A Methdology for Specification-Based Performance Analysis of Communication Protocols*

CUCS-206-85

Nihal Nounou and Yechiam Yemini

Computer Science Department
Columbia University
New York, N.Y. 10027

September 1985

ABSTRACT

This paper presents a methodology for automatically analyzing the performance of communication protocols. The methodology consists of three steps. First, a protocol designer uses a formal algebraic method to specify the functional behavior of each of the communicating processes involved in the protocol. Second, the concurrent behavior of the protocol is automatically computed. Third, the protocol designer formally specifies timing requirements and/or performance measures and requests their analysis. This analysis is carried out automatically from the formal specification of the protocol augmented with some timing information. The methodology is implemented in ANALYST: an automated protocol performance analyzer. A connection establishment and a data transfer protocol are used to demonstrate the methodology. By analyzing the timing requirements of these protocols, optimal settings of their system parameters are obtained. Results obtained from analyzing their performance measures are shown to agree with manual results previously reported in the literature.

*This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165, the New York State Center for Advanced Technology in Computers and Information Systems under NYSSTF CAT(88)-8, and a grant from AT&T.

1. Introduction

A protocol is a set of rules that regulate communications among distributed processes in a computer network. Protocol design involves a rather complex set of problems since the processes involved share common resources concurrently and asynchronously; communicate through unreliable channels that incur random message delays; and behave in a time-dependent fashion. The first step in designing protocols is to formally specify the behavior of its various communicating processes. The concurrent behavior of these processes then has to be examined to ensure that it satisfies the functional and performance objectives of the protocol. Even for a simple protocol with few communicating processes, the concurrent behavior typically includes too many possibilities to be analyzed manually.

Consequently, there has been great demand for automated tools to aid the protocol designer in verifying the functional requirements and analyzing the performance of such concurrent behavior. Functional requirements of a protocol assert that its behavior is *safe*, i.e., any goal achieved actually satisfies its functional objectives, and *live*, i.e., such goals are guaranteed to be eventually achieved. That is, verifying functional requirements of protocols is concerned with assessing qualitatively their functional behavior.

On the other hand, analyzing the performance of protocols is concerned with assessing quantitatively their timing behavior. More specifically, we view performance analysis of protocols as consisting of 1) analyzing the requirements to be met by its timing behavior (*timing requirements*) to ensure that the protocol performs efficiently, and 2) analyzing key measures that indicate how efficiently the protocol performs. For example, consider a protocol using time-out to recover from message

loss in the transmission medium. Such a protocol performs efficiently if it has a minimal probability of time-out occurring before a loss and a minimal time between a loss and a time-out. (As will be shown later, these two timing requirements are contradictory and some balanced requirement can be defined and analyzed.) One key measure of the efficiency of the protocol, for example, is the mean time starting with the sender sending a message until it receives its successful acknowledgment.

Considerable attention has been especially given in the past decade to the development of automated tools for verifying functional requirements of protocols [Suns 83]. Works on performance analysis of protocols, however, have not addressed the issue of analyzing their timing requirements and have used manual analyses of their performance measures, (see for example [Tows 79, Bux 82]). Analyzing timing requirements of protocols is very important since it ensures their efficient performance and yields optimal settings of protocol parameters. Manual analysis of performance measures analyses are protocol-dependent (that is, a completely new analysis is required for every protocol examined), follow ad-hoc techniques, and cannot be integrated with other tools in protocol development environments.

In this paper, we propose a methodology for **formally** specifying and **automatically** analyzing timing requirements and performance measures of protocols. This combination is natural since both problems require a timing model of protocols. The methodology is implemented in ANALYST: an automated protocol performance analyzer. ANALYST runs on a VAX 750 under UNIX* 4.2bsd and is currently being ported to a personal workstation with advanced graphics capabilities.

*UNIX is a trademark of AT&T

A graphics interface to the environment is partially developed.

The paper is organized as follows. In section 2, the methodology is described and demonstrated using a simple connection establishment protocol. In section 3, the methodology is also applied to a data transfer protocol. Finally, a summary of the results is presented in section 4.

2. A Methodology for Specification-Based Performance Analysis of Protocols

2.1. An Algebraic Specification Method

In this section, an algebraic specification method for capturing the communication behavior of protocols will be introduced. Addressing between processes is assumed to be *one-to-one* meaning each message has unique sender and receiver processes. Inter-process communications are *synchronous* which means that a sender (receiver) process issuing a send (receive) event is blocked until the receiver (sender) process is ready to receive (send). That is, until the sender and receiver processes rendezvous.

The communication behavior of such processes can be described as a tree (a *communication tree*). Given

- A finite set of *identifiers* $\mathcal{I}^+ = \{A, B_1, SENDER, ...\}$ whose elements are character strings starting with an uppercase letter.

- A special identifier "$" denoting *deadlock*, $\mathcal{I} = \mathcal{I}^+ \cup \{\$\}$.

- A finite set of event *names* $\mathcal{N} = \{a, b', msg, ...\}$ whose elements are character strings starting with a lowercase letter. An event *name* preceded by "!", "?", or "&" corresponds to a send, receive, or rendezvous event, respectively. Let $\mathcal{E}$ denote the set of send events, receive events, and rendezvous events.

Definition 1: A Communication Tree (CT) is a finite rooted tree such that

1. The branches are labeled with events from $\mathcal{E}$.

2. The root and some internal nodes are labeled by unique identifiers from $\mathcal{I}^+$.

3. The leaves are all labeled from $\mathcal{I}$.

A tree is *complete* iff any leaf label, except deadlock, appears as a label of an internal node. Throughout the rest of the paper, we consider only complete trees.

A *behavior sequence* of a given CT is a possibly infinite sequence of events formed by traversing a path starting from the root and continuing until a leaf is reached. If the leaf is deadlocked, the behavior sequence *terminates*; otherwise, traversal continues from the tree rooted with the leaf label. A CT is a terminating tree iff there exists a behavior sequence of the tree that terminates. Otherwise, it is *cyclic*. Let the set of all behavior sequences of a CT rooted at A be referred to as $Seq[A]$.

Given a process CT, the execution of the process starts at the root and proceeds as follows: a branch followed by a node indicates that the event labeling the branch is *sequentially* followed by the behavior represented by the node. Several trees connected to a node indicates that any of them can be executed *non-deterministically.*

Consider, for example, a simple connection establishment protocol between a terminal process T and a network process N described in [Rudi 85]. The two processes communicate through two half-duplex, FIFO transmission channels: R from terminal

to network and I from network to terminal. The channels are assumed to be reliable, i.e., they do not lose or corrupt messages. If a message m is received at one end of a channel, then let m' denote its copy delivered at the other end.

Initially, the terminal and network processes are ready for connection, and the two channels are empty. The terminal either sends a call-request message ($!req$) to R or receives an incoming-call message ($!inc'$) from I, upon which it becomes connected. Subsequent incoming-call messages received are then ignored. The terminal can terminate the call by sending a terminate message ($!term$) and then becomes ready again. Fig. 1 depicts a CT that describes the communication behavior of the terminal process. Channel R accepts call-request and terminate messages from T and delivers copies of them to N. Similarly, channel I accepts incoming-call messages from N and delivers copies of them to T. For the sake of simplifying the analysis of the protocol, we assume that at most one call-request message and one terminate message reside in R at any time, and that one incoming-call message resides in I at any time. The network becomes connected if it either receives a call-request message ($!req'$) from R or sends an incoming-call message ($!inc$) to I. Subsequent call-request messages received are then ignored. The network becomes ready again if it receives a terminate message ($?term'$) from R. Any terminate messages received when the network is ready are ignored. Fig. 2 depicts the configuration of these four processes and the ports through which they communicate.

CTs can be formally defined using universal algebra [Grat 68].

Definition 2: An expression E in the algebra of CTs is either

1. $I \in \mathcal{I}$,

2. $e \bullet \mathrm{E}$ (sequential composition, $e \in \mathcal{E}$),
3. $\mathrm{E} + \mathrm{E}$ (non-deterministic composition),

   or

4. $\mathrm{E} \mid \mathrm{E}$ (concurrent composition).

Concurrently composing two CTs (expressions) produces a *composite* CT (expression). A composite CT informally includes a rendezvous event for every pair of corresponding send and receive events, and a shuffling of other events belonging to the two CTs. A formal definition of the concurrent composition is given in the next section.

Let the set of expressions in the algebra of CTs be denoted by $\mathcal{X}$. The meaning of operations in an algebra are described by *equational axioms*. For example, the following is an axiom of the algebra of CTs introduced above: $\mathrm{E} + \$ \equiv \mathrm{E}$, $\mathrm{E} \in \mathcal{X}$ (deadlock axiom), where $\equiv$ is a relation that defines equivalence of expressions. For a complete set of the axioms of the algebra of CTs and a definition of the equivalence relation, see [Noun 86].

The communication behavior of a process can be specified algebraically by either a single expression or a set of equations of the form $I = \mathrm{E}$, $I \in \mathcal{I}^+$ and $\mathrm{E} \ \#\in\# \ \mathcal{X}$. Such an equation corresponds to a CT whose root is labeled I and behavior is described by E. The algebraic specifications of the processes in the connection establishment protocol are given by

PROCESS T
$T = !req \cdot C_1 + ?inc' \cdot C_1$
$C_1 = ?inc' \cdot C_1 + !term \cdot T$
END

PROCESS N
$N = ?req' \cdot C_2 + !inc \cdot C_2 + ?term \cdot N$
$C_2 = ?req' \cdot C_2 + ?term \cdot N$
END

PROCESS R
$R = ?req \cdot R_1 + ?term \cdot R_2$
$R_1 = !req' \cdot R + ?term \cdot R_3$
$R_2 = !term' \cdot R$
$R_3 = !req' \cdot R_2$
END

PROCESS I
$I = ?inc \cdot I_1$
$I_1 = !inc' \cdot I$
END

Let the *scope* of communication between two processes be a set that includes the *names* of events with which they communicate, $name(!e) = name(?e) = e$. The *scope* of a pair of identifiers is equal to the *scope* of the processes they belong to. Events a and b are *co-events* iff they have the same *names* and one is a send event and the other a receive event.

