COMPEURO 89
VLSI and Computer Peripherals
Hamburg, May 1989

PERFORMANCE ANALYSIS OF PROTOCOLS

Bernhard H. Walke
Dataprocessing Techniques
Department of Electrical and Electronics Engineering
Fern University of Hagen
D-5860 Iserlohn, West Germany

ABSTRACT: "Performance Analysis of protocols" is interpreted in this paper more widely as usual, to not only cover the aspect of a protocol's traffic behaviour but also such important areas as consistency of a specification, correctness of an implementation, and conformance to an existent standard. An overview is given on the various techniques being currently in use to formally specify protocols and prove some properties of the specification, to implement a formally specified protocol and verify the implementation, to analyze an implementation with respect to its dynamic traffic behaviour by simulation, to apply analytic modeling techniques to a protocol to compute performance figures like delay and throughput and to perform performance tests to verify the conformance to a given standard. Besides making developers of protocols aware of the various techniques and tools available today one goal of this paper is to collect and present basic judgements on the power and limits of methods and tools currently available in the area of protocol development.

1. INTRODUCTION

The usual approach to the solution of complex problems involves dividing the overall problem into a number of subproblems which can be solved separately. This decomposition technique is also utilized when designing data communication networks which are so complex that they are partitioned into a number of separate subsystems that interact with each other in a well-defined manner in order to perform the overall task.

This division is done in a hierarchical or top-down fashion, starting with an abstract specification for the system. At the top level of the hierarchy, the system description is decomposed into a set of interacting modules. A module at one level is divided into several interacting submodules at a level below. This refinement is repeated until the tasks performed by the submodules can be easily understood and implemented.

In terms of existent standards of the International Organization for Standardization (ISO) a communication network is specified hierarchically by an abstract layered architecture as defined for Open Systems Interconnection (OSI) /1/, where the interacting modules are open systems, each containing a set of submodules called layers. Each layer contains sub-submodules called service entities, which communicate on a peer-to-peer basis using protocols. A layer, say (N-1), provides a (N-1)-service at its upper layer interface by exchanging (N)-protocol data units (PDUs) with some service entities of layer (N), and uses the (N-2)-service at its lower layer interface via the (N-1)-protocol, c.f. Fig. 1. A (N)-protocol comprises a set of well-defined data units which might be used by a (N)-service entity of a system A to communicate with its peer (N)-service entity in system B.

A (N)-service entity is responsible to provide partly or completely the service of layer (N) by carefully following the service specification which specifies, what reactions to events at the upper and lower layer interfaces, respectively, are permitted, dependent on the previous history represented by the current state of the entity. So, a protocol can be defined as the vehicle to carry the information needed to co-ordinate

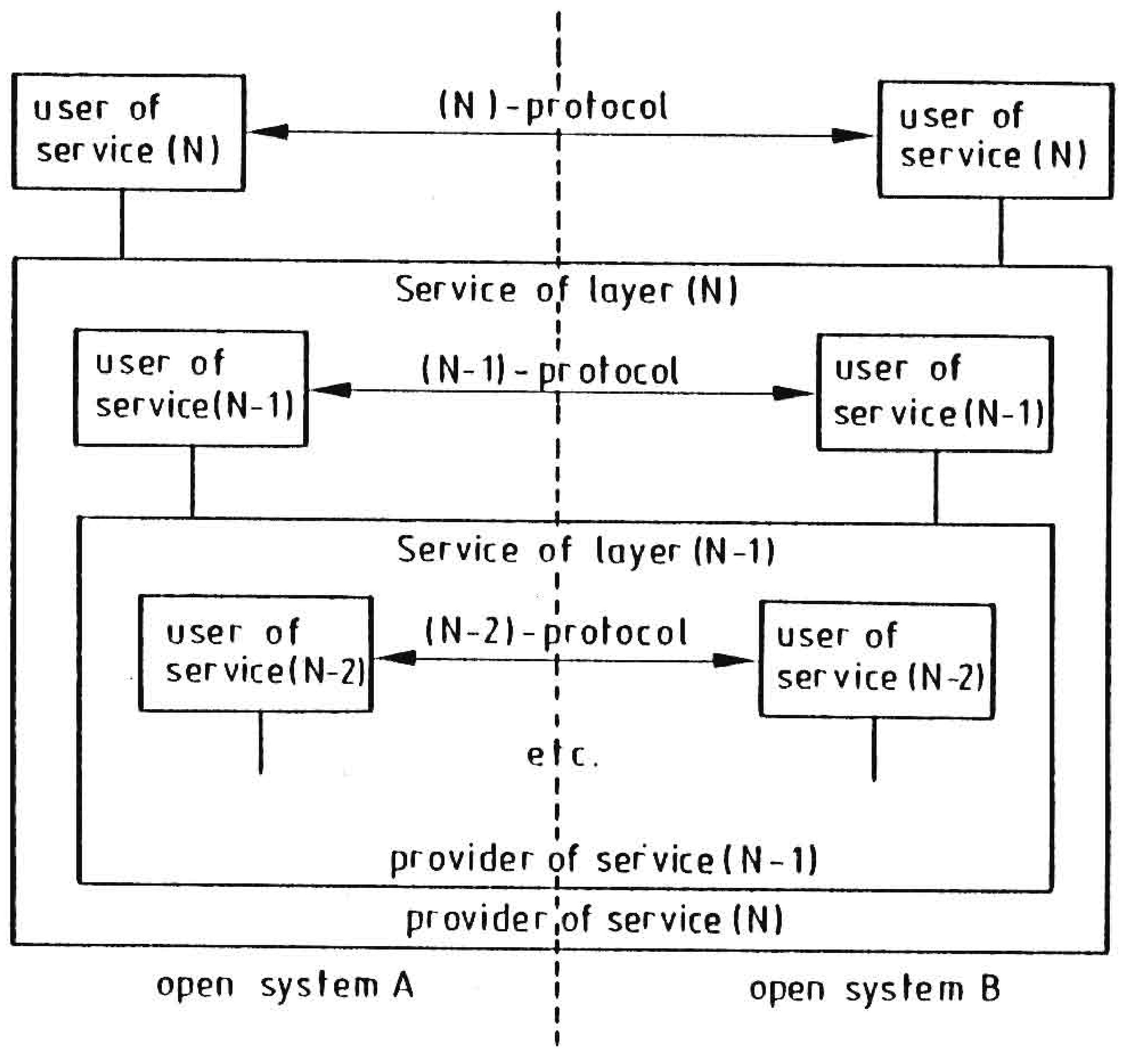

Fig. 1: Reference Model defining layers as service providers and protocols to communicate between entities of the same layer but of different systems

the internal state transitions of two communicating entities. Each entity is represented by a process running in the respective system, and the protocol is used to synchronize the execution of two or more processes running in parallel in a distributed system. Synchronization is performed by exchange of PDUs issued by the interworking processes.

The growing concern regarding protocols is demonstrated by the recent flood of publications in this area. The first IFIP TC.6 sponsored international conference, dedicated specifically to protocols took place in 1978 in Liege /48/. The IEEE Transactions on Communication devoted a special issue to Computer Network Architectures and Protocols in April 1980. Since then, numerous workshops and international conferences were held to bring key researchers in the protocol area together.

Computer communication systems tend to become very complex. To manage that complexity, the systems have to be partitioned and, thereby, an increased use of protocols is required. Although protocols cover only parts of a system, experience has shown that correct protocols are difficult to design. This lead to the use of more formal specification techniques: finite-state machines, Petri-nets, formal grammars, and high level programming languages to describe protocols more formally and precise. Such techniques support an automatic machine-execution of a formal protocol definition and are enormous helpful in automating the various tools for protocol design, analysis, and implementation, which we are going to discuss in the subsequent paragraphs of this paper.

2. STATE TRANSITION MODELS

State transition models are often used to describe certain aspects of communication protocols. We consider protocols in which nodes (modeled as finite state machines) communicate by exchange of messages, which are subject to unpredictable and unbounded delays. Such protocols often admit a large number of messages (PDUs) simultaneously in transit. Understanding a protocol involves understanding channel contents, i.e. contents of messages that might be in transit at one time.

A popular model for these protocols consists of finite state machines connected by unbounded FIFO queues, which is commonly referred to as CFSM model (communicating finite state machine model) /2/. Various universal (syntactic) properties of a protocol become reachability properties in the CFSM model.

For example a CFSM protocol is free from state deadlocks, if from a given initial global state it cannot reach a global state with all channel queues empty and all state machines in states that allow only receptions but not transmissions /3/. A global state is defined as a pair (S,C) where S represents a composite state of the CFSM and C defines the channel content. A composite state consists of a vector of states, containing one state per FSM being involved as part of the CFSM model of the protocol. Freedom of deadlocks can be proven by a reachability analysis which also can demonstrate other properties like the absence of unspecified receptions /4/.

If the global state space is finite, which is the case for a limited permitted number of messages in the channel queues, an exhaustive reachability analysis is useful to automatically verify the reachability properties of the CFSM model. According to /2/ no algorithm exists to verify deadlock-freedom (and other reachability properties like unspecified receptions) when the channel queues are unbounded. But common protocols do not make use of the full generality of the CFSM model and this is the reason, why automatic verifiability of deadlock-freedom is promising. As most common protocols have finitely many reachable global states, one can generate them all and verify any reachability property by inspection. The list of these states is helpful to understand the overall functioning of a protocol, because it allows to examine all channel contents that may occur during the operation of the protocol. In doing so one is able to get a first insight into the performance of a protocol.

