Submitted for publication in *IEEE Transactions on Networking*

PERFORMABILITY OF A TOKEN BUS NETWORK UNDER TRANSIENT FAULT CONDITIONS*

W. H. Sanders†, J. F. Meyer**‡, and K. H. Muralidhar**

†The University of Arizona
Electrical and Computer Engineering Department
Tucson, AZ 85721

‡The University of Michigan
Electrical Engineering and Computer Science Department
Ann Arbor, MI 48109

**Industrial Technology Institute
P.O. Box 1485
Ann Arbor, MI 48105

ABSTRACT

Local area networks are representative of a growing class of distributed systems which demand more sophisticated methods and tools to evaluate system behavior in the presence of faults. In particular, since they are likely to exhibit degradable performance, the concept of performability can provide more refined assessments of their ability to perform under fault and error conditions. This paper presents the results of a detailed performability evaluation of a network employing the IEEE 802.4 protocol. In particular, a 30-station IEEE 802.4 token bus network operating in a hostile factory environment is evaluated using *stochastic activity networks*. Stochastic activity networks, a generalization of stochastic Petri nets, provide a convenient representation for computer networks and are formal enough to permit solution by both analysis and simulation. The evaluation results show 1) that stochastic activity networks are an appropriate model type for evaluating the performability of local area networks and 2) that the protocol is extremely tolerant to transient faults such as token losses and noise bursts under moderate network loads.

Keywords: Performability, Fault-Tolerant Computer Networks, Stochastic Petri Nets, Stochastic Activity Networks.

*Work by W. H. Sanders has been supported, in part, by the Digital Equipment Corporation Faculty Program: Incentives for Excellence.

I Introduction

Local area networks represent a fusion of concepts, techniques, and technologies from the fields of both computing and communication. The evaluation of such systems can likewise draw on methods and tools which have evolved in both fields and should attempt to bring them together in a fashion that is suited to the particular needs of network evaluation. The study described herein is based on such an approach and, we believe, illustrates its utility. Specifically, we integrate means of representing transient fault effects, as they've been treated in evaluations of computing system dependability, with modeling methods used to evaluate throughput-delay performance of communication networks. The system considered is a 30-station, token bus LAN subjected to a hostile transient fault environment of the type anticipated for industrial applications. It is shown that one can faithfully represent the details of a token-passing protocol in a network of this size, and obtain meaningful results concerning the network's performability in the presence of noise and token losses.

Past work concerning the evaluation of LANs has been primarily concerned with network performance, with less emphasis placed on issues of network dependability. Our meanings of "performance" and "dependability" here conform with terminology typically employed in the context of computing systems, i.e., with respect to some designated user-oriented or system-oriented service, *performance* refers to "quality of service, provided the system is correct " (see [1], for example). *Dependability*, on the other hand (see [2], for example), is that property of a system which allows "reliance to be justifiably placed on the service it delivers." Such service is *proper* if it is delivered as specified; otherwise it is *improper*. System *failure* is identified with a transition from proper service to improper service. Dependability is thus a collective term for more specific measures of reliability and availability. Moreover, such measures are distinct from performance measures (in the sense defined above), since the latter presume a correct (failure-free) system.

It should be noted, however, that such distinctions are not common among all system disciplines, e.g., in terminology recommended by the CCITT for telecommunication services (see [4]), performance is a more general concept that includes "item related performances" such as reliability and availability. There are also exceptions within the computing literature, e.g., a recent text by Molloy [5] views reliability and availability measures as instances of performance measures. Whatever terms are used, the real issue is whether separate eval-

uations of system performance (in the narrower sense) and system dependability suffice to determine the overall quality of the service(s) provided by that system. For an affirmative answer to this question, one must assume a binary view of the system's operational state (either up or down) and equate proper service (see above) with that delivered when the system is up. In this case, performance measures quantify the quality of proper service and dependability measures quantify the system's ability to deliver service of that quality. Accordingly, results of each type of evaluation, when taken together, can provide a rather complete assessment of service quality.

Typically, however, computer networks are *degradable* in the sense that variations (due to faults) in system structure, internal state, and environment can alter the quality of delivered service, even though that service, according to failure criteria, remains proper. For such systems, a binary correct-incorrect (up-down) classification of operational status is too coarse. Instead, a degradable system's operational state should be viewed as a multivalued variable representing the extent to which the system is faulty, i.e., which resources are faulty and, for each resource, its current error state, e.g., it has failed, is it in the process of fault recovery, it contains latent faults, etc. With such variations, the narrow view of performance is too restrictive and, although dependability measures are compatible with this view (indeed, the special nature of degradable computing systems was first investigated in a reliability context [8]), they account for service degradation only at the borderline between proper and improper service. These limitations were recognized in the mid-1970s, motivating the definition and application of unified performance-dependability concepts such as *performability* [6].

In general terms, a performability measure quantifies a system's "ability to perform" in the presence of faults where, formally, such ability is expressed by probabilities. A specific performability measure is obtained by defining just what "performance" means for the evaluation in question. When applied to computer networks, the choices here are virtually limitless, ranging from the type of binary-valued performance (success or failure) considered in reliability evaluation to continuous-valued performances such as packet transmission rates and station response times. More generally, performance, as interpreted in a performability context, can also mean "quality of service", i.e., for a specified network service, its values can reflect different degrees of satisfaction that might be experienced by the user of that service. For further discussion of this point in a related context (telecommunication networks), see

[7].

Others have likewise recognized the need to take a unified performance-dependability view when evaluating computer networks. One source of motivation for this approach has been the observation that failures in a network and congestion due to high traffic can have equivalent effects on the quality of service delivered. Specifically, Bonaventura *et al.* [9] evaluate the capability of a network to offer a specific grade of service (denoted its "service availability"), given the effect of both failures and congestion. Another example of work accounting for both failures and congestion is that of Lazaroiu and Staicut [10] who consider the concept of the "equivalent availability" of a computer network. Later work with a similar goal, but also aimed toward reducing the size of the model's state space, is that of Li and Silvester [11]. In [11], they provide an estimate of network performance in the presence of faults by considering only the most likely states in a stochastic process representing the system.