A protocol can be specified by a list of processes and the *scope* between each pair of communicating processes. The specification of the connection establishment protocol is given by

PROTOCOL Connection Establishment: T,R,N,I
$scope(T,R) = \{req,term\}$ $scope(R,N) = \{req',term'\}$
$scope(N,I) = \{inc\}$ $scope(I,T) = \{inc'\}$
END

2.2. Concurrent Composition

The concurrent composition operation, informally described above, is defined recursively in terms of the sequential composition operation and an *n-ary* generalization of the non-deterministic operation $\sum_{i=1}^{n}$ by the following axiom:

Concurrent Composition Axiom

Let $A = \sum_{i=1}^{n} a_i \bullet A_i$ and $B = \sum_{j=1}^{m} b_j \bullet B_j$

then,

(i) $A \mid B = AB$

$$= \sum a_i \bullet (A_i \mid B) \qquad \forall\ a_i \text{ such that } name(a_i) \notin scope(A,B)$$

$$+ \sum b_j \bullet (A \mid B_j) \qquad \forall\ b_j \text{ such that } name(b_j) \notin scope(A,B)$$

$$+ \sum \&e \bullet (A_i \mid B_j) \qquad \forall\ a_i \text{ and } b_j \text{ such that}$$
(1) $name(a_i)=name(b_j)=e \in scope(A,B)$
(2) a_i and b_j are *co–events*

(ii) $A \mid \$ = \$ \mid A = A$

The concurrent behavior of a protocol can be obtained through the concurrent composition of the processes involved in it. The *scope* of a pair of processes of which at least one is composite can be computed according to the following rules.

(i) $scope(A,B) = scope(B,A)$
(ii) $scope(A \mid B,C) = scope(A,C) \cup scope(B,C)$
(iii) $scope(A \mid B,C \mid D) = scope(A,C) \cup scope(B,C) \cup scope(A,D) \cup scope(B,D)$ (2.1)

An automated concurrent composition is supported by ANALYST. The concurrent behavior of the given connection establishment protocol is computed by first concurrently composing process R with process T and process I with process N. The resulting composite processes RT and NI are then concurrently composed to produce the concurrent behavior $INRT$ of the connection establishment protocol shown in Fig 3.

The specification of $INRT$ includes 25 equations and on the average each has two possible behavior sequences. This explosion in the number of possible behavior sequences is due to *call collisions* and *premature call terminations*. Call collisions occur when both the terminal and the network attempt to initialize a call

simultaneously. Premature termination of calls occur when after a call collision the terminal terminates the call before receiving the incoming-call message. The terminal assumes that this incoming-call message is a request for another connection and a second terminate is then required to end this second unnecessary connection. The performance of these behavior sequences in which there are no call collisions or premature terminations of calls will be analyzed in the next section.

Generally, a protocol designer may be interested not only in the entire concurrent behavior of a protocol, but also in certain behavior sequences belonging to it or in segments of behavior sequences that terminate with some event. The two functions introduced next can be used for isolating interesting certain behavior sequences or segments of behavior sequences.

Given $A = \sum_{i=1}^{n} a_i \bullet A_i$, let the *choice set* $CH[A] = \{a_i, i=1,...,n\}$. In a CT rooted at A, $CH[A]$ denotes the set of events that label the outgoing branches connected to A. Let $\mathcal{P}$ represent the set that includes all pairs (e, I), where $e \in \mathcal{E}$ and $I \in \mathcal{I}$.

Definition 3: *Terminating* is a function: $\mathcal{X} \times \mathcal{P} \rightarrow \mathcal{X}$ such that

$$Terminating[A, P \subseteq \mathcal{P}] = \sum_{i \neq j} a_i \bullet Terminating[A_i, P] + a_j \bullet \$ \qquad \forall\ (a_j, A_j) \in P \text{ and } a_i, a_j \in CH[A]$$

$Terminating[A, P]$ maps the CT rooted at A to another CT identical to the former, with the exception that for every branch labeled e connected to a node labeled I where $(e, I) \in P$, then node I is labeled with a "$' instead. For example, the CT corresponding to $A = \&a \bullet A + \&b \bullet A$ and the CT corresponding to $A|_T = Terminating[A, \{(\&b, A)\}]$ are depicted in Fig. 4.

Consider the concurrent behavior *INRT* of the connection establishment protocol. This behavior is cyclic describing several connections between the terminal and network processes. The *Terminating* function can be used to compute segments of *Seq*[*INRT*] that describe the concurrent behavior of the protocol during the course of one connection. This terminating behavior $INRT|_T$ starts when all four processes are in the initial state and ends when the first terminate message is delivered to the network, and the four processes are in their initial state. $INRT|_T$ is formally specified as

$$INRT|_T = Terminating[INRT, \{(\&term, INRT)\}] \tag{2.2}$$

Fig. 5 depicts $INRT|_T$ obtained using ANALYST.

Let $\mathcal{S}$ denote the set of all event pairs.

Definition 4: *Time Constraint* TC is a function: $\mathcal{X} \times \mathcal{S} \rightarrow \mathcal{X}$ such that

$$TC[A, S \subseteq \mathcal{S}] = \sum_{i \neq j} a_i \bullet TC[A_i, S] \qquad \forall (a_k, a_j) \in S \text{ where } a_k, a_j \in CH[A]$$

$TC[A, S]$ maps the CT rooted at A to another CT identical to the former, with the exception that for every node with two outgoing branches labeled a_1, a_2 where $(a_1, a_2) \in S$, then the branch labeled a_2 and its hanging subtree are pruned. For $(a_1, a_2) \in S$, event a_1 is said to *have precedence* over event a_2 so that whenever there is a choice between them, the possibility of a_2 occurring is excluded. For example, the CT corresponding to $A = \&a \bullet A + \&b \bullet A$ and the CT corresponding to $A|_{TC} = TC[A, \{(\&a, \&b)\}]$ are depicted in Fig. 6.

The TC function can be used to compute those behavior sequences of the connection establishment protocol, denoted by $INRT|_{TC1}$, in which there are no call collisions. A call collision can be avoided if in $INRT|_T$, $\&req'$ has precedence over $\&inc$ and $\&inc'$ has precedence over $\&req$. Therefore, $INRT|_{TC1}$ is formally specified

by

$$INRT|_{TC1} = TC[INRT|_T, \{(\&inc', \&req), (\&req', \&inc)\}] \quad (2.3)$$

which is computed automatically using ANALYST to give the specification shown in Fig. 7.

Behavior $INRT|_{TC2}$ describing connections without premature termination of connection can be computed from $INRT|_T$ by having both *&inc* and *&inc'* have precedence over *&term*. $INRT|_{TC2}$ is formally specified by

$$INRT|_{TC2} = TC[INRT|_T, \{(\&inc, \&term), (\&inc', \&term)\}] \quad (2.4)$$

which is computed automatically using ANALYST to give the behavior in Fig. 8.

2.3. Specification-Based Performance Analysis of Protocols

In examining the performance of protocols, the specification and analysis of timing requirements and performance measures of protocols will be addressed. These two aspects of protocol performance are based on a timing model of protocols.

2.3.1. A Timing Model of Protocols

Assume that the observer of the concurrent behavior of a protocol can record the occurrence times of events starting from some initial time. The occurrence time of a send or receive event is the time at which it is offered. The occurrence time of a rendezvous event is the time of completing the rendezvous interaction. Let these occurrence times be modeled as continuous random variables.

Consider the execution of behavior $A = \sum_{i=1}^{n} a_i \cdot A_i$ starting from initial time t_0. Due to the non-deterministic nature of protocol behavior, at each occurrence t_i time there is a choice of events that might occur. At t_1, any event belonging to $CH[A] = \{a_1,...,a_n\}$ might occur. Let the *derivative* of A with respect to event e:

$\partial_e(A) = A_i$ iff $e = a_i$, $i=1,...,n$, or otherwise undefined. For A given above, $\partial_{e_1}(A) = A_1$.

Generally, if an event e occurs at time t_i, then the expression observed at t_i $\mathcal{E}_i$ is equal to $\partial_e(\mathcal{E}_{i-1})$ and the choice set $CH[\mathcal{E}_i]$ is *enabled*. This means that at t_{i+1}, any event belonging to this choice set might occur and the cycle continues until termination. Only events belonging to a choice set enabled at t_i can occur at t_{i+1}. Note that each occurrence time t_i has a collection of choice sets associated with it for every event that might occur at t_{i-1}. The CT describing the behavior of A and the corresponding timing behavior are depicted in Fig. 9.