2.1 IMPORTANT PROPERTIES OF A PROTOCOL

One important objective might be verification of a proposed protocol against a service specification, i.e. to prove that a proposed protocol provides a specified service and possesses desirable properties and lacks some other. Then, the following properties of a protocol in general are of interest /19/:

- completeness (an allowed reception of a message by a process in a given state must be possible in the specification),
- deadlock-freeness (every stable-state N-tuple must have at least one next transmission allowed by an entity, except the system final state),
- livelock or tempo-blocking freeness (a new reception or transmission must not generate an overall system state that is one of the states on a path from the initial [illegible] the current overall system state, except for the overall system initial state for cyclic protocols),
- termination or cyclic behaviour (every interaction path starting from the overall system initial state and through a sequence of overall system states must lead to the overall system final (initial) state),
- boundedness (each time a message is entered to a channel, the total number of messages in that channel must not exceed an upper bound for that channel),
- liveness or absence of non-executable interactions (every newly created overall system state must result from any overall system state already existing).

2.2 ALTERNATING BIT PROTOCOL AS AN EXAMPLE OF A CFSM MODEL

The alternating bit protocol is known to be a simplyfied representative of common communication protocols. It is used to transmit data between two processes over two unreliable channels, c.f. Fig. 2. The complete model is represented there by ~~four~~ finite state machines (FSM) connected by ~~four~~ channels. The sender process must wait for an acknowledgement to each message before sending the next message. Each message carries a one bit sequence number that alternates for each message sent. Following /3,4/ the processes and channels each are represented by a FSM whose transitions represent transmissions. Five message types for sending (-) and receiving (+) of messages, resp. are used by the protocol, c.f. Fig. 3: ED = even data, OD=odd data, EDA/ODA=acknowledgement of ED/OD, E=end of data. two

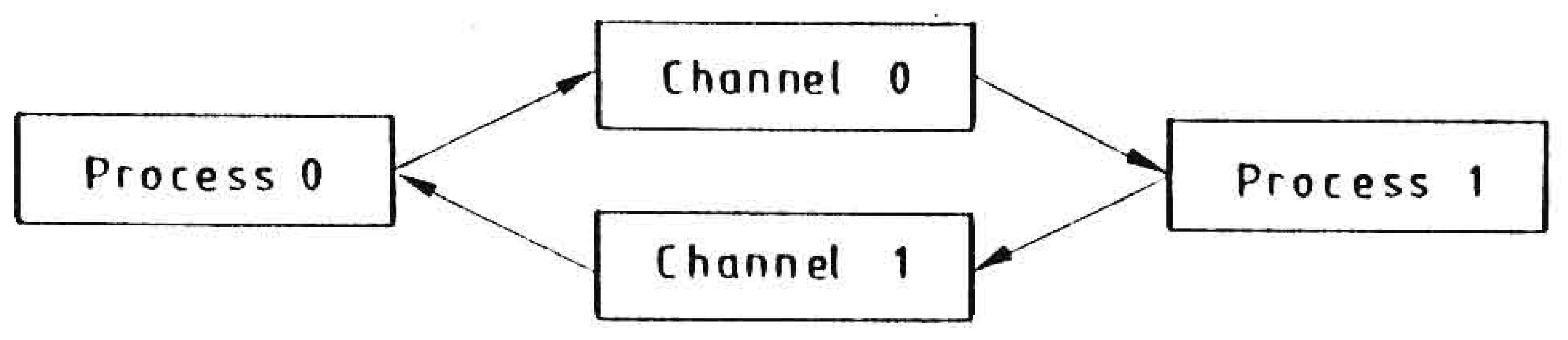

Fig. 2: Two processes communicating via separate channels

Processes 0 and 1 take turns in sending sequences of data messages; each sequence ends with an E message. Process 0 sends first, starting from state 0 and process 1 is then in state 4. A process waiting for an acknowledgement (EDA, ODA) refrains to transmit its message (ED, OD) until success.

The unreliable channels are modeled by two identical FSMs, assuming all errors being located at one place, and the rest of the channel functions according to a FIFO queue. A channel starts in state 0 and ignores/deletes some messages (remaining in state 0), whilst the others are transmitted correctly. Since any of the messages ED, OD and E may be repeated any number of times, there is no upper bound on the number of messages simultaneously in transit.

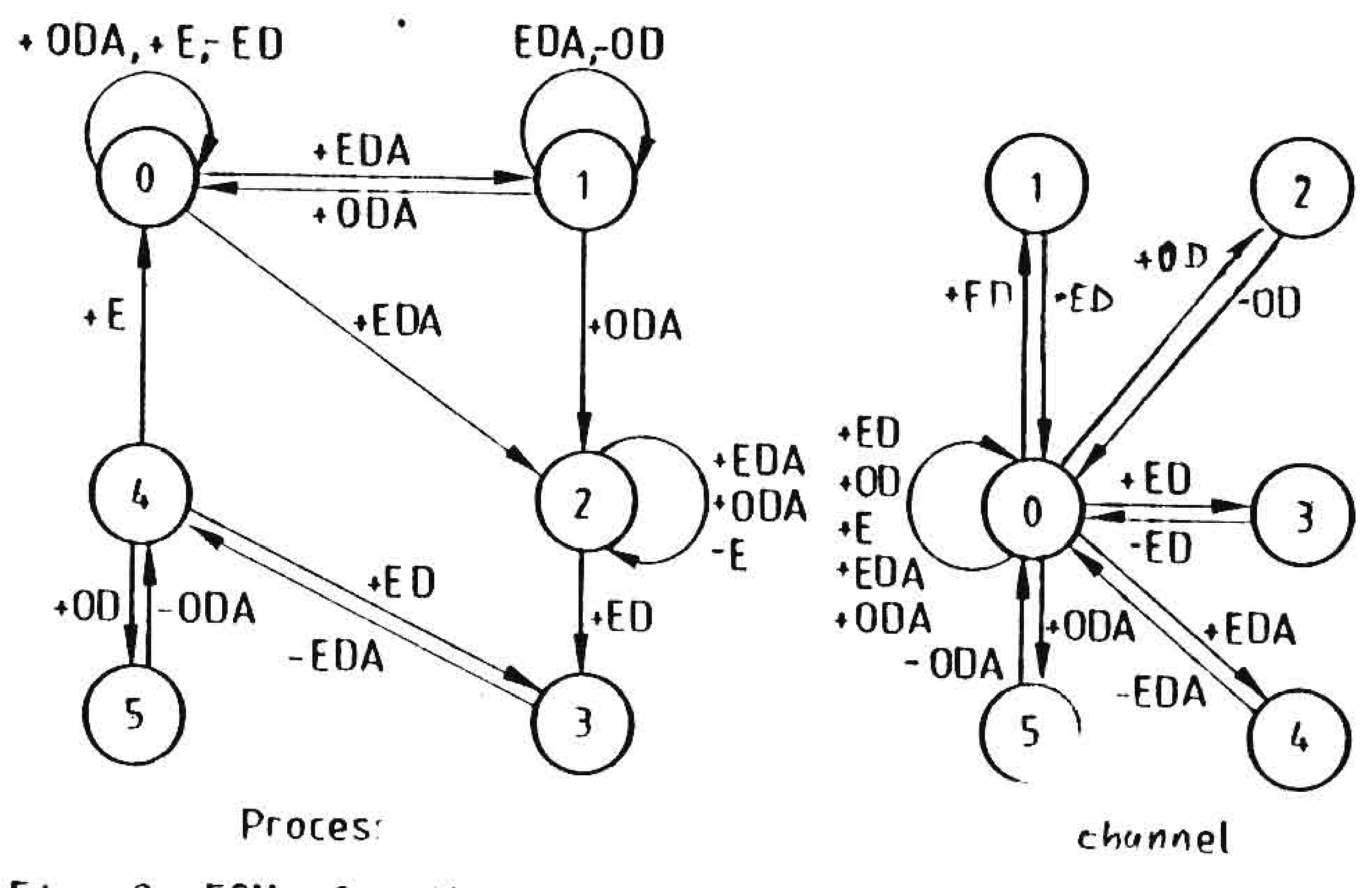

Fig. 3: FSMs for the alternating bit protocol (process) and unreliable channels.

The protocol can be proven to be deadlock free /3/. As the protocol has four FSMs each having 6 states, there are 6**4 = 1296 composite states to consider, which must be examined during an exhaustive reachability analysis. The example protocol has a cyclic communication graph and some other nice property P, which reduces the efforts needed for analysis substantially, to 16 composite states.

The example represents a CFSM protocol for which deadlock freedom can be proven to exist. However, this does not say that there exists an algorithm that automatically reads any CFSM protocol, having the P property and is able to decide, whether the protocol is deadlock free or not. It is likely, that any such algorithm would be too slow to be practical /3/.