While the concept of performability provides the needed generality to effectively evaluate computer networks, it in itself does not suggest methods by which a model may be constructed. Although simple models can be constructed at the state level, realistic systems require more sophisticated representations to account for performance and dependability in a unified manner. Conventional queueing networks, although useful in the context of strict performance evaluation, do not adapt naturally to representations of behavior in the presence of internal and external faults. Stochastic extensions of Petri nets [12, 13], on the other hand, permit the representation of both performance and dependability related characteristics by varying the interpretations given to tokens. This is evidenced by a growing body of literature concerning the development of such models and their application in model-based performance and/or dependability evaluation. Included here is the work of Behr *et al.* ([14]; Evaluation Nets), Chiola ([15]; Generalized Stochastic Petri Nets), Cumani ([16]; Stochastic Petri nets with phase type distributions), Dugan *et al.* ([17]; Extended Stochastic Petri Nets), Godbersen and Meyer ([18]; Function Nets), Marsan *et al.* ([19]; Generalized Stochastic Petri Nets), Molloy ([13, 20]; Timed Petri Nets), Natkin ([12]; Timed Petri Nets), and Törn ([21]; Simulation Nets).

As discussed in the section that follows, another class of stochastic models of this type are "stochastic activity networks". These were developed for the specific purpose of performability evaluation (including more strict evaluations of performance and dependability) and,

as demonstrated in Section 3, are able to cope with the type of complexities encountered when evaluating the performability of local area networks.

II Stochastic Activity Networks

Stochastic activity networks (SANs) [22, 23] incorporate features of both stochastic Petri nets and queueing models. Structurally, SANs have primitives consisting of *activities, places, input gates, and output gates.* Activities ("transitions" in Petri net terminology) are of two types, *timed* and *instantaneous.* Timed activities represent activities of the modeled system whose durations impact the system's ability to perform. Instantaneous activities, on the other hand, represent system activities which, relative to the performance variable in question, complete in a negligible amount of time. Cases associated with activities permit the realization of two types of spatial uncertainty. Uncertainty about which activities are enabled in a certain state is realized by cases associated with intervening instantaneous activities. Uncertainty about the next state assumed upon completion of a timed activity is realized by cases associated with that activity. Places are as in Petri nets. Gates were introduced to permit greater flexibility in defining enabling and completion rules.

The stochastic nature of the nets is realized by associating an *activity time distribution function* with each timed activity and a *probability distribution* with each set of cases. Generally, both distributions can depend on the global marking of the network. A *reactivation function* [23] is also associated with each timed activity. This function specifies, for each marking, a set of *reactivation markings.* Informally, given that an activity is activated in a specific marking, the activity is *reactivated* whenever any marking in the set of reactivation markings is reached. This provides a mechanism for restarting activities that have been activated, either with the same or a different distribution. This decision is made on a per activity basis (based on the reactivation function), and is not a net-wide execution policy.

The execution of stochastic activity networks is discussed in detail in several places, including [24]. Informally, though, SANs execute in time through completions of activities that result is changes in markings. More specifically, an activity is chosen to *complete* in the current marking based on the relative priority among activities (instantaneous activities have priority over timed activities) and the activity time distributions of *enabled* activities. A case of an activity chosen to complete is then selected based on the probability distribution

for that set of cases. These two choices determine uniquely the next marking of the network, which is then obtained by executing the input gates connected to the input of the activity chosen and the output gates connected to the chosen case. This procedure is repeated by considering the activities enabled in the new marking.

Stochastic activity networks can be solved by both analysis and simulation, depending on system characteristics. Informally, SANs can be solved via analytic methods when all activity time distributions are exponential and activities are reactivated often enough to ensure that their rates depend only on the current state. When this is the case, "base model" stochastic processes exist that can be used to obtain analytic solutions for a wide class of variables characterizing both activity and marking related behavior. If this is not the case, simulation can be used to evaluate system behavior.

In order to be effectively applied to realistic systems, model construction and solution techniques require machine implementation. Both the complexity of the construction procedures and the typical sizes of resulting base models make this a necessity. To fill this need an extensive software package, called METASAN[1] [25], has been developed specifically for the construction and solution of SAN-based performability models.

METASAN was written using UNIX tools (C, Yacc, Lex, and Csh) and contains some 37,000 lines of source code. Models consist of two parts: a description of the structure of the net, and a description of the desired performance variables and solution method to be used in the evaluation process. Solution options include analytic techniques as well as both terminating and steady-state simulation.

III Token Bus Network

The ability of stochastic activity networks to represent both performance and dependability oriented characteristics and their inherent solvability makes them a natural choice for evaluating communication networks. We now make use of them to evaluate the performability of a 30-station IEEE 802.4 token bus network under transient fault conditions. Work to date concerning the evaluation of IEEE 802.4 networks has been widespread (see, for example [26, 27, 28]) but limited by the assumption of fault-free operation. While this assumption may suffice for certain systems and environments, it is not realistic for net-

[1] METASAN is a registered Trademark of the Industrial Technology Institute.

works operating in typically hostile manufacturing environments. Studies that account for performance in the presence of faults are important in this case because, in the event of failures, a LAN does not simply stop operating, but may continue to operate at a degraded level of service.

Several assumptions are made in transforming the token bus network considered into its model representation. In particular, our evaluation presumes a static network structure of 30 nodes. This static structure dictates that 1) no station is allowed to leave the logical ring formed by token bus protocol, 2) no new stations are allowed to enter the ring, and 3) ring collapse is not allowed when the token is lost. Two types of transient faults are considered: 1) faults that cause a token loss and 2) faults that result in noise on the bus (the latter can also cause a token loss if such noise corrupts a token frame). Token loss is represented via a specified probability (a parameter of the model) of losing the token each time it's passed. Bus noise is represented by a stream of noise bursts whose interarrival times have a normal distribution; the duration of each burst is fixed at approximately 4 slot times.