Such protocol timing behavior is best modeled as a *marked point process* [Snyd 75] $[\{t_i, M_i\}, i > 0]$, where t_i is the i-th occurrence time and M_i is the i-th *mark* denoting the set of possible events that might occur at t_i. Each mark M_i is a collection of choice sets. It can be computed recursively by

$$M_i = \{CH[\partial_e(\mathcal{E}_{i-1})] \,,\, e \in M_{i-1}\}$$
$$M_0 = \emptyset \qquad (2.5)$$

Computing statistics of the occurrence times of such a marked point process is very complex since these occurrence times depend on the marks associated with them. The computations are simplified by assuming that the occurrence time of an event, measured from the time its choice set is enabled until it occurs, is independent of the other events in it's choice set. Suppose at some t_{i-1}, $\mathcal{E}_{i-1} = A$ thus enabling the choice set $CH(\mathrm{A})$ at the next occurrence time t_i. In this case, the events belonging to $CH(\mathrm{A})$ are actually *contending* with each other for occurrence at t_i. Let r_e denote the occurrence time of event e.

Lemma 1

1. The probability that event $e \in CH[A]$ occurs at t_i is given by
$$\text{Prob}(r_e = Min\{r_{e_j},\ e_j \in (CH[A] - \{e\})\}) \tag{2.6}$$
2. The occurrence time t_i is given by
$$t_i = t_{i-1} + Min\{r_{e_j},\ e_j \in CH[A]\} \tag{2.7}$$

Proof: See [Noun 84].

The mean and variance of occurrence times are of special interest since they are often required for analyzing the performance of protocols. In addition, the non-deterministic behavior of protocols can be described quantitatively by the probability of occurrence of its various behavior sequences.

Consider a terminating expression C. Given an expression A such that $Seq[A] \subseteq Seq[C]$, then *duration* of A relative to C is defined as the length of time starting when $CH[C]$ is enabled until A terminates. Attributes of the timing model of A and C are defined next.

Definition 5: A *probability attribute* **P** is a function: $\mathcal{X} \times \mathcal{X} \rightarrow [0,1]$, such that $\mathbf{P}_C(A)$ is the probability of A over the sample space $Seq[C]$.

Definition 6: A *mean-time attribute* **M** is a function: $\mathcal{X} \times \mathcal{X} \rightarrow R^+$, such that $\mathbf{M}_C(A)$ is the mean of the duration time of A relative to C.

Definition 7: A *variance-time attribute* **V** is a function: $\mathcal{X} \times \mathcal{X} \rightarrow R^+$, such that $\mathbf{V}_C(A)$ is the variance of the duration time of A relative to C.

Let $F_e(t)$ denote the probability distribution of the occurrence time of event e. The following theorems define mappings from operations in the algebra of CTs to operations on the attributes.

Theorem 1

$$\text{P1.}\quad \mathbf{P}_C(a\bullet A) = \mathbf{P}_C(a)\cdot \mathbf{P}_{\partial_a(C)}(A)$$

$$\text{P2.}\quad \mathbf{P}_C(A + B) = \mathbf{P}_C(A) + \mathbf{P}_C(B)$$

$$\text{P3.}\quad \mathbf{P}_C(a) = \int_0^\infty \prod_{e_i \in CH(C), e_i \neq a} [1 - F_{e_i}(t)]\, dF_a(t) \qquad \text{iff } a \in CH[C]$$

$$\text{P4.}\quad \mathbf{P}_C(\$) = 0 \qquad \text{iff } C \neq \$$$

Proof: P1 follows from noting that $\mathbf{P}_{\partial_a(C)}(A)$ is a conditional probability on the occurrence of a (since the choice set of $\partial_a(C)$ is only enabled after a occurs). P2 follows from noting that possible behaviors in a non-deterministic choice are mutually exclusive. P3 follows from lemma 1(i). P4 follows from the deadlock axiom given in section 2.1 and from definition 5 which indicates that $\mathbf{P}_A(A)$ is 1 for any A.

Theorem 2

$$\text{M1.}\quad \mathbf{M}_C(a\bullet A) = \mathbf{M}_C(a) + \mathbf{M}_{\partial_a(C)}(A)$$

$$\text{M2.}\quad \mathbf{M}_C(a\bullet A + b\bullet B) = \frac{\mathbf{P}_C(a)\cdot \mathbf{M}_C(a\bullet A) + \mathbf{P}_C(b)\cdot \mathbf{M}_C(b\bullet B)]}{\mathbf{P}_C(a + b)}$$

$$\text{M3.}\quad \mathbf{M}_C(a) = \int_0^\infty \prod_{e_i \in CH(C)} [1 - F_{e_i}(t)]dt \qquad \text{iff } a \in CH[C]$$

$$\text{M4.}\quad \mathbf{M}_C(\$) = 0$$

Proof: Proof of M1 and M2 is straightforward. M3 follows from lemma 1(ii). M4 follows from the definition of deadlock behavior.

Theorem 3

$$\text{V1.}\quad \mathbf{V}_C(a\bullet A) = \mathbf{V}_C(a) + \mathbf{V}_{\partial_a(C)}(A)$$

$$\text{V2.}\quad \mathbf{V}_C(a\bullet A + b\bullet B) = \frac{\mathbf{P}_C(a)\cdot [\mathbf{V}_C(a\bullet A) + \mathbf{M}^2_C(a\bullet A)] + \mathbf{P}_C(b)\cdot [\mathbf{V}_C(b\bullet B) + \mathbf{M}^2_C(b\bullet B)]}{\mathbf{P}_C(a + b)} - \mathbf{M}^2_C(a\bullet A + b\bullet B)$$

V3. $\mathbf{V}_C(a) = 2\int_0^\infty t \prod_{e_i \in CH(C)} [1 - F_{e_i}(t)]\, dt - \mathbf{M}_C^2(a)$ iff $a \in CH[C]$

V4. $\mathbf{V}_C(\$) = 0$

Proof: Proof of M1 and M2 is straightforward. V3 follows from lemma 1(ii). V4 follows from the definition of deadlock behavior.

For a detailed proof of how P3, M3, and V3 follow from lemma 1, see [Noun 84].

To simplify the computations involved in P3, M3, and V3, assume that all occurrence times of events are exponentially distributed. Let λ_e denote the exponential rate of the occurrence time of event e. P3, M3 and V3 are reduced to

$$\mathbf{P}_C(a) = \frac{\lambda_a}{\sum_{e_i \in CH(C)} \lambda_{e_i}} \tag{2.8}$$

$$\mathbf{M}_C(a) = \frac{1}{\sum_{e_i \in CH(C)} \lambda_{e_i}} \tag{2.9}$$

$$\mathbf{V}_C(a) = \frac{1}{(\sum_{e_i \in CH(C)} \lambda_{e_i})^2} \tag{2.10}$$

In summary, given an expression and the probability distributions of the occurrence times of all events belonging to it, the attributes of the timing model of this expression can be computed using the theorems above. The set of algebraic equations describing the expression is mapped to a set of simultaneous linear equations where the attributes are the variables. The mapping rules defined are

automated in ANALYST.

Given a recursively defined expression $A = X \bullet A + B$, we intuitively expect that the number of repetitions of behavior X is a random variable with a modified geometric distribution.

Corollary 3.1
Let $A = X \bullet A + B$ then,

$$\mathbf{M}_A(A) = \frac{\mathbf{P}_A(X)}{[1-\mathbf{P}_A(X)]} \cdot \mathbf{M}_A(X) + \mathbf{M}_A(B)$$

$$\mathbf{V}_A(A) = \frac{\mathbf{P}_A(X)}{[1-\mathbf{P}_A(X)]} \cdot \mathbf{V}_A(X) + \frac{1}{[1-\mathbf{P}_A(X)]} \cdot \mathbf{M}_A^2(a) + \mathbf{V}_A(B)$$

Proof: Using theorems 3.1, 3.2, and 3.3. (see [Noun 84] for a detailed proof).

2.3.2. Specification and Analysis of Timing requirements and Performance Measures

Any timing requirement or performance measure that can be specified in terms of the attributes of the timing model of protocols, can be automatically analyzed using ANALYST.

For example, consider the connection establishment protocol. To ensure that the protocol performs efficiently, its timing behavior should meet a timing requirement stating that the probability of those behavior sequences without premature termination of calls is maximized. Since behavior $INRT|_{TC2}$ given in Fig. 7, then the timing requirement is formally specified by

$$\mathbf{P}_{INRT|_T}(INRT|_{TC2}) \geq 1 - \epsilon \qquad (2.11)$$

where ϵ is a small probability error.

$\mathbf{P}_{INRT|_T}(INRT|_{TC2})$ is plotted versus the rate of termination of calls $\lambda_{\&term}$ in

Fig. 10. The figure illustrates how the probability of behaviors without premature termination decreases as the rate of termination increases. For the given data and $\epsilon = 0.05$, then the optimal rate of termination is given by

$$\lambda_{\&term} \leq 10.1 occurrences/sec. \tag{2.12}$$

An interesting measure of the performance of the connection establishment protocol is the probability of collisions p_c given by

$$p_c = 1 - \mathrm{P}_{INRT|_T^{(INRT|_{TC1})}} \tag{2.13}$$

In Fig. 11, p_c is plotted versus $\lambda_{\&req}$ for two different $\lambda_{\&inc}$. The figure shows that the probability of collisions is saturated for a wide range of $\lambda_{\&req}$. It decreases when the ratio $\frac{\lambda_{\&req}}{\lambda_{\&inc}}$ is very low or very high. For a certain $\lambda_{\&req}$, an increase in $\lambda_{\&inc}$ causes an increase in p_c.

3. An Example: A Simple Data Transfer Protocol

In this section, the proposed methodology will be used to analyze the performance of a simple data transfer protocol that employs stop-and-wait and retransmission on time-out strategies. Results obtained have been automatically computed using ANALYST.