State spaces of more realistic protocols tend to be substantially greater than shown for the alternating bit protocol. Nevertheless, progress has been made in developing tools being able to state existence or absence of some interesting properties of a protocol being specified in one of the usual forms like text, graphic representation and/or formal language code /5/. Communication protocols can be synthesized by building a directed graph for the activity of a protocol entity, which requires observation of four simple rules given in /19/ defining a step-by-step generation of states and connecting arcs and resulting, finally, in a FSM as shown in Fig. 3.

3. FORMAL DESCRIPTION TECHNIQUES

Most of the tools and methods for protocol validation, implementation and testing require that the specification of the protocol be given in a formal language. If one aims at performing a consistency verification, that a given protocol specification provides the communication service defined for that layer, c.f. Fig. 1, the service specification should also be given in the same language. Most specifications of computer communication protocols, however, are given in an informal manner, which makes the application of automated tools and methods difficult.

To reduce the inherent ambiguities of informal specifications, the above introduced state transition diagram technique is often applied. Additionally, in most OSI application layer protocols the ASN.1 /6/ is used to specify the parameter data structures of the PDUs /7/ as well as their encodings /8/. It is important to note that the FSM models and the ASN.1 notation describe different aspects of protocols. The former describe "major" states and kind of input/output interactions, ignoring the parameters, while the latter describe the data types of the interaction parameters and the coding of these parameters in the PDUs. However, even taken together, they do not cover all aspects of the protocol specification /5/. What they do not cover are such questions as how the parameter values of the output interaction of a protocol entity depend on state variables and/or previouly received interaction parameters. These aspects are covered by full specification in formal languages, only.

In this context the CCITT X.409 recommendation is to mention which presents a standardized specification technique for application layer protocols for Message Handling Services (MHS) and will become the subject of an independent recommendation in the 1988 version of the X.200 series.

3.1 STANDARDIZED FORMAL LANGUAGES

Many new formal languages have been developed in recent years for specifying software modules and especially protocols. These languages differ in the emphasis they place on
- abstractness (how implementation independent they can be),
- concurrency (how suitable they are for specifying parallel systems),
- analyzeability (how readily specifications can be verified), etc.

There is not, and never can expected to be, a single and universal formal language, any more than there will be a universal natural language or programming language /9/. Currently, protocols of communication systems have attracted much attention by the formal specifier because they involve well-defined, self-contained problems and present technical challenges due to the inherent concurrency and distributedness.

In view of the efforts of ISO to define standards for OSI it is highly desirable that designers and implementers world-wide will read and interpret the standards in a compatible way. However, most of the ISO standards are written in natural language. To ensure precision and analyzeability it was decided to develop some mathematically based languages and to call such languages Formal Description Techniques (FDTs). The objectives were /10/
- to enable unambiguous, clear and precise descriptions of OSI standards to be written,
- to allow such descriptions to be verified for consistency and correctness,
- to provide clear guidance to designers and implementers of OSI systems as to what, but not how, they should implement,
- to act as a sound basis for conformance testing.

Being pointless to produce definitive formal descriptions of OSI standards using FDTs which were not themselves standards, ISO has been developing standards for two particular FDTs: Estelle and LOTOS. Estelle /11/ (Extended Finite State Machine Language) draws together the research which has been undertaken on FSMs and the languages they define. LOTOS /12/ (Language of Temporal Ordering Specification) synthesises work on algebraic techniques for specifying data types and concurrency. Both Estelle and LOTOS have already been balloted by ISO as Draft International Standards.

Another standardized FDT has been available from CCITT since 1980 with SDL (Specification and Description Language) for describing telecommunications systems, generally. SDL was further developed in 1984 /13/ and the latest update was finished in 1988.

All three FDTs are therefore substantially stable and have been widely applied, especially SDL has been excessively used for defining standards e.g. of the future European Mobile Radio Network of CEPT/GSM.

3.1.1 Comparison of the standardized languages

There is presently some competition between the three languages, which each have certain advantages. By joint agreement of the ISO and CCITT groups concerned, each of the FDTs is equally acceptable, however, the three FDTs habe different strengths and weaknesses. This is due to the emphasis layed on different aspects, so they should be seen as complementary rather than in competition.

The above mentioned working groups have produced a commom document, which gives guidelines for the application of FDTs to OSI /14/, explains the nature of a FDT and gives carefully chosen examples in each of the FDTs. In /15/ the criteria for good formal descriptions are explained and a more technical comparison of the FDTs is provided. In what follows we give here a short characterization of the three FDTs according to /9,23/.

Estelle: The concept of extended FSMs (EFSM) is used, where one major state and a number of state variables are used. EFSMs (called module) are running in parallel and interchanging information messages (called interactions). The way of exchanging interactions is structured: A protocol specifier must declare, for each information path (called channel) between modules (and for both directions on it), the complete set of interactions that can flow on it together with the content of each interaction, in terms of elementary information fields. At each end of a channel (called access point) a FIFO queue models the storage capacity of the model. Different channels can be associated to the same queue, thereby ordering the arriving interactions according to their order of arrival. Actions are expressed in a subset of the PASCAL language. Estelle is rather implementation oriented and therefore very suitable for giving reference implementations of protocols and to build compilers which will generate some of the communication software automatically. This feature makes it especially suitable for performance analysis through simulation, using rapid prototypes, during the development process of a protocol.

SDL has a basis similar to Estelle but has been widely used for implementing telecommunication networks. In difference to Estelle it has its own special abstract data type (ADT) language ACT ONE /16/ and has multiple representations (graphical and program like). An automatic translation is defined between the two types of representation. SDL is implementation oriented, but its data typing feature allows it to be more abstract, and the variety of existing tools and representation make it easier to use at present.

LOTOS is based on the theory of process algebra and originates from three algebraic languages: ACT ONE being used as ADT language for defining data types (e.g. service data units) and two languages for formally specifying communication systems: Milner's CCS /17/ and Hoare's CSP /18/. LOTOS thus has features for describing static (data typing) and dynamic (process behaviour) aspects of the system. Data types are described abstractly in terms of their components and the effects of operations on them, avoiding implementation concerns (like definition of length of a field etc.). Dynamic behaviour is represented by processes interacting synchronously by events. Two main differences of LOTOS from Estelle and SDL should be mentioned: LOTOS descriptions are more abstract, requiring a bigger step to implementations and a fully formal definition exists making rigorous analysis of a description possible.

One important aspect when applying specification methods to define protocols are the possibilities of a method to explicitly reference the concept of time and of putting time constraints on the system performances. A method should be able to specify

a) that an action must be executed within T seconds,
b) that the environment will react after/within T seconds,
c) a time-dependent waiting status (e.g. "wait T seconds").

This requirement is not completely fulfilled by any of the three languages, but Estelle appears to be most powerful /23/.

This is not the place to discuss the pros and cons of the three languages in depth. Such a discussion is possible only after having given an introduction into the application of these techniques which can be found in /20,21,22,24/. Moreover, the tools available for analysis should be taken more carefully into account. A quite deep discussion of such tools for static and dynamic analysis of these specification methods can be found in /5/.

3.2 NOT STANDARDIZED FORMAL METHODS

Besides the three languages there are a number of other important specification methods, currently not standardized, and associated tools. Many tools are based on the Petri net formalism, others use different specification methods. Various logic-based methods are used, e.g. temporal logic /46/ allows the specification and verification of liveness properties of protocols.

3.2.1 Petri Nets

Petri nets (PNs) are described here in more detail, because we'll later discuss enhancements to them, which advantageously support performance evaluation of protocols defined by PNs. A PN comprises a set of places P, a set of transitions T, and a set of directed arcs A. Places are graphically represented as circles and transitions as bars, c.f. Fig. 4. Transitions and places, respectively, are connected through directed arcs. Places may contain tokens, which are drawn as black dots. A vector M, whose ith component represents the number of tokens in the ith place defines the state (usually called marking) of the PN. M' defines the initial marking of the PN. Following /25/ a PN is defined formally by

$$PN = (P,T,A,M),\ P = \{p1,p2,...,pn\},\ T = \{t1,t2,...,tn\},\ A \in \{P \times T\} \cup \{T \times P\},\ M' = \{m'1,m'2,...,m'n\}. \quad (1)$$

A place pi is an input (output) to (from) a transition ti (i=0,1,..) if an arc exists from the place (transition) to the transition (place). A place inhibits a transition if an inhibitor arc (dashed line with circular head) connects the place to the transition. A transition is enabled and can fire, when all of its input places contain at least as much tokens as indicated at its arcs and all of its inhibitor places are empty. When firing, a transition removes the respective number of tokens from each input place and places as many tokens in each output place as indicated on the respective arcs (no indication on an arc is equivalent to one token). The distribution of tokens in places is thereby modified, resulting in a new marking of the PN.

In Fig. 4 a PN for the well-known multiple-reader-single-writer problem is presented as an example /27/. Five tasks are assumed to exist. A task working on its local data (token in place p1) may wish to read (token in p6) or write (token in p2). Access to global data is limited by introduction of place p5 to one task in write status and three tasks at most in read status. A token in place p3 indicates a task being in write status. The number of tokens in places p5 plus p7 gives the number of tasks in read status. A firing transition may disable other transitions that are said to be in conflict with the one that fires (e.g. t1 and t4 are in conflict).

