of control information:

connect initiate: 21.08 * CSVTSCI

connect confirm: 11.85 * CSVTSCI

Service times: DDCMP
(µsec)

PDP	11 / 45	11 / 40	11 / 34
CSVTRDU	970	1470	1550
BSVTRDU	4.2625	5.2820	6.6125
CSVTRCI	850	1340	1455
BSVTRCI	4.2625	5.2820	6.6125

Table 1: Some of the service times for NSP and DDCMP in dependence of the type of machine used
Notation: C: constant - B: length-dependent (per bit);
S: sending activity - R: receiving activity;
DU: data unit - CI: control information;
e.g., CSVTSCI = constant servicetime required for sending control information

5.2 Accuracy Establishment

The phase of accuracy establishment, first of all, required very precise criteria to be provided with respect to the way in which to run the intercomparison experiments so as to obtain really comparable results and data about the behaviour of the simulator and the real system, respectively.
Such criteria to be provided, for instance, related to the configuration of the experiment, the CPU requirements of user tasks, the request sequence of the data exchange phase in the highest layer (assuming intertask communication), etc. In the modeling system, these criteria could largely be controlled directly by way of the input data set. In the real system, dedicated user programs and measurement facilities had to be created for execution of the experiment. The resulting program ('VAL') was assigned an input interface permitting partly interactive definition of experimental boundary conditions. This allowed a sufficiently broad range of experiments to be conducted so that two VAL results could easily be used for calibration (see 5.1 above). Moreover, it was ascertained that all new experiments for accuracy establishment were complex and interrelated so as not to be considered simple sequential series of isolated processing steps measured in calibration. For instance, in intertask communication during the data exchange phase it was no longer attempted to have the individual requests processed in an isolated way; instead, overlapping of several activities on the sending and the receiving sides was ensured. As an example, precise observation of one experimental run, among other facts, has indicated that, in the simulation experiment, two data units less were transmitted via the transmission channel in order to handle a set of requests than would have been necessary in isolated processing of the respective requests. The number and types of data units were found to be exactly the same as observed in the real system (measured by 'VAL'). Against this background, the results of the intercomparison experiments described briefly in table 2 become even more significant. For the comparisons the following boundary conditions were chosen:
- user tasks running in the same or in different computers and communicating by means of the intertask communication facilities provided by the Network Service Layer.
- a standard communication sequence (set-up of a logical link, exchange of user data,

deletion of the link).
- constant length of user data exchanged during a communication sequence (e.g. 8, 800, 1600 Bit in the experiments reflected by table 2).

The measurements of the intercomparison experiments refer to the time interval required to execute the total communication sequence.
Relative deviations in most cases were below 1%, in all the experiments performed they were below 5%. Comparison with corresponding work in the modeling field underlines the quality of the simulator presented. The degree of conformity achieved between model and reality (in the area of communication software) can be explained by the quality of the modeling concept and the care exercised in model formulation and calibration.

direction of data transfer:	data unit length (bit)	real measure-ments (sec)	results of simulation experiments (sec)	relative deviation in %
PDP45 to 11/45 using an 'internal link'	8	18.158	18.154	0.022
	1600	19.358	19.350	0.041
PDP11/45 to 11/34 using a DA-channel (omission of DDCMP)	8	19.60	19.77	0.13
	1600	23.24	23.23	0.04
PDP11/45 to 11/45 using a DL-channel (with DDCMP, 9600 Baud asynchr.)	8	218.75	215.10	1.67
	800	453.6	455.6	0.44

Table 2: Results of intercomparison experiments for accuracy establishment

It has to be mentioned that the accuracy establishment was only performed for the model of DECnet communication software, i.e. of the layers up to the Network Application Layer (NAL). The behaviour of user tasks as seen by the communication software is the same in the real system as in the model, taken into account that we mapped the corresponding DECnet-interfaces one-to-one into the model. So, for our purpose, it was possible to use deterministic load for the real measurements and for the simulator during accuracy establishment of the model of DECnet layers. One should notice that even with deterministic load it is possible to create highly complex situations for the communicating software.
In order to delimit the results of the validation carried out, it should be pointed out that the expenditure involved in obtaining measured values from the real system did not make it meaningful, in connection with the imminent introduction of Phase III DECnet, to carry out comprehensive validation. So, especially during the data exchange phase of the intertask communication, no 'transmit interrupt' requests were admitted, the phases of connection setup and cleardown were not validated in detail and, moreover, more complex load profiles could be imagined. Nevertheless, it can be assumed in the light of validation results of the accuracy attained (see table 2) that the modeling system MOSAIC(DECNET) will be fit for a broad range of applications.