In addition, several assumptions are made concerning the individual stations. Specifically, it is assumed that:

1. Each station is up and operational (i.e., in the UP state), and will not be turned off during the course of simulation. Thus, it is not necessary to consider the OFF_LINE state.

2. All frames sent are of the type request_with_no_response (i.e., no acknowledgments are expected at this layer). This allows the states AWAIT_IFM_RESPONSE and CHECK_ACCESS_ CLASS to be combined. This is in accordance with the MAP 2.1 network architecture (i.e. IEEE 802.2 with class I option).

3. Each protocol machine implements only the highest priority class (class 6).

4. Each station always knows its successor and its predecessor in the logical ring.

Several additional assumptions are made concerning the network. In particular

1. Frame generation is modeled as a Poisson process. Frame size is fixed at 1024 bytes.

2. The data rate = 10 M bits/sec, Octet_interval = 0.8 μs, and Slot_time_interval = 5.6 μs.

Slot_Time Based Timers		
Name	*Slot Time Units*	*Value in μs*
bus_idle_timer	40	224
response_window_timer	2	11.2
token_pass_timer	4	22.4

Octet Interval Timers		
Name	*Octet Time Units*	*Value in μs*
token_rotation_timer	187500	150000
token_hold_timer	5000	4000

Table 1: Timer Values for Network Model

Table 1 shows numerical values for the various timers, delays, and other network parameters.

A Token Bus Network Model

The SAN model used in this study is based on the reduced state transition diagram given in Figure 1. This specification was obtained from the standard access control machine (ACM) state transition specification [29], taking into account the assumptions made in the previous section. In particular, note that DEMAND_IN and DEMAND_DELAY need not be represented since the structure of the logical ring is static. Additionally, it is not necessary to consider the OFF_LINE state since it is assumed that no stations leave the ring voluntarily during the operation period. Finally, states AWAIT_IFM_RESPONSE and CHECK_ACCESS_CLASS can be combined since all data frames sent are of the type "request_with_no_response".

Preconditions for a change in protocol state and postconditions after the state change are modeled by input gates and output gates respectively. Activity names are associated with protocol transition names (i.e., *receive_token*, *do_pass_token*). The token bus itself is represented by the place *pbus* in the model. The marking of this place indicates the activity taking place on the bus. For convenience, each possible marking of *pbus* is referred to by a textual label to aid in understanding node behavior. The markings for data frames,

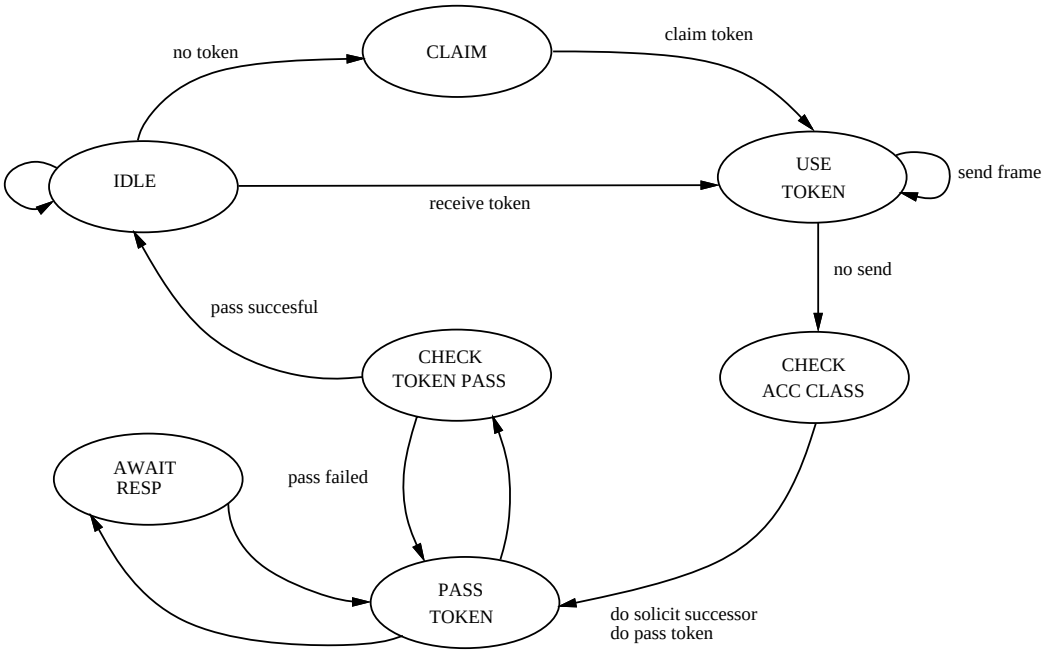


Figure 1: Reduced State Transition Diagram

data frames corrupted by noise, claim frame, solicit successor frame, corrupted protocol frame, and token frame are thus *data_frame*, *corrupt_data*, *claim_frame*, *sol_succ_frame*, *corrupt_protocol*, and *token_frame* respectively. The marking representing an idle bus is *idle* and the markings representing token frames are the non-zero integer less than or equal to 30.

Given this bus representation, the model for the entire network is constructed by developing a representation for a single station and replicating it 30 times, holding the place *pbus* common among all the replicate station models. Communication between stations is thus only through the bus, as it is in a real network. A stochastic activity network model for a single station is shown in Figure 2. In order to make the diagram more readable, several places (e.g., *pbus*) are pictured in several locations in the diagram, and are filled with a unique pattern to indicate that they actually represent a single place in the SAN. Activity, input gate, and output gate parameters are given in Tables 3, 4, and 5 respectively, which are found in the Appendix.