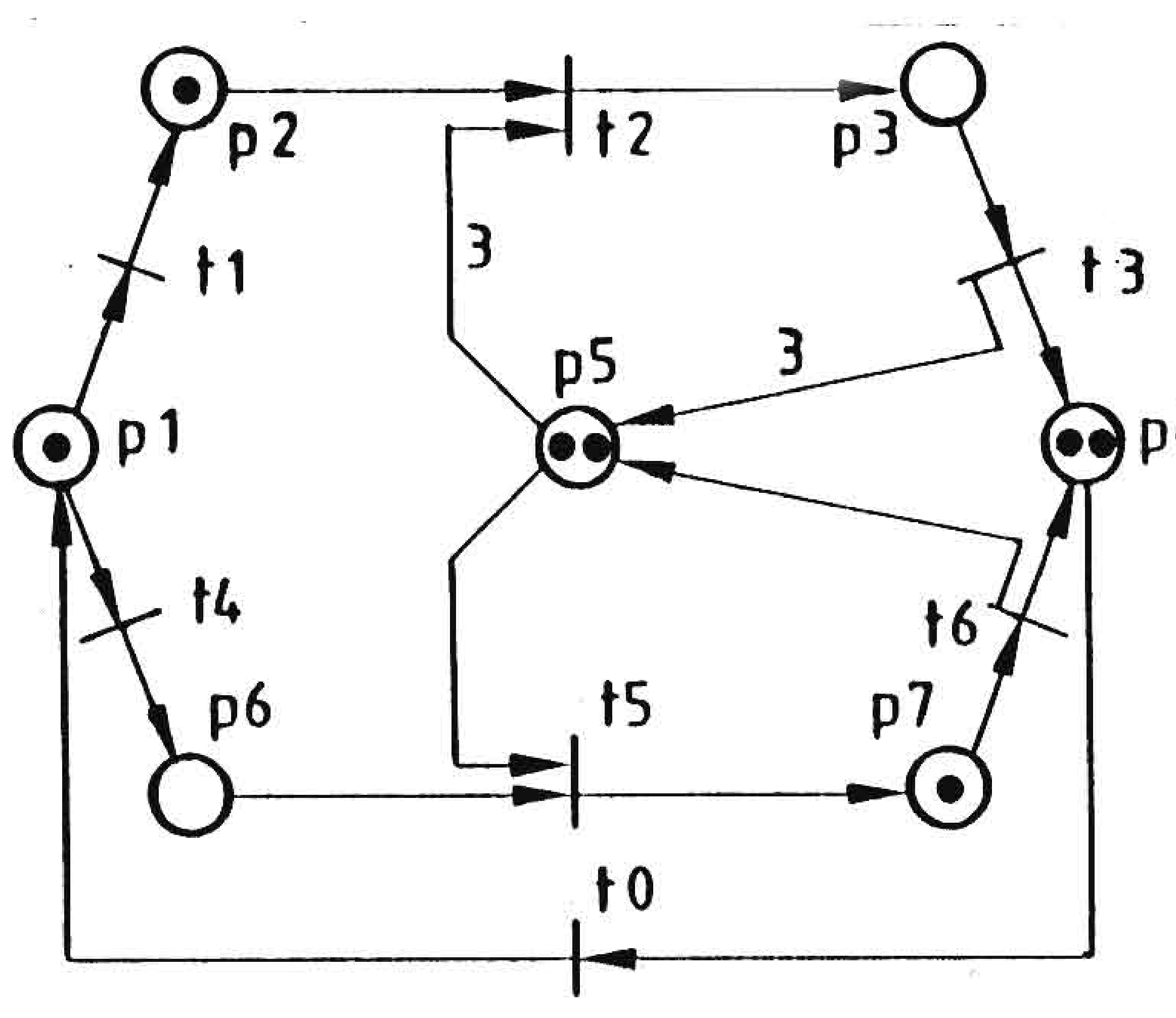

Fig. 4: Petri-net definition of the multiple-reader-single-writer problem with five tasks

The reachability set of a PN with a given initial marking M' is defined as the set of all markings that can be reached from M' by a sequence of transition firings. If the PN is such that, for any permitted marking, the maximum number of tokens in any place is notgreater than k, the PN is said to be k-bounded.

Due to the large number of places and transition resulting from a very detailed modeling needed to validate the design of a protocol, the reachability analysis becomes quite tedious. It can, however, be performed automatically to derive properties like liveness, boundedness and of properties like existence of invariants etc. The complexity of algorithms to prove such properties is exponential /26/.

4. PERFORMANCE EVALUATION OF PROTOCOLS

Much research has been done in the area of performance evaluation of communication protocols, mainly in the domain of packet-switched data networks and local area networks. Examples are data-link control procedures, flow-control protocols, and end-user-to-end-user protocols where such analyses yield results for throughput and/or response time. These analyses have traditionally used traffic and queuing theory. An area just being born is the estimate of performance direct from a formal protocol description. One means to obtain such a performance estimate direct from the formal protocol specification is via simulation, c.f. /47/. Here, the formal specification contains enough additional information for a simulation model to be automatically compiled from the specification. Running the simulation model yields the desired performance estimates. There exist an impressive number of tools for analysis, which are mainly based on one of the following techniques

1) statistical evaluation using simulation,
2) exact analysis, based on Markovian modeling, using
   - product form queueing networks (PFQN),
   - stochastic Petri networks (SPN),
3) approximate analytic modeling and mixed techniques,

In what follows, a brief introduction to the current state of the art in using these techniques is given.

4.1 PERFORMANCE EVALUATION BY SIMULATION

Statistical performance evaluation through simulation is the most general analysis method to study complex systems, since it allows them to be described by models with different levels of details down to the finest details. Simulation of complex protocols, however, brings up certain problems of which the principal is the computation time needed to produce results that are sufficiently precise.

Together with the development of computational methods for exact analysis of Markovian queueing systems during the 1970's and 80's, new approaches were made and gave birth to a wave of new simulation packages using the same basic model description. This development was complemented by substantial progress made in the field of statistical support for interpretation of simulation output and for producing "good" pseudo-random number sequences, making simulation for performance analysis a broadly accepted instrument /28, 29, 30, 31, 32, 33, 34/. This acceptance is still growing. Due to the rapid increase in computing capacity available from moderate-cost workstations and mainframes, and due to the user friendly interfaces offered by professional simulation packages, simulation tends to become a dominating method for protocol performance analysis.

4.1.1 Simulation tools and languages

A wide variety of simulation languages exist /33, 34, 35/ resulting from different approaches. Although the first were designed some twenty years ago they are still widely used.

One common approach is to create a set of program modules that achieve the functions specific to discrete event simulation, like management of the event schedule, gathering of statistics, generation of random numbers according to various probability distributions, etc. The approach takes advantage of a programming environment, where mathematical and graphical libraries and input/output interfaces exist.

At the author's institute MODULA-2 is used for performance analysis of protocols by simulation. In using such a modern language one is able to take advantage of newly introduced concepts to support good programming style, to ease development of error free programs and to have at one's disposal features like processes and coroutines, otherwise only found in simulation languages. Simulation languages are specifically adapted to the needs of modeling a system:

SIMSCRIPT /36/, for example is similar to a natural language and uses notions of entities, attributes and entity sets to ease definition and handling of objects like queues and customers. All the events are described in subroutines, including the end of simulation. A main program initializes the variables, starts the simulation by programming the first event and defines the end of simulation. The statistics demanded from variables of a simulation must be required explicitly /37/.

SIMULA /34/, is based on ALGOL and offers besides sophisticated data structures the concept of processes to describe activity sequences. Thereby, parallelism and synchronization of processes can be expressed, which results in a more comprehensive way of system definition.

GPSS /38/ is also based on the process concept. It comprises some 50 operations, each followed by a list of arguments and describing a basic activity of simulation like 'generate' (a random number), 'queue' (wait in a queue), 'depart' (from a queue), and describes activities as seen by the customer (called 'transaction' in GPSS). Statistics (mean and maximum of queue lengths and response times) are automatically produced at the end of simulation.

There exist an impressive number of further simulation languages, which partly combine the features of the languages mentioned. Some of them incorporate some advanced, or to a special application adapted features, c.f. /37/ for references to that languages. Simulation is very often one technique offered as part of a tool besides others, which is used to provide statistical analysis results when exact or approximate analytical methods are not applicable. Therefore, besides the languages mentioned, also the tools using combined techniques (simulation, exact/approximate analysis) should be considered, c.f. /54,55,58,59,60/.

One technique usually applied there is to decompose the problem at hand into sub-problems, some of which possibly can be analyzed in isolation using exact or approximate analytical techniques. The respective results are represented then by use of an equivalent server in the system's model and only the remaining part is solved using simulation /63/.

4.2 EXACT ANALYSIS, BASED ON MARKOVIAN MODELING

Queueing analysis has been widely used to analyze systems, by means of quite simplified models. Such models mainly can be characterized through definition of arrival processes to a number of queues, related service processes and the system structure, mainly being represented by a service discipline.