3.1. An Algebraic Specification

The protocol involves four processes: a sender S, a receiver R, a sender-to-receiver transmission channel D, and a receiver-to-sender transmission channel A. The channels are assumed to be half-duplex and FIFO, but unreliable. Channel D accepts messages (msg) from the sender and delivers copies of them (msg') to the receiver. Channel A accepts acknowledgments (ack) from the receiver and delivers copies of them (ack') to the sender. The sender process is a composite process resulting from an original sender concurrently composed with a source of messages

and a time-out timer. Two rendezvous events: *&get* (get a message from the source) and *&tout* (time-out interrupt), are obtained from these compositions, respectively. In addition, the two channels result from concurrent composition with a loss process producing the rendezvous events *&lm* (loss of a message) and *&la* (loss of an acknowledgment).

The sender and receiver processes are initially ready for data transfer, and the two channels are empty. When the sender process S gets a message (*&get*) from the source, it sends it (*!msg*), and waits either for its acknowledgment or a time-out. If an acknowledgment is received (*?ack'*) from channel A, the sender becomes ready again. If time-out occurs, it re-transmits the same message and waits again for acknowledgment or time-out. The receiver process R loops through the following behavior: when it receives a message (*?msg'*) from channel D, it sends an acknowledgment *![ack]* to channel A and becomes ready again. For the sake of limiting the size of the concurrent behavior of the protocol, we assume an upper bound of 2 on the number of messages (acknowledgments) traveling through channels at any one time. Fig. 12 depicts the configuration of these processes and the ports through which they communicate.

The specifications of the protocol and the four processes involved in it are as follows:

PROTOCOL Data Transfer: S,D,R,A
$scope(S,D) = \{msg\}$ $scope(D,R) = \{msg'\}$
$scope(R,A) = \{ack\}$ $scope(A,S) = \{ack'\}$
END

PROCESS S
$S = \&get.T$
$T = !msg.W$
$W = \&tout.T + ?ack'.SEND$
End

PROCESS R
$R = ?msg'.R_1$
$R_1 = !ack.R$

PROCESS D
$D = ?msg.D_1$
$D_1 = !msg'.D + \&lm.D + ?msg.D_2$
$D_2 = !msg'.D_1 + \&lm.D_1$
END

PROCESS A
$A = ?ack.A_1$
$A_1 = !ack'.A + \&la.A + ?ack.A_2$
$A_2 = !ack'.A_1 + \&la.A_1$
END

The concurrent behavior of the protocol *ADRS* includes 53 equations each with, on the average, three possible behaviors sequences. Most of the behavior sequences belonging to this concurrent behavior are due to unnecessary re-transmissions of messages. An unnecessary re-transmission is due to a premature time-out i.e., time-out in cases when no message or acknowledgment is lost.

The concurrent behavior *ADRS* is cyclic describing transmissions of several messages. The behavior terminating with the first acknowledgment delivered to the sender, i.e., event $\&ack'$, is given by

$$ADRS\,|_{T1} = Terminating[ADRS, \{(\&ack', ***S)\}] \tag{3.1}$$

where "*" matches any identifier, and therefore "$***S$" would match any identifier in the concurrent behavior *ADRS* in which the identifier corresponding to the sender is S and the identifiers corresponding to the receiver and two channels are any identifier. $ADRS\,|_{T1}$ starts with event $\&get$. The behavior, denoted by $ADRS\,|_T$ that starts with sending a message at the sender and ends with the delivery of the first acknowledgement is given by

$$ADRS\,|_T = ADRT_T \tag{3.2}$$

The reader is referred to [Noun 86] for a complete listing of *ADRS* and $ADRS\,|_T$

3.2. Specification and Analysis of Timing Requirements and Performance Measures

Time-out is introduced in the data transfer protocol and other similar re-transmission protocols in order to recover from situations in which messages are lost. For these protocols to perform efficiently, their timing behavior should meet a timing requirement stating that

1. The probability of those behavior sequences corresponding to premature time-outs is minimized.

2. Time-out occurs promptly after a loss thus minimizing unnecessary delays.

Whenever time-out is contending with any other event in the protocol and time-out occurs, then this time-out is premature. The reason is that time-out should not occur before a loss or before allowing a message or its acknowledgment to reach its destination. The behavior, $ADRS\,|_{TC}$, in which a time-out occurs only after a loss can be computed from $ADRS\,|_{T}$ by having all other events in the protocol have precedence over *&tout*. These behavior sequences are formally specified by

$$ADRS\,|_{TC} = TC[ADRS\,|_{T}\{(\&lm,\&tout),(\&ack,\&tout),(\&la,\&tout),(\&msg,\&tout),(\&msg',\&tout),(\&ack',\&tout)\}] \quad (3.3)$$

The timing requirement stated above is formally specified by

$$\text{Minimize } 1 - \mathbf{P}_{ADRS\,|_{T}}(ADRS\,|_{TC}) \text{ and } \mathbf{M}_{ADRS\,|_{TC}}(ADRS\,|_{TC}) \quad (3.4)$$

$\lambda_{\&msg}$, $\lambda_{\&ack}$ represent message and acknowledgment transmission rates, respectively. For a transmission channel bandwidth b and message and acknowledgment length l, $\lambda_{\&msg}=\lambda_{\&ack} = b/l$. $\lambda_{\&msg'}$, $\lambda_{\&ack'}$ represent the sum of propagation and processing rates of messages and acknowledgments, respectively.

By varying the rate of time-out $\lambda_{\&tout}$, $\mathbf{M}_{ADRS\,|_{T}}(ADRS\,|_{T})$ is plotted against $1 - \mathbf{P}_{ADRS\,|_{T}}(ADRS\,|_{TC})$ in Fig. 14, for two different values of the rates of message and acknowledgment loss. Increasing the rate of time-out, causes an

increase in the probability of behaviors without premature time-outs. It also causes a decrease in the mean time of one transmission. For the same time-out rate, an increase in the rate of loss causes a decrease in the probability of premature and an increase in the mean time, as expected. This indicates a trade-off between the two goals of the specified timing requirement of the protocol. Some balanced then criterion has to be chosen instead and accordingly optimal time-out rate can be computed.

In Fig. 15, the probability of behaviors without premature time-outs is plotted against time-out rate for tow different rates of loss. The figure indicates that a change in the rate of loss does not affect the probability of premature time-outs, especially at very low and very high time-out rates. For a time-out rate of about 1, the probability of premature time-outs for both rates of error loss less than 0.87.

Subsequently, $\lambda_{\&tout}$ will be set at 1, and we assume that behavior $ADRS|_{T1}$ is approximated by the much simpler behavior

$$C = \&get.ADRS|_{TC} \tag{3.5}$$

Consider the timing behavior of C depicted in Fig. 16 starting at t_0 when a message arrives at the source. In the figure, r_w, r_v, and r_f are the *waiting time*, *virtual transmission time*, and *transfer time*, respectively. The *waiting time* r_w is the time that a message arriving at the source when the protocol system is busy has to wait before being served by the protocol. The protocol system is busy if the sender is waiting for the acknowledgment of a previously sent message. The *virtual transmission time* r_v is the time starting with sending a message at the sender until receiving its acknowledgment. Let t_w, t_v, and t_f denote the mean

of those times, respectively. Let σ_v denote the variance of τ_v and λ denote the rate of message arrivals. From eq. 3.5 and the specification of $ADRS|_{TC}$ in Fig. 13,

$$t_v = \mathbf{M}_C(ADRT|_{TC}) \tag{3.6}$$
$$\sigma_v = \mathbf{V}_C(ADRT|_{TC}) \tag{3.7}$$

using the Pollaczek-Khinchine formula [Klei 75] we get

$$t_w = \frac{\lambda \cdot [t_v^2 + \sigma_v]}{2 \cdot [1 - \lambda t_v]} \tag{3.8}$$

and from Fig. 16,

$$t_f = t_w + t_v \tag{3.9}$$

$\lambda_{\&lm}$ is related to the bit error probability p_{ber} of the transmission channels, as given by

Prob. of message loss $= 1 - (1 - p_{ber})^l$
and from $ADRS|_{TC}$
Prob. of message loss $= \mathbf{P}_{AD1RW|_{TC}}(\&lm)$
which using P1 in Theorem 3.1 is equal to

$$\frac{\lambda_{\&lm}}{\lambda_{\&msg} + \lambda_{\&lm}} \tag{3.10}$$

The same applies to the rate of acknowledgment loss assuming that p_{ber} for both channels are equal.

A plot of t_f against p_{ber} for several channel bandwidths is given in Fig. 17. The figure shows that t_f is not affected by change in p_{ber} for small p_{ber}. A similar result has been obtained by manual analysis [Tows 79].

Maximum throughput TH is the average transmission rate of useful data between the sender and receiver (i.e., excluding any acknowledgments or re-transmissions required by the protocol) achieved when the sender always has a new message to send [Bux 80]. Since only one message occupies the protocol system at a time,

TH is given by

$$TH = l/t_v \tag{3.11}$$

A plot of TH against message length l for several p_{ber} in Fig. 18 shows a degradation in TH for large values of l. This is due to the increase in probability of channel loss for large l, as indicated in eq. 3.10. As p_{ber} decreases this degradation is evident with very long messages and TH saturates for a range of l. A similar result has been obtained by Bux et al., [Bux 80] in analyzing the effect of changing the channels error bit probability on the throughput of a more complex data transfer protocol: a class of HDLC procedures.

In summary, for terrestrial channels (where p_{ber} is very small e.g., 10^{-10}) the maximum throughput of the given data transfer protocol only suffers degradation at large message lengths. The mean transfer time of the protocol is also not affected by any slight change in bit error probability. However, for satellite channels with higher bit error probability, all the protocol parameters should be considered.