Token losses in the network are modeled by assuming that there is a certain probability that the token will be lost each time it is passed. This probability is represented by the

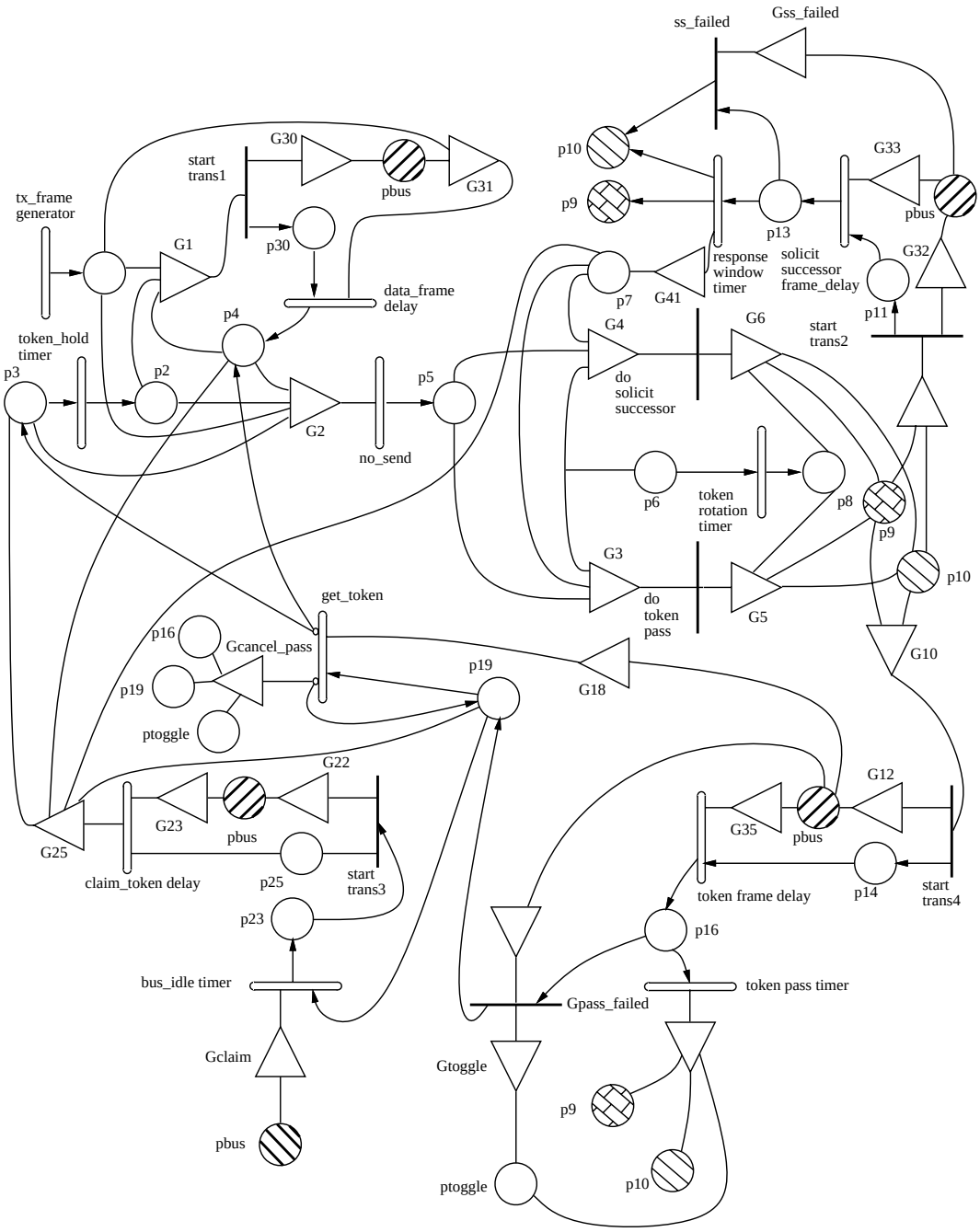


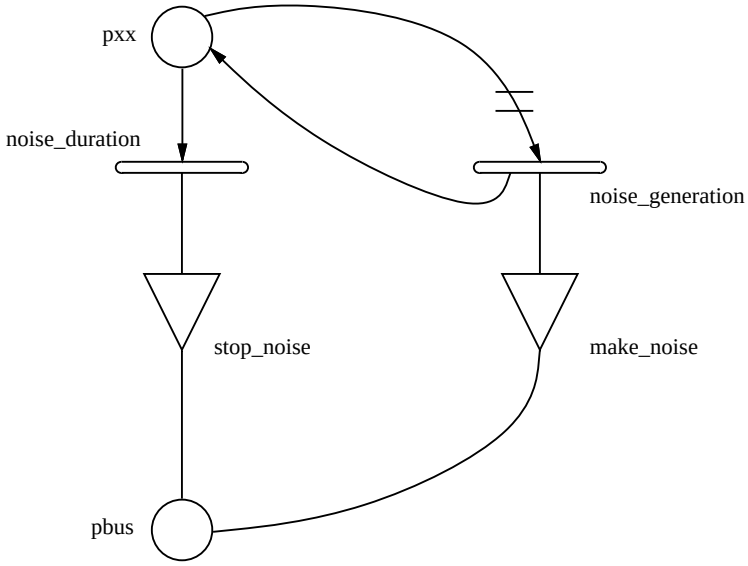
Figure 2: Station Model

cases of the activity *get_token* in Figure 2. A stochastic activity network model representing the generation of noise bursts is shown in Figure 3. Noise bursts are assumed to arrive in a normally distributed fashion. The effects of noise bursts on the token bus network are two-fold. If a noise burst affects a data frame, that data frame is retransmitted and the effects are recorded by the corrupted data on the bus (marking of place *pbus* is *corrupt_data*). If a noise burst affects any protocol frame, actions specified in the IEEE specification are taken and the effects are recorded by the corrupted protocol on the bus (marking of place *pbus* is *corrupt_protocol*).