One example of such a model is shown in Fig. 5 where a cyclic service discipline is assumed in a model with N arrival streams into N separate queues, and each arriving customer requires a randomly distributed service time with mean β_i. A switch-over time t between service of consecutive customers is also included. This model has been very successfully applied to evaluate the mean transfer delay-throughput characteristics of the token-ring media-access control protocol /39,62/ according to standard IEEE 802.5 for local area networks (LANs). Other models of comparable complexity have been applied in /39/ to evaluate the performance of other concurrent LAN protocols, cf. Fig. 5, namely the protocols carrier sense multiple access with collision detection (CSMA/CD) and slotted ring.

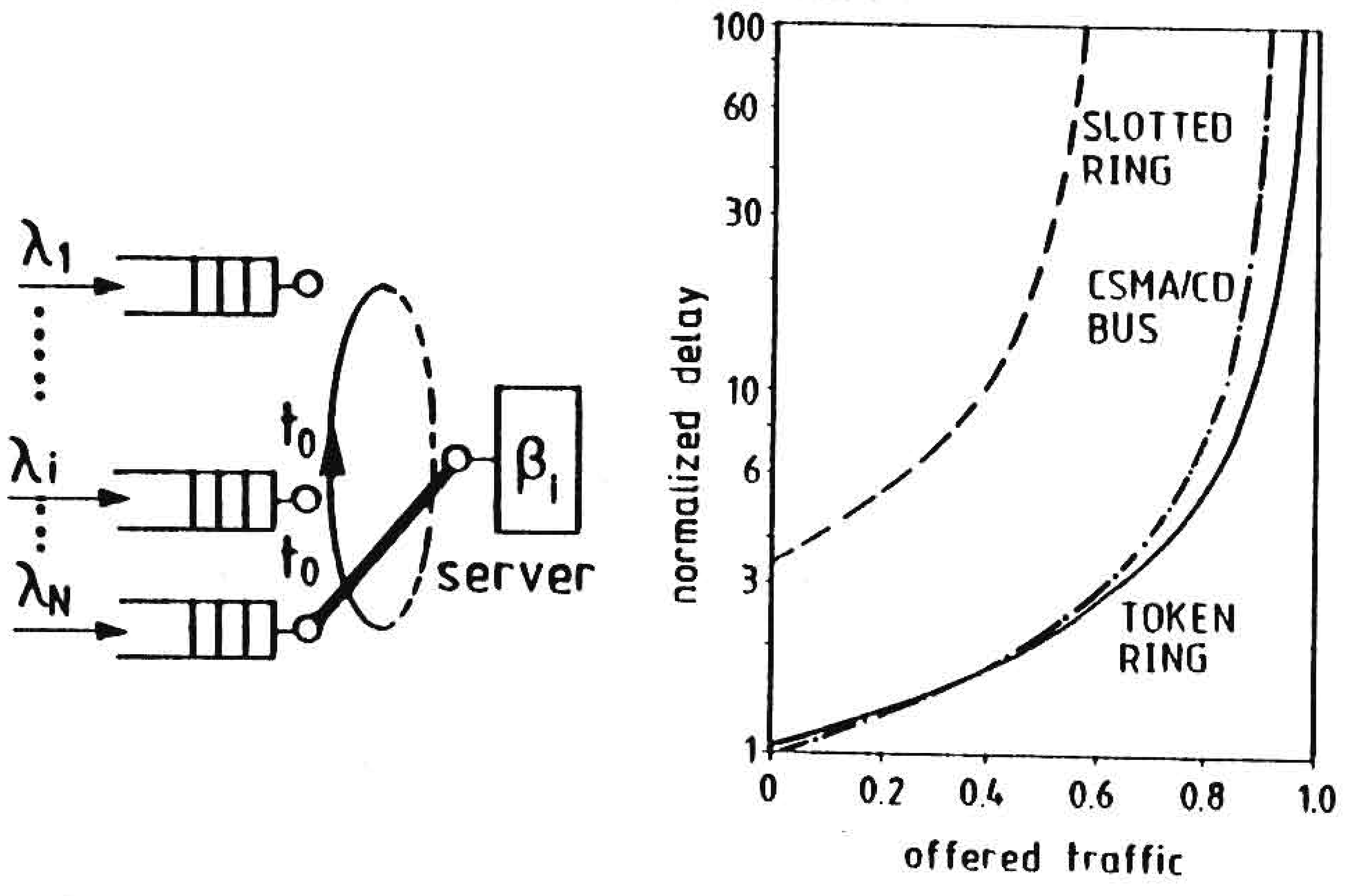

Fig. 5: Cyclic service of a token-ring model and resulting performance /39/. Included are also results of other LAN protocols. Parameters are the arrival rate λ_i and the mean service time β_i (i=1,2,...,N)

This type of analysis has been applied advantageously in teletraffic systems analysis since the 1930's when the notion of protocol was still unknown and since 1960 to analyze single- and multi-server models of computer systems /40,41,42,43,44,45/. The state of the art has evolved during this time period of application dramatically to a very high standard, being represented today by a number of very powerful tools. They contain the expert knowledge of two generations of researchers in the field of queueing systems and their analysis, and are mainly based on a class of models having the Markovian property (at least for some imbedded states of the underlying process). Using this type of analysis, quite complex protocols, including the consideration of time-out of a protocol /72/ and blocking /79/ can be tackled. Further, co-operation of a large number of independently in a network configuration operating servers can exactly be analyzed by a class of queueing models, called product form queueing networks and their extensions.

4.2.1 Queueing Networks (QNs)

The term queueing network denotes a class of queueing models, for which exact or approximate analysis techniques have been developed. They are well established as practical tools in computer system performance analysis and capacity planning. The principal reason for their success is the combination of expressive power and solution efficiency that they afford.

A queueing network is a collection of stations arranged in a way that customers proceed from one to another in order to fulfill their service requirement. Stations represent system resources and are characterized by a service rate, while customers represent jobs in the system consuming capacity of the station according to a given service time distribution. Each station has an associated queue where jobs may wait prior to service and a queueing discipline, which determines the order of service for the waiting jobs. Customers may be devided into several classes, such that all customers within a class are statistically identical with respect to routing probabilities and service demands. Served jobs leave a station and join the queue of another station according to the routing probability p(i,r)(j,s) describing the fraction of departures from a station i in class r that go next to station j in class s. Each class may be open or closed, cf. Fig. 6. In the closed-network part of Fig. 6 a constant number N of class-2-jobs circulate at all times, requiring at any queue at most N waiting positions. In an open network (state dependent) arrival and departure processes exist. A station together with its queue is called a service center.

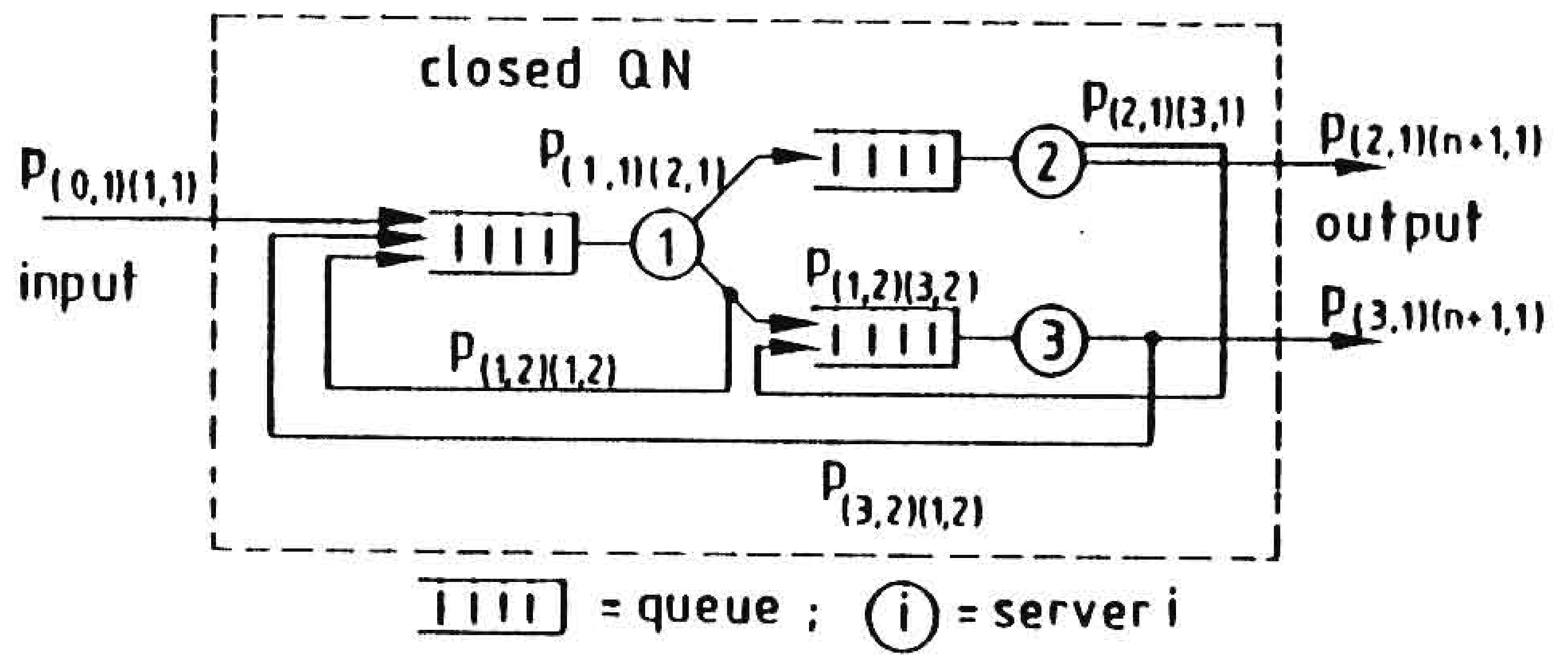

Fig. 6: Mixed queueing network being open w.r.t. class 1 and closed w.r.t. class 2.