4. Summary

A methodology for automatically analyzing the performance of protocols was introduced. An algebraic method is used for specifying the communication behavior of protocols. Timing behaviors of protocols were modeled as marked point processes whose probability, mean-time, and variance time attributes are homomorphic images of expressions in the specification algebra. These attributes can be used to formally specify timing requirements and performance measures of protocols. Given algebraic specifications of processes involved in a protocol and the rates of event in its concurrent behavior, timing requirements and perfromance measures of the protocol can be analyzed.

The methodology is implemented in ANALYST which has been used in automatically obtaining results reported in the paper. Analyzing the performance of a connection establishment protocol and a data transfer protocol, yielded optimal settings of critical system parameters and insights to the efficiency of the protocols.

References

[Bux 80] W.Bux,K.Kummerle, and H.Truong.
Balanced HDLC Procedures: A Performance Analysis.
IEEE Transactions on Communications COM-28(11):1889-1898, November, 1980.

[Bux 82] W.Bux and K.Kummerle.
Data Link-Control Performance: Results Comparing HDLC Operational Modes.
Computer Networks 6:37-51, 1982.

[Grat 68] G.Gratzer.
Universal Algebra.
Springer-Verlag, 1968.

[Klei 75] L.Kleinrock.
Queueing Systems.
Wiley Interscience, 1975.

[Noun 84] N.Nounou and Y.Yemini.
Algebraic Specification-Based Performance Analysis of Communication Protocols.
In *Proceedings of the Fourth IFIP International Workshop on Protocol Specification, Testing and Verification.* North-Holland, June, 1984.

[Noun 86] N.Nounou.
Automated Perfroamnce Analysis of Protocol.
PhD thesis, Columbia Univ., 1986.

[Rudi 85] H.Rudin.
An Informal View of Formal Protocol Specification.
IEEE Communications Magazine 23(2):46-52, March, 1985.

[Snyd 75] D.Snyder.
Random Point Processes.
Wiley-Interscience, 1975.

[Suns 83] C.Sunshine.
Experience with Automated Verification Systems.
In *Proceedings of the Third IFIP International Workshop on Protocol Specification, Testing and Verification.* 1983.

[Tows 79] D.Towsley and J.Wolf.
On the Statistical Analysis of Queue Lengths and Waiting Times for Statistical Multiplexers with ARQ Retransmission Schemes.
IEEE Transactions on Communications COM-27(4):693-702, April. 1979.

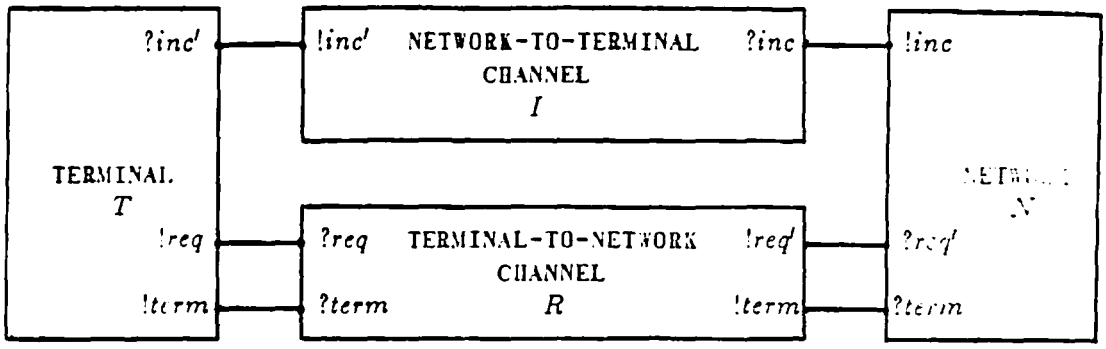

Figure 1: Configuration of the connection establishment protocol

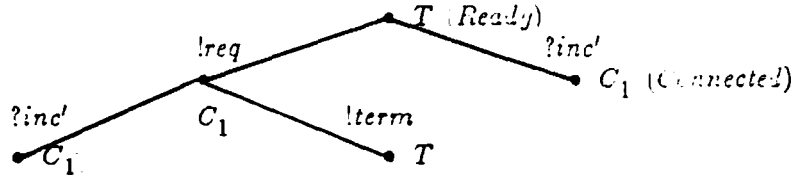

Figure 2: A CT describing the communication behavior of the terminal T in the connection establishment protocol

$$INRT = \&inc \cdot C_2RTI_1 + \&req \cdot C_1R_1NI$$
$$C_2RTI_1 = \&req \cdot C_1R_1C_2I_1 + \&inc' \cdot C_1RC_2I$$
$$C_1R_1NI = \&term \cdot INR_3T + \&req' \cdot C_1RC_2I + \&inc \cdot C_1R_1C_2I_1$$
$$C_1R_1C_2I_1 = \&inc' \cdot C_1R_1C_2I + \&term \cdot C_2R_3TI_1 + \&req' \cdot C_1RC_2I_1$$
$$C_1RC_2I = \&term \cdot C_2R_2TI$$
$$INR_3T = \&inc \cdot C_2R_3TI_1 + \&req' \cdot C_2R_2TI$$
$$C_1R_1C_2I = \&term \cdot C_2R_3TI + \&req' \cdot C_1RC_2I$$
$$C_2R_3TI_1 = \&req' \cdot C_2R_2TI_1 + \&inc' \cdot C_1R_3C_2I$$
$$C_1RC_2I_1 = \&inc' \cdot C_1RC_2I + \&term \cdot C_2R_2TI_1$$
$$C_2R_2TI = \&term' \cdot INRT$$
$$C_2R_3TI = \&req' \cdot C_2R_2TI$$
$$C_2R_2TI_1 = \&term' \cdot I_1NRT + \&inc' \cdot C_1R_2C_2I$$
$$C_1R_3C_2I = \&req' \cdot C_1R_2C_2I$$
$$I_1NRT = \&inc' \cdot C_1RNI + \&req \cdot C_1R_1NI_1$$
$$C_1R_2C_2I = \&term' \cdot C_1RNI$$
$$C_1RNI = \&term \cdot INR_2T + \&inc \cdot C_1RC_2I_1$$
$$C_1R_1NI_1 = \&inc' \cdot C_1R_1NI + \&term \cdot I_1NR_3T + \&req' \cdot C_1RC_2I_1$$
$$INR_2T = \&inc \cdot C_2R_2TI_1 + \&term' \cdot INRT$$
$$I_1NR_3T = \&inc' \cdot C_1R_3NI + \&req' \cdot C_2R_2TI_1$$
$$C_1R_3NI = \&req' \cdot C_1R_2C_2I + \&inc \cdot C_1R_3C_2I_1$$
$$C_1R_3C_2I_1 = \&inc' \cdot C_1R_3C_2I + \&req' \cdot C_1R_2C_2I_1$$
$$C_1R_2C_2I_1 = \&inc' \cdot C_1R_2C_2I + \&term' \cdot C_1RNI_1$$
$$C_1RNI_1 = \&inc' \cdot C_1RNI + \&term \cdot I_1NR_2T$$
$$I_1NR_2T = \&inc' \cdot C_1R_2NI + \&term' \cdot I_1NRT$$
$$C_1R_2NI = \&term' \cdot C_1RNI + \&inc \cdot C_1R_2C_2I_1$$