B Model Execution

Changes in the ACM state during network operation are represented by activity completions in the model and places represent the complete state of each network node. For example, when a station receives the token from its previous station or generates a token in the case of a token loss, the markings of places *P3* and *P4* are set to one. A marking of one in place *P3* enables the activity *token_hold_timer* which has an activity time equal to the value of 5000 octets (see Table 1). The marking of place *P1* indicates the number of outstanding packets that are queued at that station. Gate *G1* holds if the markings of places *P1* and *P3* are non-zero and the marking of place *P2* is zero (the marking of place *P2* indicates that *token_hold_timer* has not expired). This allows for the transmission of a packet queued at the station. The start of transmission is signaled by the completion of the activity *start_trans1* which changes the marking of place *pbus* to *data_frame*. The time for transmitting a packet is given by the activity time of the timed activity *data_frame_delay*. Packets are transmitted until there are none to send or until the *token_hold_timer* expires (i.e. activity *token_hold_timer* completes). When transmission ceases, the predicate for gate *G2* is satisfied, enabling the instantaneous activity *no_send*.

At this point, one of two things can happen, depending of the **maximum intersolicit count**, as represented by place *P7*. If the marking of *P7* is not zero, the token is passed, and if it is zero, successors are solicited. The token is passed by enabling the activity *do_pass_token* and successors are solicited by enabling the activity *do_solicit_successor*. In the case of solicit successor, the predicate of gate *G9* is satisfied enabling activity *start_trans2*. Solicit successor frames on the bus are indicated by a marking of *sol_succ_frame* in the place *pbus*. Once the solicit successor is successful, the token is passed. Upon com-



Gate	Function
stop_noise	if (MARK($pbus$)==$corrupted\_carrier$) MARK($pbus$)=$idle$; else if (MARK($pbus$)==$corrupted\_CF$) MARK($pbus$)=$claim\_frame$;
make_noise	if (MARK($pbus$)==$data\_frame$) MARK($pbus$)=$corrupted\_data$; else if (MARK($pbus$)=$idle$) MARK($pbus$)=$corrupted\_carrier$; else if (MARK($pbus$)=$claim\_frame$) MARK($pbus$)=$corrupted\_CF$; else MARK($pbus$)=$corrupted\_protocol$;

Activity	Distribution
noise_generation	normal($time\_between\_noise$, $time\_between\_noise$ * .2)
noise_duration	determ(22.4)

Figure 3: Noise Burst Model

Activity Name	*Distribution*	*Parameters in μs*
tx_frame_generator	exponential	varied
data_frame_delay	deterministic	800.0
solicit_succ_frame_delay	deterministic	20.8
token_frame_delay	deterministic	20.8
get_token	deterministic	0.1
claim_token_delay	deterministic	dependent on station ID
noise_generation	normal	varied
noise_duration	deterministic	22.4

Table 2: Timed Activity Distributions for Network Model

pletion of either a solicit successor operation or activity *do_pass_token*, the predicate for gate *G10* is satisfied and a token frame is placed on the bus. This action enables the timed activity *token_frame_delay*. The activity time of *token_frame_delay* is equivalent to the token transmission time interval. If the token pass fails, the token pass is tried a second time and if it fails again, a solicit successor is done to find the next station to which the token is passed.

In the case of a token loss, signified by the completion of the timed activity *bus_idle_timer*, the instantaneous activity *start_trans3* is enabled to start a claim token phase. Claim token frames are indicated by a marking of *claim_frame* in the place *pbus*. Claim frames are sent for a period whose length is a function of station ID. At the completion of the activity *claim_frame_delay* a token is generated and the network resumes normal operation.

Model execution thus mimics actual protocol operation. A change in protocol state, an expiration of a timer, a frame arrival, etc., all cause a change in the marking of the network. By observing the execution of the stochastic activity network model one can infer the protocol behavior. Table 2 lists the timed activities associated with the model with their corresponding distributions and parameter values.

C Performance Variables

The performance variables studied include the mean response time, the mean token rotation time, and the fraction of time each type of frame (data or protocol) is on the bus, for a variety of workload and environmental conditions. Nine types of bus activities are considered, corresponding to the different data and protocol frames that can be on the bus

together with possible error conditions. Each of these variables is formulated as the fraction of time that the particular condition existed in steady-state. Specifically, the following types of bus activity are considered: data transmission, idle bus, token on the bus, solicit successor frame present, corrupted protocol frame present, corrupt carrier present, claim frame present, corrupted data present, and listen for solicit successor. The name used for each variable in the input code and reporting of the results is, respectively, *data_on_bus*, *idle_bus*, *token_on_bus*, *soll_succ_frame*, *corrupt_pro*, *corrupt_carrier*, *claim_frame*, *cdata_on_bus*, and *listen_soll_succ*.

Response time characteristics of the network under a variety of conditions are also evaluated. In this case, the mean number of packets at a node in steady-state (including the one in transmission, if any) is estimated and Little's result is used to obtain the expected combined queueing and transmission time. Response times for several different stations are studied to see whether position in the logical ring was a significant factor. For the remainder of this study, the station next in the ring after the *high station* is referred to as the *low station*, and the station equally distant from each of these stations, logically, is referred to as the *middle station*. Response times for the low and high stations are estimated for all conditions studied. Response times for the middle station are estimated only for high load conditions. In the results section, the variables *num_in_sysHS*, *num_in_sysLS*, and *num_in_sysMS* refer to the expected number of packets at each station (in steady-state) for the high, low, and middle stations respectively. Expected response times are then computed by dividing these estimates by the chosen arrival rate at the station. In addition, the expected token rotation time is estimated for each experiment. In the tables given in the Appendix, this variable is referred to as *token_rotation*.

D Discussion of Results

Three sets of simulation experiments were conducted to evaluate the performability of the 30 station token bus network. The resulting performance variable values are given in Tables 6 to 10, which are found in the Appendix. All intervals reported in the tables are at a 90% confidence level.