4.2.1.1 Product Form Queueing Networks (PFQN)

PFQNs are a special form of queueing networks, being often referred to as BCMP networks /49/, that have a product form solution. The service-time distributions and service disciplines permitted at a station must be such, that Markovian input processes to a service center are transformed into Markovian departure processes. This property offers the possibility to aggregate service centers and replace them by an equivalent service center, without changing the behaviour of the remaining network. The aggregated service center can be solved separately /50,51/. Efficient exact analytic solution techniques exist, which make possible interactive tools, enabling the user to explore a large design space rapidly /53,54,55,56,57,58,59,60/. Some of these tools are compared and evaluated in /37,61/.

The notation used to describe the models allows them also to be developed very rapidly: only few parameters are required and these correspond directly to components of the application being modeled. One principal limitation of PFQN is that they do not have a general construct for representing synchronization. The ability to represent concurrency of processes is a very important requirement when protocols are modeled, which normally contain parallelisms. To cover this, extended QNs (EQNs) were developed, offering an approximate analytic solution.

4.2.1.2 Extended Queueing Networks (EQNs)

EQNs are QNs that have been augmented with passive resources, fork and join nodes and split nodes. These passive resources are represented by tokens counting the number of resources that are currently available. In EQNs, which e.g. can be described by RESQ /52/, special nodes are defined where customers can create, acquire, release or destroy a token. A customer is blocked, when a token is not available when needed, and waits until another customer creates or releases it.

EQNs might also contain set nodes where data variables can be modified. The data variables can be defined globally or associated with an individual customer. Such extensions make EQNs well-suited for application for performance analysis of protocols, and make them with respect to their descriptive power equivalent to general-purpose simulation languages. When using data variables, in fact, simulation is used to evaluate the models.

4.2.2 Stochastic Petri Networks (SPNs)

Stochastic Petri nets are a newer approach to system performance modeling and analysis that are attracting increasing interest. They were introduced by Molloy in 1981 /67/ and consist of PN models in which an exponentially distributed firing time is associated with each transition. The formal definition of a PN given in Eq. (1) is extended thereby to

$$SPN = (P, T, A, M', R), \qquad (2)$$

where $R = \{r1, r2, \ldots, rm\}$ is the set of firing rates (possibly marking dependent) associated with the transitions.

Molloy has shown that SPN's are isomorphic to continuous time Markov chains, thus the standard analysis techniques for this class of model can be applied to compute performance measures of interest /68,69/. The SPN markings correspond to the states of the respective Markov chain, which is finite in case of k-bounded SPN's. In /68,69/ a combination of immediate transitions (firing in zero time) and transitions with exponential firing time is introduced and was termed as general SPN (GSPN).

The major problem of SPN's is one of efficiency, due to the large state space resulting already from moderate sized problems. Aggregation techniques suitable to support the solution of smaller problems by exact or approximate decomposition are currently not available.

One important question investigated in /68/ is whether SPN's and EQN's are fundamentally the same, or whether there are intrinsic differences that make one approach more powerful than the other. The following observations were stated there:

a) any open, closed or mixed single class EQN model can be expressed as a SPN of equal complexity
b) any multi-class EQN can be represented as a SPN of comparable complexity, if one allows to specify scheduling disciplines for transitions and routing probabilities per class
c) no algorithm is known to construct an equivalent EQN from the SPN representation
d) there exist synchronization mechanisms that can be represented in the SPN notation but not in the EQN notation.

Fig. 7 illustrates the construction of an equivalent SPN model from a simple EQN model /68/. It can be seen that there is less semantic information conveyed in the SPN primitives, which reduces the conceptual information available to the reader from the net diagram.

EQN related element	SPN-equivalent
queue, passive resource	transition
allocate/and release node	transition
Routing probability	decision place

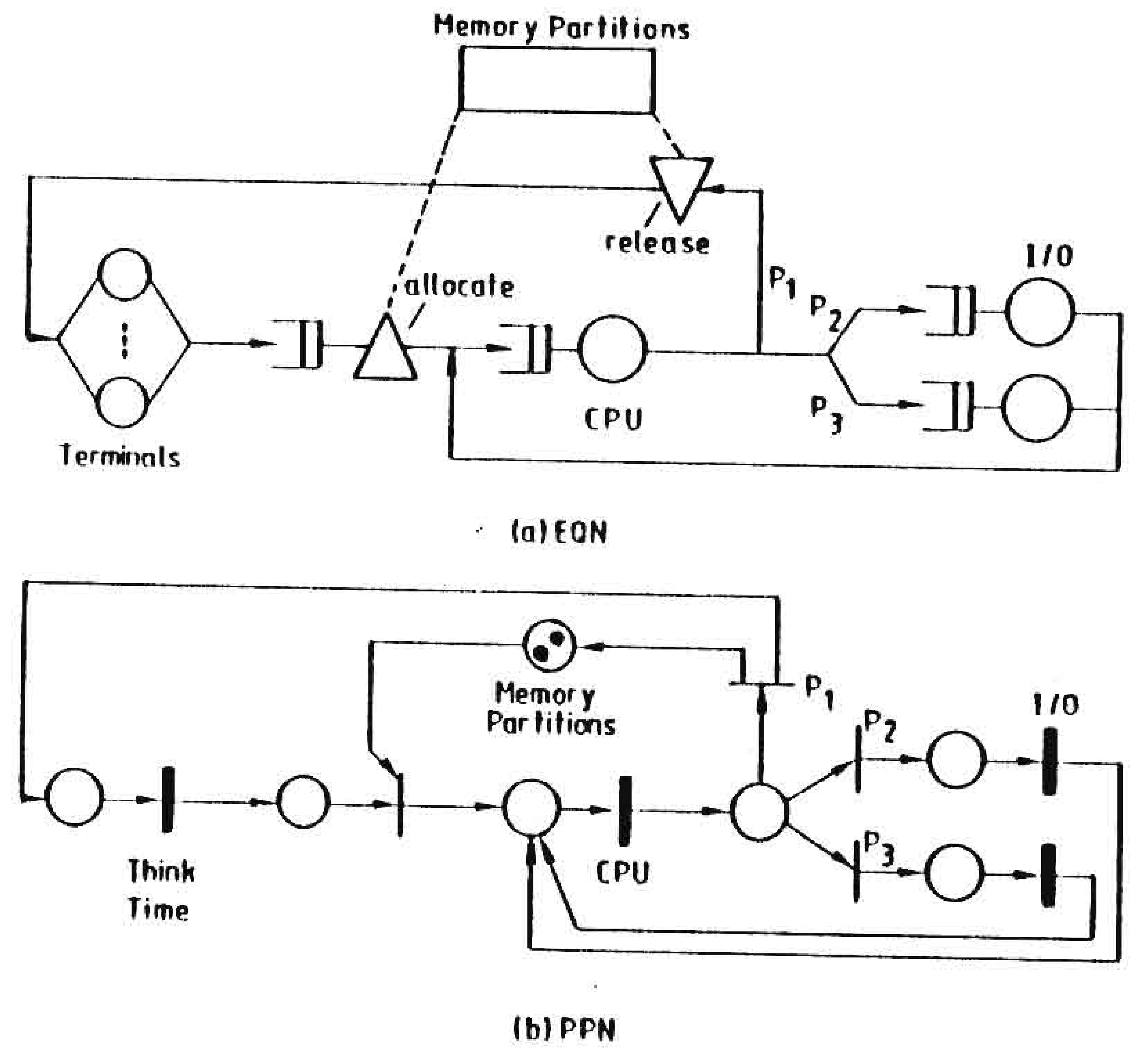

Fig. 7: Single class EQN and equivalent SPN /68/. The thick bars represent timed transitions

It can be concluded from the observations a) to d), that SPNs have more expressive power than EQNs but a more complicated appearance, especially to persons accustomed to the QNs notation.

With respect to the efficiency of evaluation, SPNs, in general, require resources that are exponential in the size of the model to produce results. So, only models of relatively modest size may be solved at all, and even for small models the efforts needed to compute performance parameters are orders of magnitudes larger than for EQNs. Thanks to the existence of good packages e.g. /70,71,76/ for SPNs and the growing capacity available from modern computers, resources needed are believed to be a weaker argument only against SPNs.

4.3 APPROXIMATE ANALYTIC MODELING AND MIXED TECHNIQUES

Process synchronization mechanisms, i.e. share of a common resource set and co-operation between parallel processes usually are of major importance for the performance of protocols. However, their direct description within a queueing network model is not as straightforward and violates the product form assumptions. Examples of such mechanisms are simultaneous resource possession, a process blocking another one, blocking itself while waiting for a signal, or activating a process by sending some information. To model these influences, approximately, supplementary service centers were introduced in queueing networks introducing some well defined serialization delay. The service center's parameters were defined iteratively and verified through simulation /64,65/. This technique also was broadly applied to model resource constraints in data base management systems, see references in /66/.