6.0 Conclusions

The MOSAIC modeling system and some of the adaptations necessary to describe the specific properties of DECnet computer networks have been presented. The modeling system emphasizes the detailed mapping of communication protocols and even of complete protocol hierarchies. An outstanding feature of MOSAIC(DECNET) is its large flexibility offered for modeling different network configurations including structural variations (even concerning the internal structure of the protocol hierarchy). A user-friendly interactive, graphical interface is provided to facilitate experimentation. Special emphasis has been placed on careful validation. For this purpose dedicated monitors have been implemented and introduced into the existing DECnet configuration available. The validation results prove that a high degree of accuracy can be still achieved with a generalized modeling system.
Our current research activities comprise the modeling of DECnet Phase III, the embedding of the modeling system into a planning system for (semi-)autonomous network optimization, comparison of network performance evaluation methods and distributed performance evaluation.

7.0 References

/BERG78/ Berglund, R.G.
Comparing Network Architectures.
DATAMATION, Feb. 1978, pp. 79 - 85

/CHAT77/ Chattergy, R., Pooch, U.W.
Integrated Design And Verification of Simulation Programs. Computer, No. 4, 1977, pp. 40 - 45

/DAHL68/ Dahl, O.J., Myhrhaug, B., Nygaard, K.
SIMULA67- Common Base Language.
Norsk Regnesentral; Oslo, 1978

/DIDI82/ Didic, M., Wolfinger, B.
Simulation of a Local Computer Network Architecture Applying a Unified Modeling System.
Computer Networks, Vol.6, 1982

/EFRO75/ Efron, D.C.
A Methodology for Tuning and Verifying Package Simulation Models.
H.J. Highland (ed.): Proc. Symp. on the Simulation of Computer Systems; Boulder, 1975, pp. 160 - 173

/GERL80/ Gerla, M., Kleinrock, L.
Flow Control: a Comparative Survey.
IEEE Transactions on Communications
Vol.28, No.4, 1980, pp. 553 - 574

/ISO 81/ International Organisation for Standardization
ISO/TC97/SC16: Data Processing - Open Systems Interconnection - Basic Reference Model.
Computer Networks, Vol. 5, Nr. 2, 1981, pp. 81 - 118

/MUEH80/ Mühlhäuser, M.
MOSAIC(DECNET) - Ein Modellierungssystem zur Leistungsanalyse für DECnet - Rechnernetze durch rechnergestützte Simulation.
Diplomarbeit, Karlsruhe 1980 (in German)

/MUEL80/ Müller, T.
Interaktives graphisches Interface für ein Modellierungssystem zur Simulation von Kommunikationsflüssen in Rechnernetzen.
Diplomarbeit, Karlsruhe 1980 (in German)

/PAWL82/ Pawlita, P.F., Strack-Zimmermann, H.W.
Public Services and the Transdata Network Architecture.
Proc. Sixth International Conference on Computer Communication, London 1982, pp. 621 - 626

/POUZ75/ Pouzin, L., Zimmermann, H.
A Tutorial on Protocols.
Proc. IEEE Vol.66, No.11, 1978

/SCHW80/ Schwetman, H.D.
Validating System Models: A Case Study.
Computer Science Dep. Purdue University, West Lafayette, report CSD-TR 333, 1980

/SUND80/ Sundstrom, R.J., Schultz, G.D.
SNA's first six years: 1974 - 1980
Proc. Fifth International Conference on Computer Communication, Atlanta 1980, pp. 578 - 585

/TEOR75/ Teorey, T.J.
Validation Criteria for Computer System Simulations.
H.J. Highland (ed.): Proc. Symp. on the Simulation of Computer Systems; Boulder, 1975, pp. 160 - 173

/WECK80/ Wecker, S.
DNA: The Digital Network Architecture.
IEEE Transactions on Communications
Vol. Com.-28, No. 4, 1980

/WOLF79/ Wolfinger, B.
Modelle zur rechnergestützten Simulation von
Kommunikationsflüssen in Rechnernetzen.
Dissertation, Karlsruhe 1979 (in German)

/WOLF80/ Wolfinger, B.
MOSAIC(BERNET): Ein Modellierungssytem zur Leistungsprognose
und -analyse für das BERNET-Rechnernetz.
Kernforschungszentrum Karlsruhe, KfK-Bericht Nr. 3024
Karlsruhe, 1980 (in German)

/WOLF82/ Wolfinger, B., Drobnik, O.
Simulation of Protocol Layers of Communication in Computer Networks.
S. Schoemaker (ed.): Computer Networks and Simulation II
North-Holland Publ. Comp., Amsterdam 1982, pp. 141 - 165