In the first set of experiments, network behavior was studied under varying arrival rates of data packets to the station (5% to 85% of bus capacity and a Poisson arrival process), a token loss probability of 0.001, and a noise burst rate with an expected time between bursts

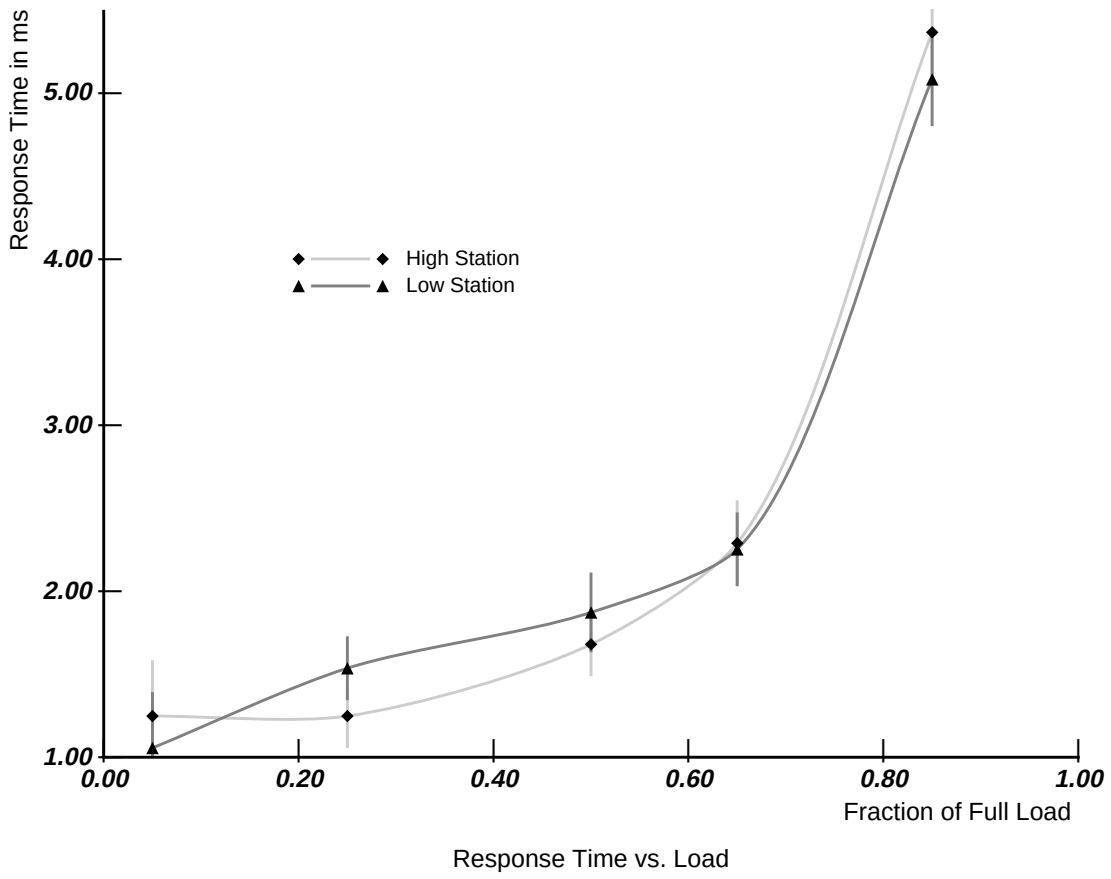


Figure 4: Response Time vs. Load (Token Loss Prob. .001, Mean Time Between Noise Bursts 400 ms)

of 400 milliseconds. The resulting values of the performance variables are given in Table 6.

The expected response time to requests at the high and low station was calculated using these results, and is shown in Figure 4. This time is the sum of the queueing delay and the frame transmission time (in this case transmission time is a constant 800 μs). The response time is good even at 85% load, where it has an average value of 5.25 milliseconds. Though Figure 4 indicates different values for response time at the low station and the high station, these differences are not statistically significant, since their confidence intervals overlap. It should also be noted that for this, and all other experiments, the high priority maximum token hold time for each station in the network was set at 4 milliseconds.

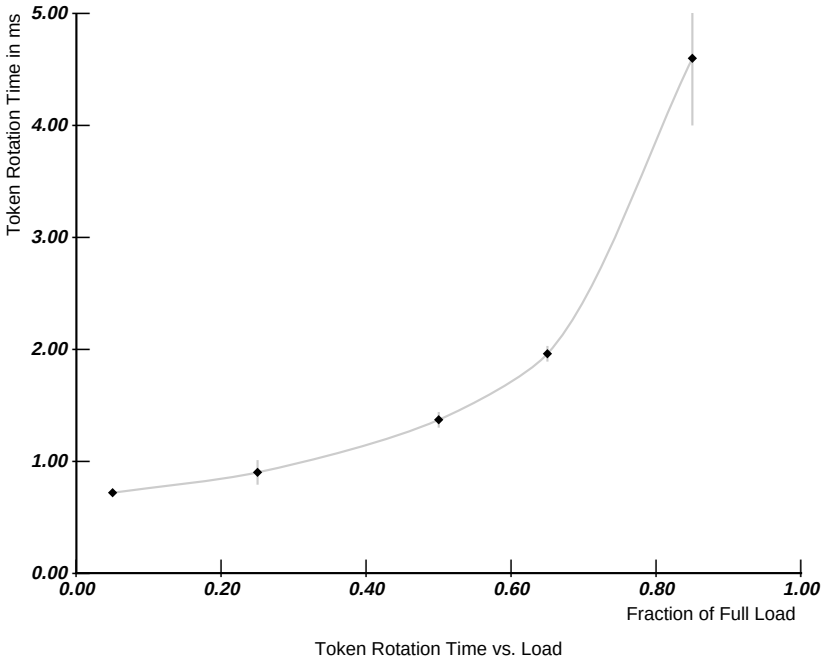


Figure 5: Token Rotation Time vs. Load (Token Loss Prob. .001, Mean Time Between Noise Bursts 400 ms)

The expected token rotation time for the network, as a function of network load, is shown in Figure 5. As expected, this time increases with increase in load. However, even at 85% load, the token rotation time has a reasonable average value of 4.5 milliseconds. This behavior is due to the large value of high priority token hold time used for the stations. It indicates that the network was not saturated at this combination of high priority token hold time, packet size, and load conditions.

The behavior of the other performance variables, as a function of network load, is shown in Figure 6. These results include the fraction of time data frames, token frames, solicit successor frames, claim frames, and corrupt data and protocol frames, are present on the bus. This behavior is shown to indicate that the model developed for the network closely follows the expected behavior of the token bus network. As depicted in Figure 6, when the load increases, the percentage of time that the data frames are on the bus increases and that of other frames decreases.