One interesting approximative approach to model synchronization is described in /66/, where a flag mechanism (using a Petri-net transition) is introduced, which is demonstrated to be reduceable to an equivalent exponential server with correlated arrival and service processes. Applying this technique results in an approximate solution of the queueing network.

Another interesting approach for the approximate analysis of QNs with blocking, where a job is forced to reside in the source station (keeping this server blocked) until a place in the destination station becomes available, was proposed in /74,75/. The approximation is based on the fact that models having the same state space, are equivalent and can be characterized by the same performance parameters. Therefore, a non-blocking closed QN with an appropriate number of circulating jobs is derived and solved, having the same state space as the blocking network. The method is approximative, because for networks with multiple stations (more than two stations) the state space of a blocking network is not isomorphic to the state space of a nonblocking network. The results received by the method are especially well applicable for protocol analysis, where many applications can be modeled through two stations communicating under some blocking conditions.

Performance analysis of communication systems does not always rely on a detailed analysis of the underlying protocols. Analyses which do so, usually rely on an analysis of the protocol state space transition graph. For nontrivial protocols the state space tends to become very large, resulting in a very complex analysis. In /77/ a new performance analysis technique which applies multi-class queueing models was proposed. Although systems with many thousand states can be handled by such models, state reduction techniques known from research in the protocol validation area are proposed to be used also for performance analysis. The method of projections /78/ to reduce the state space is shown in /77/ to substantially reduce the complexity of the performance analysis and to produce faithful performance results.

Separable (completely decomposable) QNs do not provide a sufficiently rich set of model constructs. E.g. to model priority scheduling and memory constraints would result in non-separable networks. Since even separable networks are intractable when there are many classes, considerable research on approximate analysis techniques was spent. Results of these efforts were incorporated into production tools for capacity planning, thereby extending the width of their applicability substantially. However, the interaction of several approximations can lead to problems not encountered when using the approximations in isolation, and the sensitivity to small changes in parameter values can be more important than their average or worst case errors /73/. There does not seem to be any easy solutions to this problem resulting from the conflicting goals of accuracy and efficiency. However, taking into account that workload parameters have very large inherent inaccuracies, this observation should not be overemphasized. Concluding this paragraph it can be stated that a number of promising approaches for approximate performance analysis of cooperating servers to model protocols have been developed.

5. CONFORMANCE TESTING

To achieve the objective of open systems interconnection (OSI), systems must be tested in an approved manner to certify that they conform to the relevant standards. The ISO and CCITT are now working together to produce a common methodology and framework for conformance testing /80/ applicable to OSI standards and similar CCITT X-series and T-series recommendations. The work is already being used as the basis for standardization of test suites for X.25 terminals, the OSI connection-oriented transport protocol (ISO 8073) and MHS (X.400 series). The major aspects of the testing methodology and framework are presented in /81/.

Conformance testing can be considered a case of consistency validation. A given protocol implementation, usually called "implementation under test" (IUT) is checked against the protocol specification, which acts as reference. The validation method is testing. The IUT is stimulated by test input which is generated by one or several test modules /5/. The output generated by the IUT in response to the test input must be observed and compared with the protocol specification in order to determine, whether the observed output is a possible one according to the specification. If any behaviour of the IUT does not conform to the reference specification, an error is indicated.

Testing can be performed to various extent and therefore, for conformance testing four categories have been identified:
- the basic interconnection test,
- a functional range test,
- full conformance tests,
- specific conformance resolution tests /81/.

A successful test (without an error being found) does not imply that the IUT conforms completely. A test suite is a sequence of individual test cases, which together have a good chance of detecting most errors of an implementation. Some tools for automated testing and the definition of test cases are existing already, see /5/ for more details. Most tools generating test sequences are based on EFSMs models, because many test methods have been published for such models. An overview of methods used in this field can be found in /82/. The set of test cases developed for the ISDN D-channel network layer protocol in /83/, and the method to derive test sequences from a protocol specification given in Estelle in /84/ may serve as examples to illustrate possible approaches in conformance testing.

6. CONCLUSIONS

The discussion presented in this paper shows that many methods and tools exist that have successfully been used in the development process of protocols, mainly for their specification, validation, performance analysis and testing. The rapid growth of of new ideas in these fields and their quick implementation as part of existent tools is very impressive. However, the potential for a much larger application of these or related tools exists. It is coupled with the definition of formal specifications of the OSI protocols and services, written in an accepted FDT and the integration of specification methods using EFSMs and performance analysis methods based on FSM descriptions into common tools. There are a number of areas, where further research and development is needed and is expected to provide important improvements to the state of the art. None of the aspects addressed in this paper as being relevant for the performance analysis of protocols can be termed as sufficiently well covered by the existent techniques. Much further efforts have to be spent to make the methods in use more mature and efficient, and to bring them into a position to adequately solve the respective tasks.

REFERENCES

/1/ ISO 7498, Information Processing Systems - Open Systems Interconnection - Basic Reference Model, International Organization for Standardization, 1984.

/2/ D. Brand, P. Zafiropulo: On communicating finite state machines. J. ACM 30 (1983), 323-324.

/3/ J. Pachl: Protocol description and analysis based on a state transition model with channel expressions. Protocol Specification, Testing and Verification VII, H. Rudin, C.H. West (Eds.), Elsevier Science Publishers B.V. (North Holland), 207-219.

/4/ P. Zafiropulo, C.H. West, H. Rudin, D.D. Cowan, D. Brand: Towards analyzing and synthesizing protocols. IEEE Trans. Comm. COM-28 (1980) 651-661.

/5/ G.v. Bochman: Usage of protocol development tools: The results of a survey. Protocol Specification, Testing and Verification VII, H. Rudin, C.H. West (Eds.), Elsevier Science Publishers B.V. (North Holland), 139-161.

/6/ F. Caneschi, E. Merelli: An Architecture for an ASN.1 Encoder/Decoder. Computer Networks and ISDN Systems 14 (1987), 297-303

/7/ ISO DIS 8824 Information Processing - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)

/8/ ISO DIS 8825 Information Processing - Open Systems Interconnection -Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)

/9/ K.J. Thurner: LOTOS: A practical formal description technique for OSI. Internat. Open Systems, Online Publications, Pinner, UK, 1987 265-279

/10/ ISO: TC97/SC21/N1541 - WG1 Programming of Work, Geneva, Sept. 1986

/11/ ISO: DP9074rev. - Estelle, A Formal Description Technique based on an Extended State Transition Model. Geneva, August 1986

/12/ ISO: DP8807rev. - LOTOS, A Formal Description Technique based on the Temporal Ordering of Observational Behaviour. Geneva, July 1986

/13/ CCITT: Z.100 to Z.104 - SDL, Specification and Description Language. Geneva, Red Book 1984.

/14/ ISO: TC97/SC21/N1534 - Guidelines for the Application of FDTs to OSI. Geneva, September 1986

/15/ ISO:TC97/SC21/N1543 - Development for the Application of FDTs to OSI. Geneva, September 1986

/16/ H. Ehring et.al.:ACT ONE, an Algebraic Specification Language with two Levels of Semantics. Bericht Nr. 83-03, Berlin University, 1983

/17/ A.J.R.G. Milner: A Calculus of Communication Systems. LNCS 92, Springer Verlag, Berlin 1980

/18/ C.A.R. Hoare: Communicating Sequential Processes. Prentice Hall, London, 1985

/19/ D.P. Sidhu: Synthesis of Communication Protocols. ICC 1982, 4C.5.1-4

/20/ L. Logrippo, A. Obaid, J.P. Briand, M.C. Fehri: An Interpreter for LOTOS, A Specification Language for Distributed Systems. Software Practice and Experience 18(4), 365-385.

/21/ R. Saracco: CCITT SDL: Overview of the Language and its Applications. Computer Networks and ISDN Systems 13 (1987), 65-74.

/22/ J.P. Courtiat: How could Estelle become a better FDT? Protocol Specification, Testing and Verification VII, H. Rudin, C.H. West (Eds.), Elsevier Science Publishers B.V. (North Holland), 43-60.

/23/ J. Bruijning: Evaluation and Integration of Specification Languages. Computer Networks and ISDN Systems 13 (1987), 75-89.

/24/ G. T'Hoft: Formal description techniques: Communication tools for data communication specialists. Computer Networks and ISDN Systems 14 (1987), 311-321

/25/ J.L. Peterson: Petri nets. ACM Computing Surveys, Vol. 9, Sept. 1977, 223-252

/26/ M. Jantzen, R. Valk: Formal properties of place/transition nets. LNCS 84, Springer Verlag, Heidelberg, New York 1980

/27/ O. Herzog, W. Reisig, R. Valk: Petri-Netze: ein Abriss ihrer Grundlagen und Anwendungen. Informatik Spektrum (1984) 7: 20-27

/28/ J.P.C. Kleijnen: Statistical Techniques in Simulation, Part I and Part II, Marcel Dekker, Inc. New York 1974/1975.