Figure 3: The concurrent behavior of the connection establishment protocol

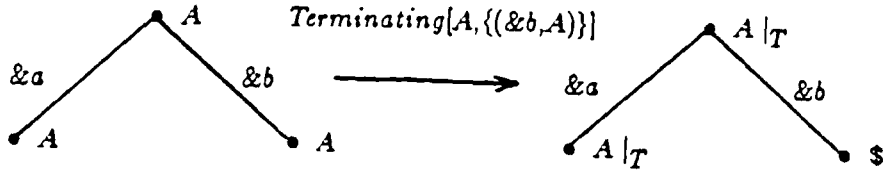

Figure 4: An illustration of the *Terminating* function

$$INRT\,|_T = \&inc \cdot C_2RTI_1\,|_T + \&req \cdot C_1R_1NI\,|_T$$
$$C_2RTI_1\,|_T = \&req \cdot C_1R_1C_2I_1\,|_T + \&inc' \cdot C_1RC_2I\,|_T$$
$$C_1R_1NI\,|_T = \&term \cdot INR_3T\,|_T + \&req' \cdot C_1RC_2I\,|_T + \&inc \cdot C_1R_1C_2I_1\,|_T$$
$$C_1R_1C_2I_1\,|_T = \&inc' \cdot C_1R_1C_2I\,|_T + \&term \cdot C_2R_3TI_1\,|_T + \&req' \cdot C_1RC_2I_1\,|_T$$
$$C_1RC_2I\,|_T = \&term \cdot C_2R_2TI\,|_T$$
$$INR_3T\,|_T = \&inc \cdot C_2R_3TI_1\,|_T + \&req' \cdot C_2R_2TI\,|_T$$
$$C_1R_1C_2I\,|_T = \&term \cdot C_2R_3TI\,|_T + \&req' \cdot C_1RC_2I\,|_T$$
$$C_2R_3TI_1\,|_T = \&req' \cdot C_2R_2TI_1\,|_T + \&inc' \cdot C_1R_3C_2I\,|_T$$
$$C_1RC_2I_1\,|_T = \&inc' \cdot C_1RC_2I\,|_T + \&term \cdot C_2R_2TI_1\,|_T$$
$$C_2R_2TI\,|_T = \&term' \cdot \$$$
$$C_2R_3TI\,|_T = \&req' \cdot C_2R_2TI\,|_T$$
$$C_2R_2TI_1\,|_T = \&term' \cdot I_1NRT\,|_T + \&inc' \cdot C_1R_2C_2I\,|_T$$
$$C_1R_3C_2I\,|_T = \&req' \cdot C_1R_2C_2I\,|_T$$
$$I_1NRT\,|_T = \&inc' \cdot C_1RNI\,|_T + \&req \cdot C_1R_1NI_1\,|_T$$
$$C_1R_2C_2I\,|_T = \&term' \cdot C_1RNI\,|_T$$
$$C_1RNI\,|_T = \&term \cdot INR_2T\,|_T + \&inc \cdot C_1RC_2I_1\,|_T$$
$$C_1R_1NI_1\,|_T = \&inc' \cdot C_1R_1NI\,|_T + \&term \cdot I_1NR_3T\,|_T + \&req' \cdot C_1RC_2I_1\,|_T$$
$$INR_2T\,|_T = \&inc \cdot C_2R_2TI_1\,|_T + \&term' \cdot \$$$
$$I_1NR_3T\,|_T = \&inc' \cdot C_1R_3NI\,|_T + \&req' \cdot C_2R_2TI_1\,|_T$$
$$C_1R_3NI\,|_T = \&req' \cdot C_1R_2C_2I\,|_T + \&inc \cdot C_1R_3C_2I_1\,|_T$$
$$C_1R_3C_2I_1\,|_T = \&inc' \cdot C_1R_3C_2I\,|_T + \&req' \cdot C_1R_2C_2I_1\,|_T$$
$$C_1R_2C_2I_1\,|_T = \&inc' \cdot C_1R_2C_2I\,|_T + \&term' \cdot C_1RNI_1\,|_T$$
$$C_1RNI_1\,|_T = \&inc' \cdot C_1RNI\,|_T + \&term \cdot I_1NR_2T\,|_T$$
$$I_1NR_2T\,|_T = \&inc' \cdot C_1R_2NI\,|_T + \&term' \cdot I_1NRT\,|_T$$
$$C_1R_2NI\,|_T = \&term' \cdot C_1RNI\,|_T + \&inc \cdot C_1R_2C_2I_1\,|_T$$

Figure 5: A terminating behavior of the connection establishment protocol describing one connection

$$INR_2\,|_{TC1} = \&inc \cdot C_2RTI_1\,|_{TC1} + \&req \cdot C_1R_1NI\,|_{TC1}$$
$$C_2RTI_1\,|_{TC1} = \&inc' \cdot C_1RC_2I\,|_{TC1}$$
$$C_1R_1NI\,|_{TC1} = \&term \cdot INR_2T\,|_{TC1} + \&req' \cdot C_1RC_2I\,|_{TC1}$$
$$C_1RC_2I\,|_{TC1} = \&term \cdot C_2R_2TI\,|_{TC1}$$
$$INR_2T\,|_{TC1} = \&req' \cdot C_2R_2TI\,|_{TC1}$$
$$C_2R_2TI\,|_{TC1} = \&term' \cdot \$$$

Figure 6: A behavior of the connection establishment protocol in which there are no call-collisions

$$INRT\,|_{TC2} = \&inc \cdot C_2RTI_1\,|_{TC2} + \&req \cdot C_1R_1NI\,|_{TC2}$$
$$C_2RTI_1\,|_{TC2} = \&req \cdot C_1R_1C_2I_1\,|_{TC2} + \&inc' \cdot C_1RC_2I\,|_{TC2}$$
$$C_1R_1NI\,|_{TC2} = \&req' \cdot C_1RC_2I\,|_{TC2} + \&inc \cdot C_1R_1C_2I_1\,|_{TC2}$$
$$C_1R_1C_2I_1\,|_{TC2} = \&inc' \cdot C_1R_1C_2I\,|_{TC2} + \&req' \cdot C_1RC_2I_1\,|_{TC2}$$
$$C_1RC_2I\,|_{TC2} = \&term \cdot C_2R_2TI\,|_{TC2}$$
$$C_1R_1C_2I\,|_{TC2} = \&req' \cdot C_1RC_2I\,|_{TC2} + \&term \cdot C_2R_2TI\,|_{TC2}$$
$$C_1RC_2I_1\,|_{TC2} = \&inc' \cdot C_1RC_2I\,|_{TC2}$$
$$C_2R_2TI\,|_{TC2} = \&term' \cdot \$$$
$$C_2R_2TI\,|_{TC2} = \&req' \cdot C_2R_2TI\,|_{TC2}$$

Figure 7: A behavior of the connection establishment protocol in which there are no premature terminations of calls

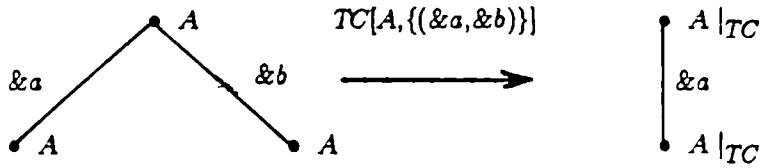

Figure 8: An illustration of the *TC* function

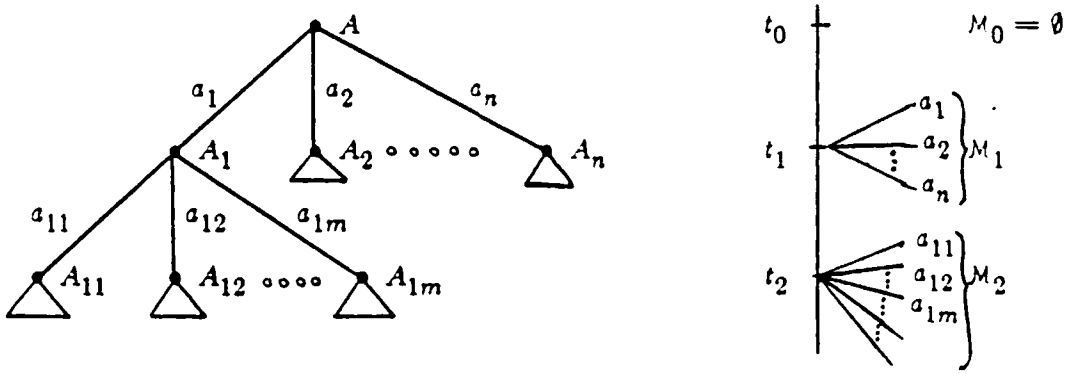

Figure 9: A CT and the corresponding timing behavior

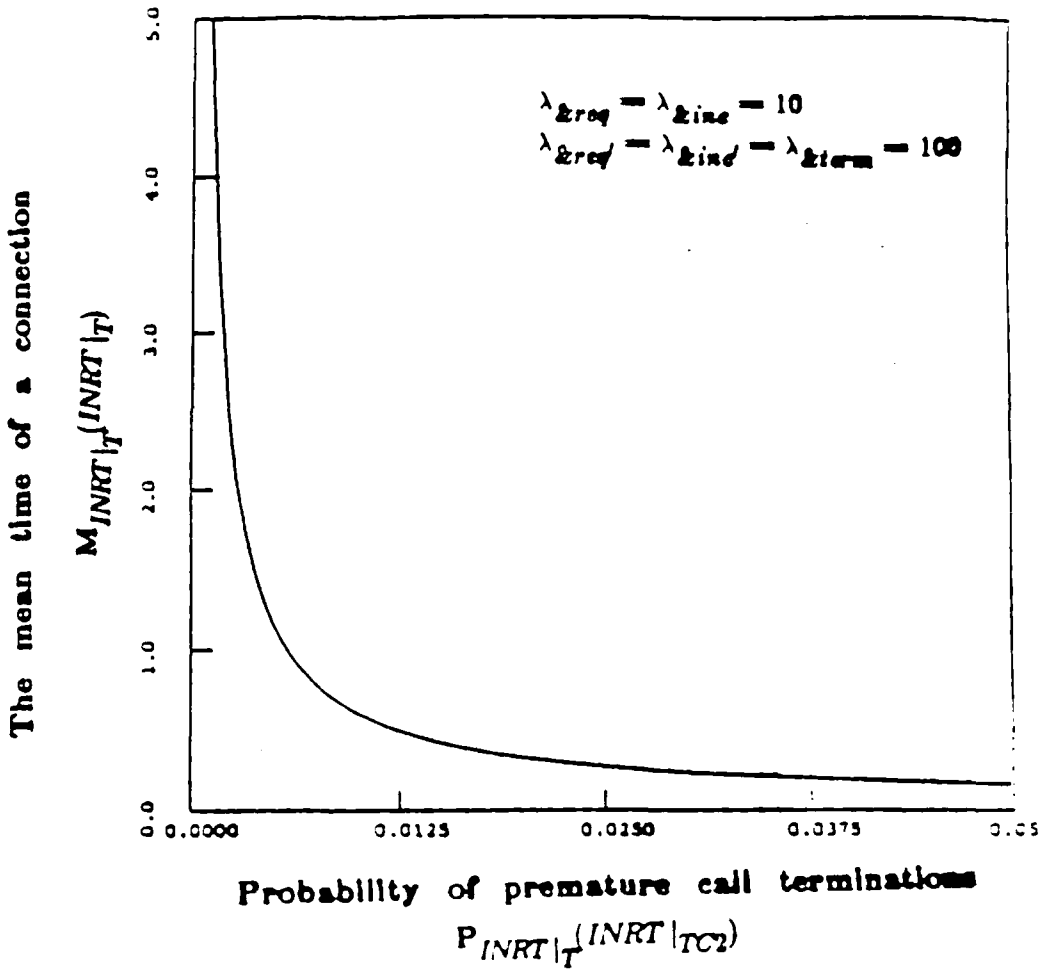

Figure 10: The mean time of a connection $M_{INRT|_T}(INRT|_T)$ versus the probability of premature termination $1 - P_{INRT|_T}(INRT|_{TC2})$