In the second set of experiments, the network behavior was studied at 50% and 85%

Figure 6: Bus Activity in Steady-State (Token Loss Prob. .001, Mean Time Between Noise Bursts 400 ms)

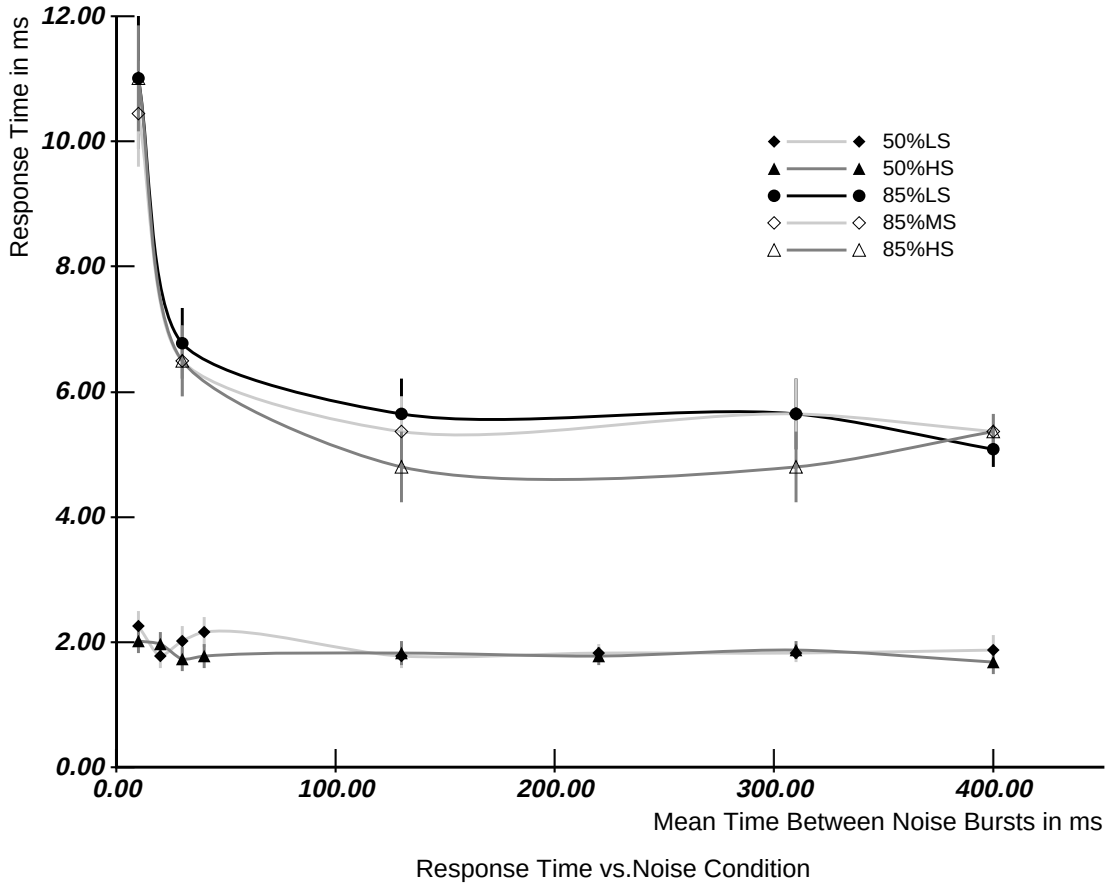


Figure 7: Response Time vs. Noise Condition (Data Transmission Time .8 ms)

load for varying noise burst rates (the expected time between noise bursts was 10 to 400 ms). The resulting values of the performance variables are given in Tables 7 and 8.

The expected response time to requests at the high and low station under varying noise burst rates was calculated from these results, and is shown in Figure 7. It is interesting to note that token bus network behavior, as reflected by changes in the expected response time, is relatively immune to noise bursts. This is further emphasized by the fact that the experiments were conducted under extremely high noise burst rates. Specifically, the average response time under the condition of 50% load and high noise burst rate is 2.2 ms and is statistically very close to that with low noise burst rate (2 ms).

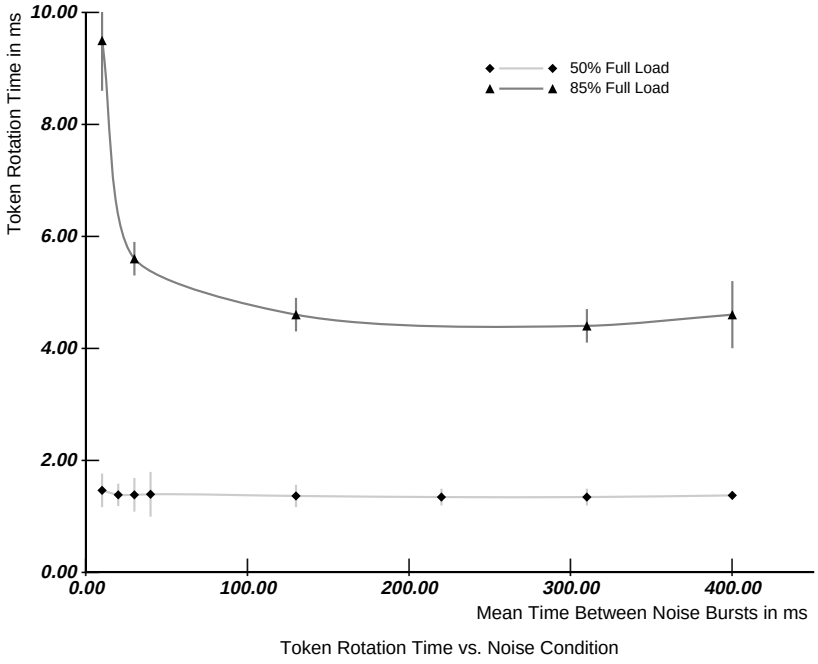


Figure 8: Token Rotation Time vs. Noise Condition (Data Transmission Time .8 ms)

The expected response time of the high, low, and middle stations of the network under 85% load for varying noise burst rate is also shown in Figure 7. This figure suggests an increase in expected response time at very high noise burst rates. This increase is attributed to the increase in the chance of a noise burst affecting a data frame, with increasing load. However, it should be noted that the network generally exhibits high immunity to noise bursts, which is extremely important in factory floor applications. The average value of the response at 85% load and very high noise burst rate is 11 ms.