/29/ O.J. Dunn, V.A. Clark: Applied Statistics: Analysis of Variance and Regression. John Wiley+Sons, New York, 1987

/30/ H. Stenger: Stichproben. Physica-Verlag, Heidelberg, Wien, 1986

/31/ P. Bratley, B.L. Fox, L.E. Schrage: A Guide to Simulation. Springer-Verlag, New York, Berlin 1983.

/32/ C.H. Sauer, E.A. MacNair: Simulation of Computer Communication Systems. Prentice-Hall, Inc. Englewood Cliffs, New Jersey 1983

/33/ R.S. Lehmann: Computer Simulation and Modeling: An Introduction. Distrib. by Halsted Press Div. of John Wiley+Sons, New York, 1977

/34/ W.R. Franta: The Process View of Simulation. North-Holland, New York 1977

/35/ G. Fishman: Concepts and Methods in Discrete Event Digital Simulation. John Wiley+Sons, New York 1973

/36/ P.J. Kiviat, H.M. Markowitz, R. Villanueva: SIMSCRIPT II.5 Programming Language. ed. Alasdar Mullarney, CACI, Los Angeles, 1983

/37/ L. Coyette, D. Duong, B. Delosme: Evolution of performance evaluation packages. Modeling Techniques and Performance Evaluation, S. Fdida and G. Pujolle (eds.) Elsevier Science Publishers B.V. North Holland 1987, 311-321

/38/ T. Schriber: Simulation using GPSS. John Wiley+Sons, New York 1974

/39/ W. Bux: Local-area subnetworks: A performance comparison. IEEE Trans.Comm. Vol. COM-29, 10, Oct. 1981, 1465-1473.

/40/ E.C. Molina: Applications of the theory of probability to telephone trunking problems. BSTJ 6 (1927), 461-494.

/41/ C.D. Crommelin: Delay probability formulae when the holding times are constant. POEEJ 25(1932), 41-50

/42/ J. Riordan: Stochastic Service Systems. John Wiley +Sons, New York 1962

/43/ R.B. Cooper: Introduction to Queueing Theory. The Macmillan Comp. New York 1972

/44/ L. Kleinrock: Queueing Systems Vol. 1 and Vol. 2. John Wiley + Sons, New York 1975/76

/45/ D. Gross, C.M. Harris: Fundamentals of Queueing Theory. John Wiley+Sons, New York 1974

/46/ B. Hailpern, S. Owicki: Verifying network protocols using temporal logic, ICC 1980, 18-28

/47/ W. Bauerfeld: Protocol performance prediction. Proc. ICC, Boston, Mass. June 1983

/48/ A. Danthine (ed.): Proc. Computer Network Protocols Symposium, Liege, Belgium, Febr. 1978 (see also Computer Networks, Vol. 2, No. 4/5, Sept/Oct. 1978

/49/ F. Baskett, K.M. Chandy, R.R. Muntz, F.G. Palacios Open, closed and mixed networks of queues with different classes of customers. J. ACM, Vol. 22, 248-260

/50/ G. Balbo, S.C. Bruell: Computational aspects of aggregation in multiple class queueing networks. Perform. Evaluation, Vol. 3 (1983), 177-185

/51/ E.D. Lazowska, J. Zahorjan, G.S. Graham, K.C. Sevcik: Quantitative System Performance. Englewood Cliffs, NJ, Prentice Hall, 1984

/52/ C.H. Sauer, E.A. MacNair, J.F. Kurose: The research queuing package: Past, present, and future. Proc. 1982 Nat. Computer Conf., AFIPS 1982

/53/ BEST/1: Reference Manual. BGS Systems, Inc. Waltham MA, 1986

/54/ RESQ Users Guide, IBM Corp., 1982

/55/ PAWS/A Users Guide, Information Research Associates, 1983

/56/ A.E. Krzesinski, M. Booyens, P.S. Kritzinger, P. Teunissen, S. van Wyk: SNAP - An analytical multiclass queueing network analyzer. Proc. Int. Conf. Mod. Techn. Tools Perform. Analys., Paris, France, May 1984

/57/ MAP Reference Manual. Quantitative System Performance, Inc. Seattle, WA 1986

/58/ S.K. Tripathi, A.K. Agrawala, M. Abrams, K.K. Ramakrishnan, M. Singhal: STEP-1: A user friendly performance analysis tool. Proc. Int. Conf. Mod. Techn. Tools Perform. Analys., Paris, France, May 1984, 201-221

/59/ M. Veran, D. Potier: QNAP 2: A portable environment for queueing networks modeling. Proc. Int. Conf. Mod. Techn. Tools Perform. Analys., Paris, France, May 1984, 25-63

/60/ H. Beilner, J. Maeter: COPE: Past, present and future. Proc. Int. Conf. Mod. Techn. Tools Perform Analys., Paris, France, May 1984, 181-199

/61/ G. Bolch, G. Zeis: Softwaretools zur Leistungsbewertung von Rechensystemen. Angewandte Informatik 11/1987, 470-480

/62/ V. Rego, L.M. Ni: Analytic models of cyclic service systems and their application to token-passing local networks. IEEE Trans. Comp. Vol. 37, No. 10, Oct. 1988, 1224-1234

/63/ P.J. Courtois: Decomposability: Queueing and Computer System Applications. Academic Press, NY 1977

/64/ S.C. Agrawal, J.P. Buzen: The aggregate server method for analyzing serialization delays in computer systems. ACM TOCS 1, 2, May 1983,116-143

/65/ A. Thomasian: Queueing network models to estimate serialization delays in computer systems. Proc. Performance 83, Maryland, North Holland 1983 45-59

/66/ D. Mailles, S. Fdida: Queueing systems with flag mechanisms. Modeling Techniques and Perform. Evaluat., S. Fdida, G. Pujolle (eds.), Elsevier Science Publishers, North Holland 1987, 167-190

/67/ M.K. Molloy: On the integration of delay and throughput measures in distributed processing models. Ph.D. dissertation, UCLA 1981

/68/ M. Vernon, J. Zahorjan, E.D. Lazowska: A comparison of performance Petri nets and queueing network models. Modeling Techniques and Perform. Evaluat., S. Fdida, G. Pujolle (eds.), Elsevier Science Publishers, North Holland 1987, 191-202

/69/ G. Balbo, S.C. Bruell, S. Ghanta: Combining queueing networks and generalized stochastic Petri nets for the solution of complex models of system behaviour. IEEE Trans. Computers, Vol. 37, No. 10, Oct 1988, 1251-1268

/70/ M. Ajmone-Marsan, G. Balbo, G. Ciardo, G. Conte: A software tool for the automatic analysis of generalized stochastic Petri net models. Proc. Int. Conf. Modeling Techniques Tools Perform. Analys., Paris, France, May 1984

/71/ G. Chiola: Great SPN Users Manual, Vers. 1.3, Dipart. Informatica, Univ. Torino, Aug. 1983

/72/ D. Manfield, P. Tran-Gia, H. Jans: Modeling and performance analysis of inter-processor messaging in distributed systems. Performance Evaluation 7 (1987), 285-298

/73/ J. Zahorjan, E.D. Lazowska, K.C. Sevcik: The use of approximations in production performance evaluation software. Modeling Techniques and Perform. Evaluat., S. Fdida, G. Pujolle (eds.), Elsevier Science Publishers, North Holland 1987, 297-307

/74/ I.F. Akyildiz: On the exact and approximate throughput analysis of closed queueing networks with blocking. IEEE Trans. Software Eng., Vol. 14, No. 1, 62-70

/75/ I.F. Akyildiz: Mean value analysis for blocking queueing networks. IEEE Trans. Software Eng., Vol. 14, No.4, April 1988, 418-428

/76/ G. Chiola: A graphical Petri net tool for performance analysis. Modeling Techniques and Perform. Evaluat., S. Fdida, G. Pujolle (eds.), Elsevier Science Publishers, North Holland 1987, 323-333

/77/ P.S. Kritzinger: Protocol performance using image protocols. Protocol Specification, Testing and Verification VII, H. Rudin, C.H. West (Eds.), Elsevier Science Publishers B.V. (North Holland), 321-335

/78/ S.S. Lam, A.U. Shankar: Protocol verification via projections. IEEE Trans. Software Engin., Vol. SE-10, 4, July 1984, 325-342

/79/ Z. Papir: Validation of admission delay model for two-link virtual route with window flow control. Performance Evaluation 7 (1987), 83-86

/80/ ISO/TC97/SC21/WG16-1, OSI Conformance testing methodology and framework, edited by D. Rayner, Egham, Sept. 1986. Part 1: General Concepts, ISO/TC97/SC21 N1525, Part 2: Abstract test suite specification, ISO/TC97/SC21 N1526, Part 3-6: ISO/TC97/SC21 N 1514, (ISO DP 9646/1 and DP 9646/2)

/81/ D. Rayner: Progress on standardizing OSI conformance testing. Computer Standards and Interfaces 5 (1986), 317-334

/82/ B. Sarikaya, G. Bochman: Synchronization and specification issues in protocol testing. IEEE Trans. Communic. Vol.32, No. 4, April 1984, 389-395

/83/ E.P. Rathgeb, C. Homann, H.L. Truong, G. Waldmann: Protocol testing for the ISDN D-channel network layer. Protocol Specification, Testing and Verification VII, H. Rudin, C.H. West (Eds.), Elsevier Science Publishers B.V. (North Holland), 421-434