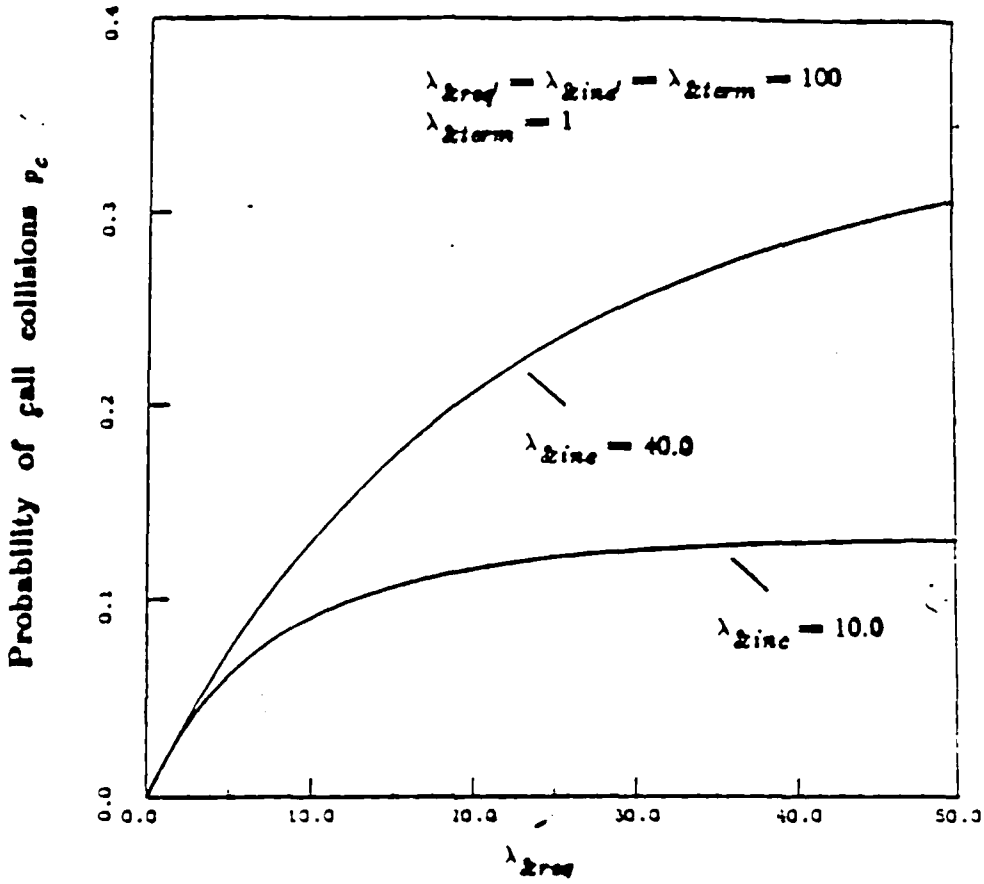

Figure 11: The probability of call collision p_c versus $\lambda_{\&req}$ for various $\lambda_{\&inc}$

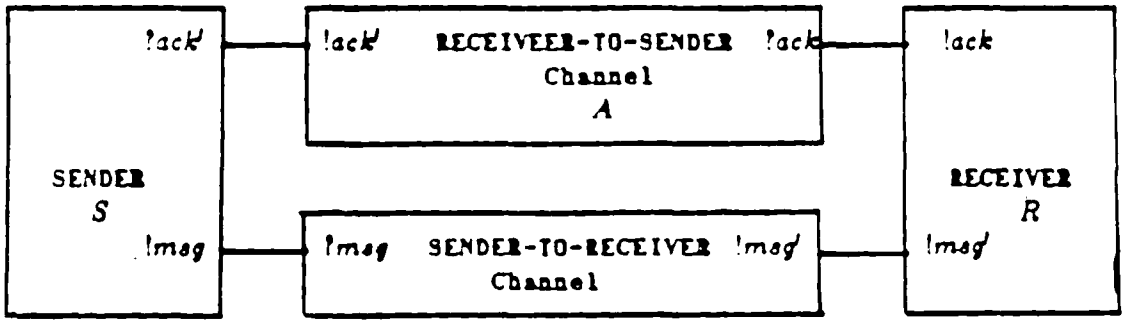

Figure 12: Configuration of the data transfer protocol

$$ADRS\,|_{TC} = \&get.ADRT\,|_{TC}$$
$$ADRT\,|_{TC} = \&msg.AD1RW\,|_{TC}$$
$$AD1RW\,|_{TC} = \&msg'.ADR1W\,|_{TC} + \&lm.ADRW\,|_{TC}$$
$$ADR1W\,|_{TC} = \&ack.A1DRW\,|_{TC}$$
$$ADRW\,|_{TC} = \&tout.ADRT\,|_{TC}$$
$$A1DRW\,|_{TC} = \&ack'.\$ + \&la.ADRW\,|_{TC}$$

Figure 13: A behavior of the data transfer protocol in which there are no premature time-outs

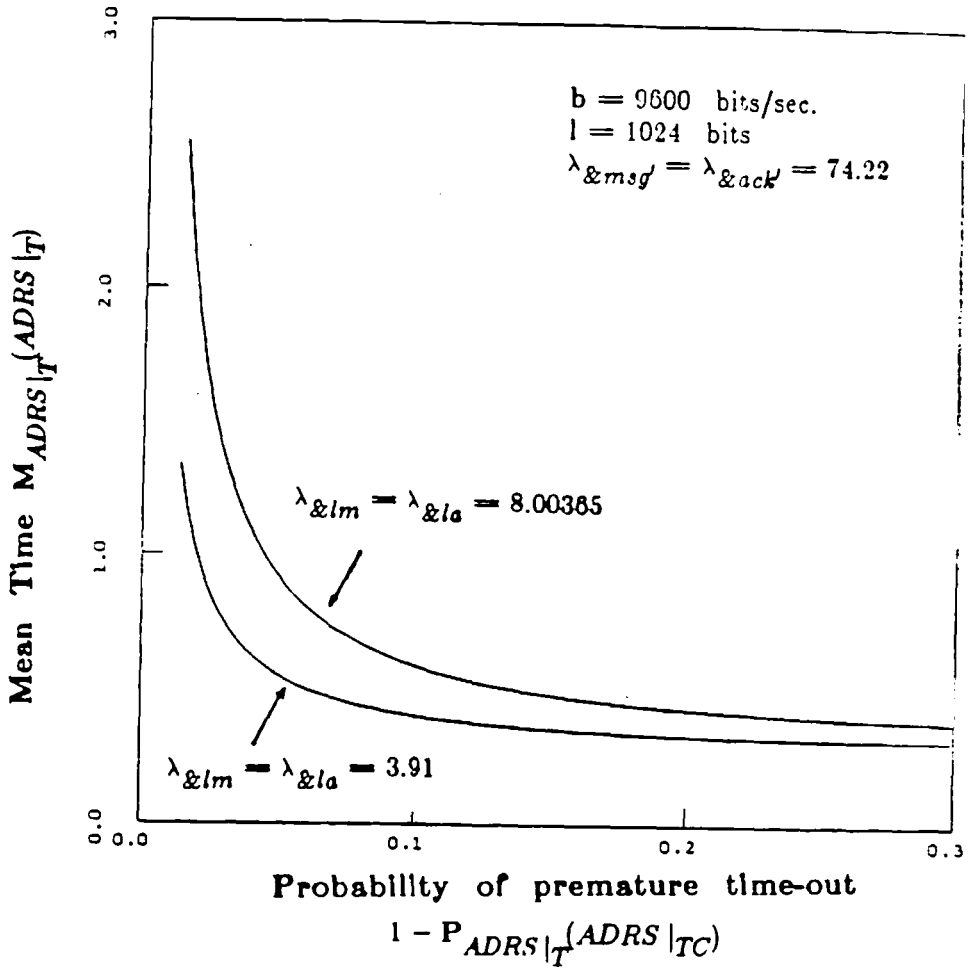

Figure 14: Mean time $M_{ADRS\,|_T}(ADRS\,|_T)$ versus the probability of premature time-out $1 - P_{ADRS\,|_T}(ADRS\,|_{TC})$

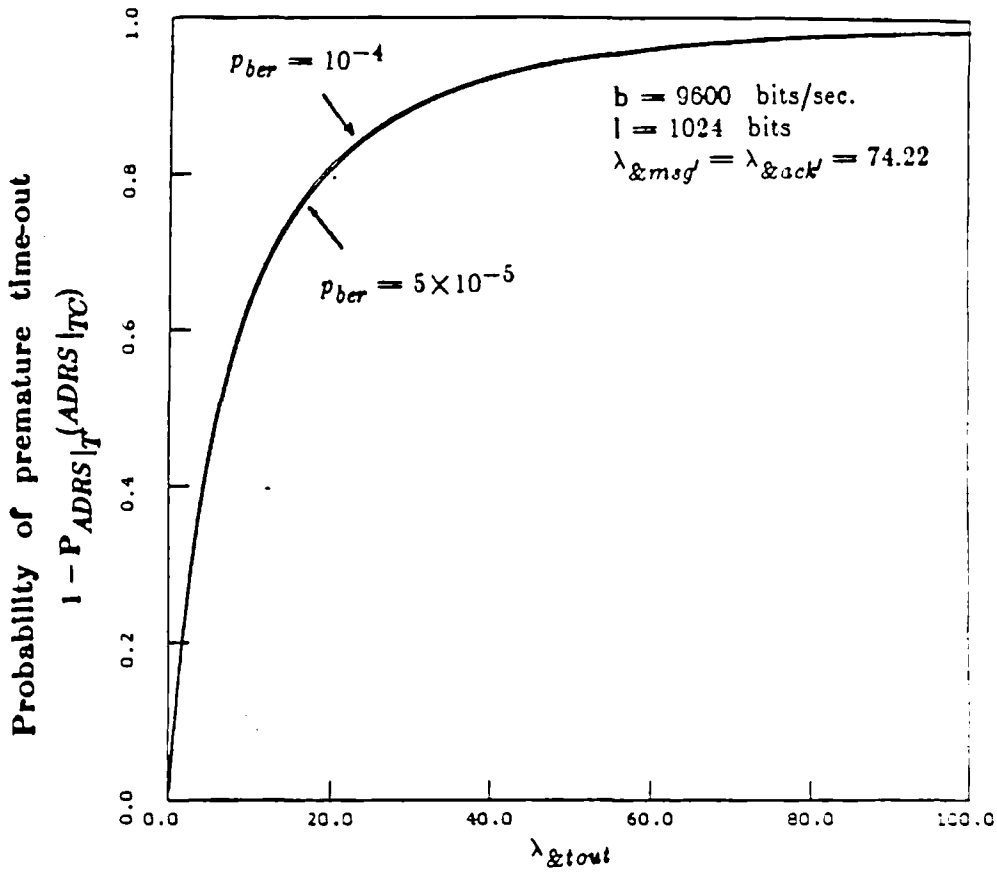

Figure 15: The probability of premature time-out $1 - P_{ADRS|_T}(ADRS|_{TC})$ versus rate of time-out $\lambda_{\&tout}$ for two different rate of loss $\lambda_{\&lm}$ and $\lambda_{\&la}$

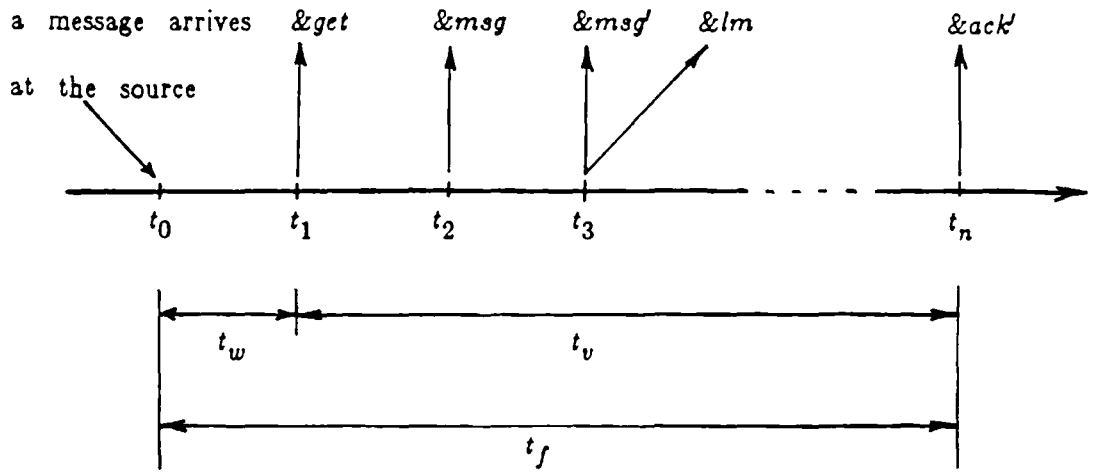

Figure 16: A timing behavior of $ADRS|_T$

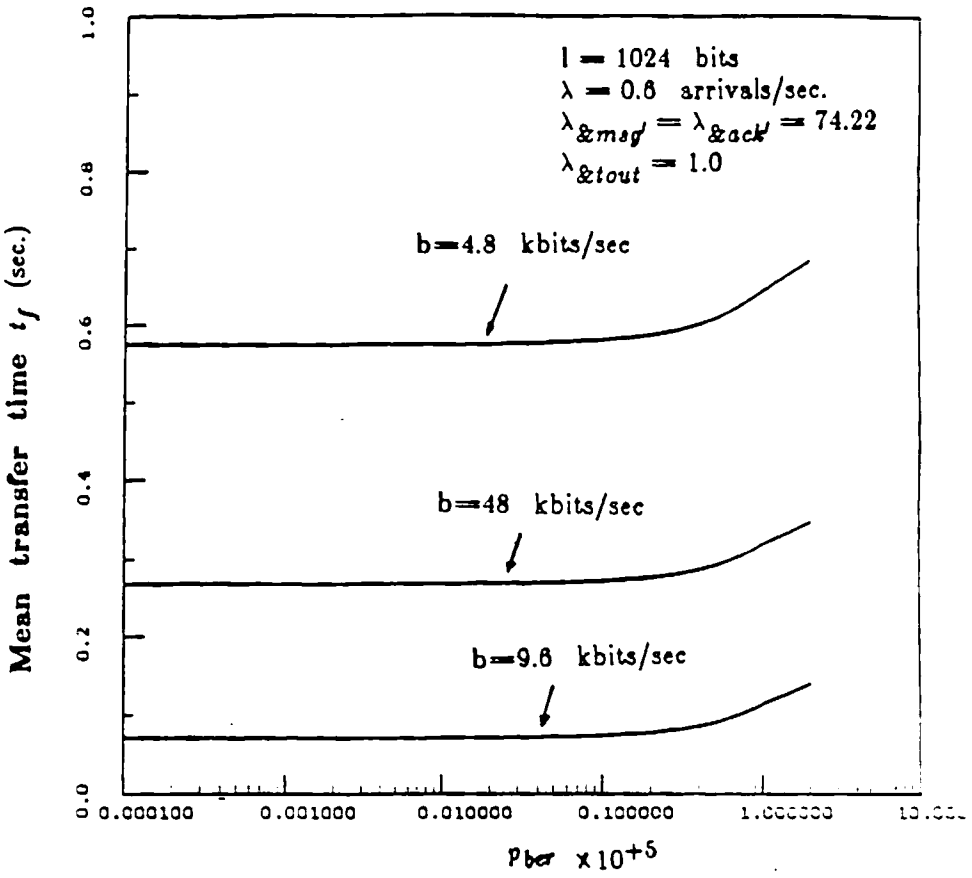

Figure 17: Mean transfer time t_f versus bit error probability p_{ber} for various channel bandwidth b

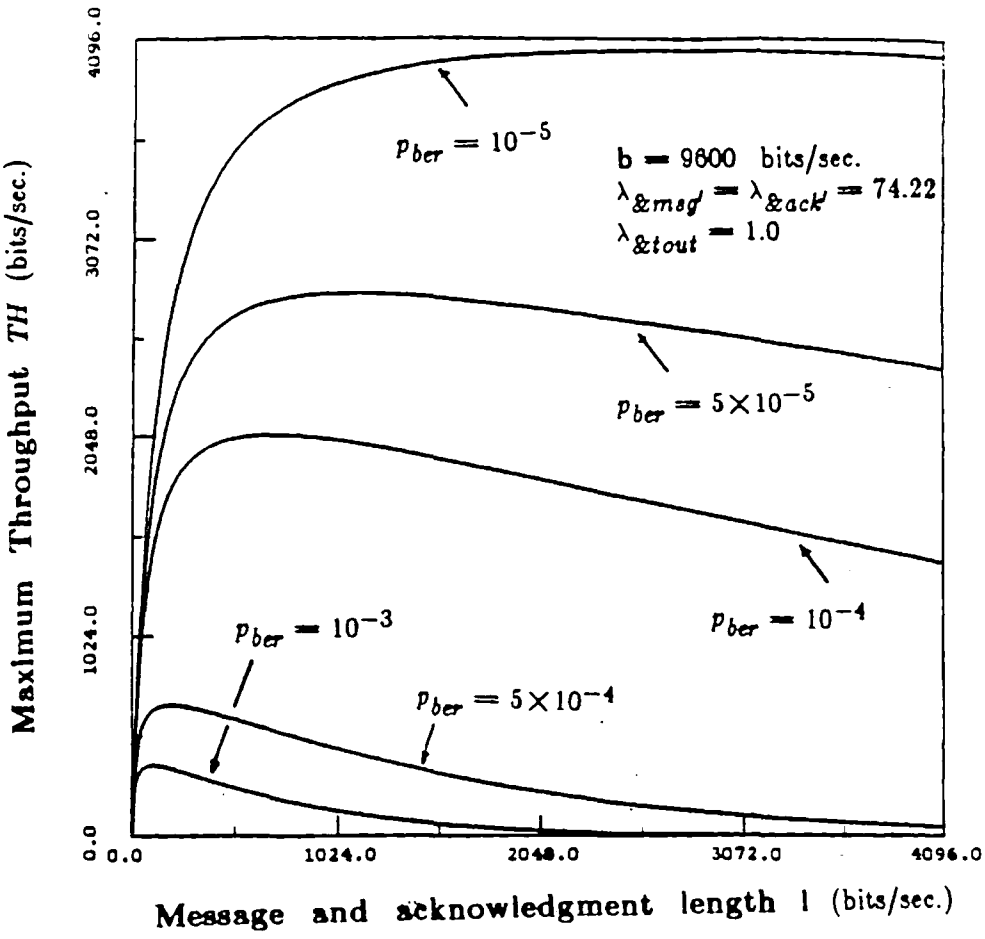

Figure 18: Maximum Throughput *TH* versus message and acknowledgment lengths l for various bit error probability p_{ber}

Table of Contents

List of Figures