The expected token rotation time for the network under 50% and 85% load and varying noise burst rate is shown in Figure 8. It should be noted that there is no appreciable difference in the expected token rotation time in the network under 50% load and low noise burst rate (expected time between noise bursts was 400 ms) and 50% load and high noise burst rate (expected time between noise bursts was 10 ms). However, the expected token rotation time doubles with 85% load and high noise burst rate compared to that at 85%

load and low noise burst rate.

In the third set of experiments, the network behavior was studied under 50% and 85% load and varying token loss probability (0.001 to 0.1). The token loss probability indicates the probability that a station will lose the token when passing it to the next station. (Note that a token loss probability of 0.1 implies that, on the average, one token is lost for every 10 token passes. This represents an extremely severe condition in the network.) As discussed when the model was developed, it is assumed that this token loss will not bring down the entire logical ring. The resulting values of performance variables for this set of experiments are given in Tables 9 and 10.

The expected response time at the high and low station calculated from these values is shown in Figure 9. Though the expected response times differ for the high and the low stations, they are not statistically different because of their confidence interval widths. The response time of the token bus network increases from 2 ms to 3 ms at 50% load and token loss probability varying from 0.001 to 0.1. The response time at the low, middle, and high station under an 85% load is also shown in Figure 9. Note that, at this loading, the response time increases with increase in token loss probability, and the increase depends on the position of the station in the logical ring. Most significantly, the expected response time for the middle station increases from 4.5 ms with 0.005 token loss probability to 27 ms with 0.1 token loss probability.

Finally, the expected token rotation time for the network as a function of the token loss probability is shown in Figure 10. As can be seen from the figure, at 50% load there is relatively little influence due to changes in token loss probability. Indeed, when there is a very high loss probability (0.1), the token rotation time still has a reasonable value (2.5 ms). However, at a higher load (85%), the expected token rotation time does increase with increasing token loss probability. In particular, the expected token rotation time increases from 4.5 ms to 8 ms when the token loss probability is varied from 0.005 to 0.1.

IV Conclusions

Hostile environment conditions persisting in factory floors result in transient faults, such as noise bursts and token losses, which must be accounted for when modeling the network. These faults have not been accounted for in previous studies. Stochastic activity networks

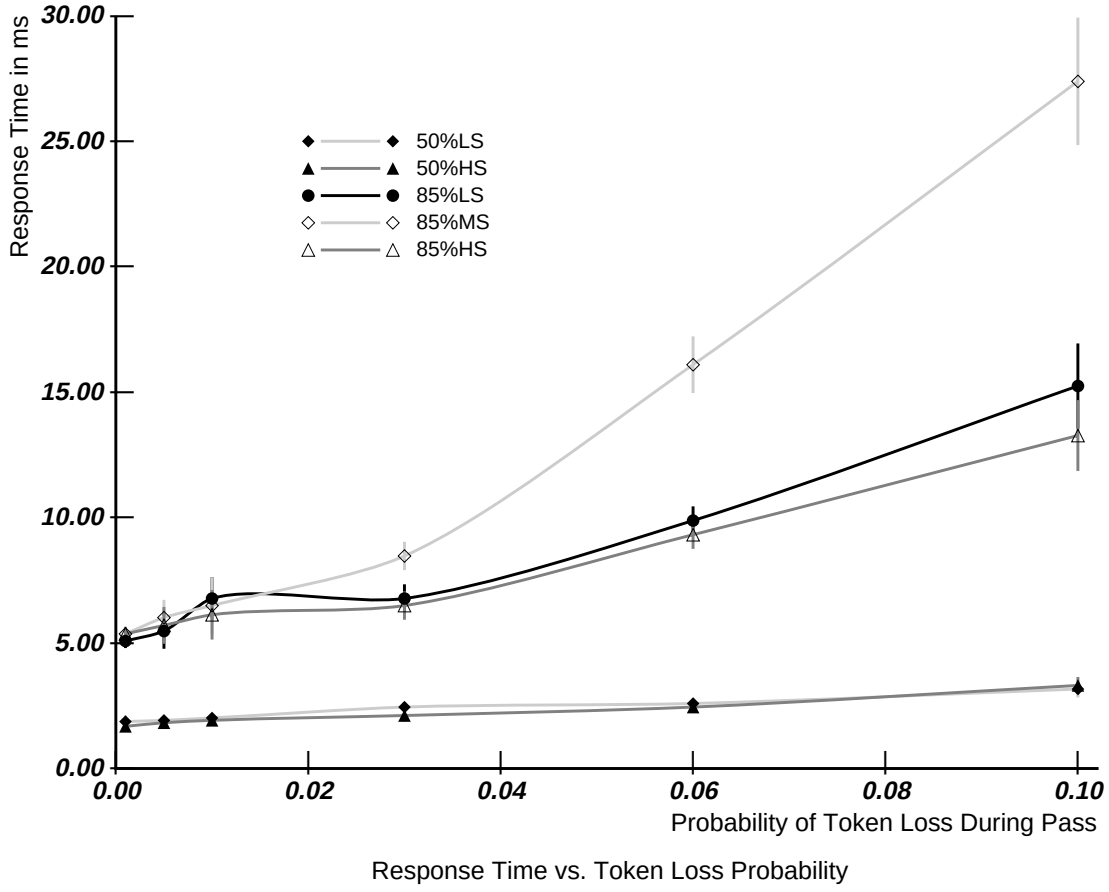


Figure 9: Response Time vs. Token Loss Probability (Data Transmission Time .8 ms)

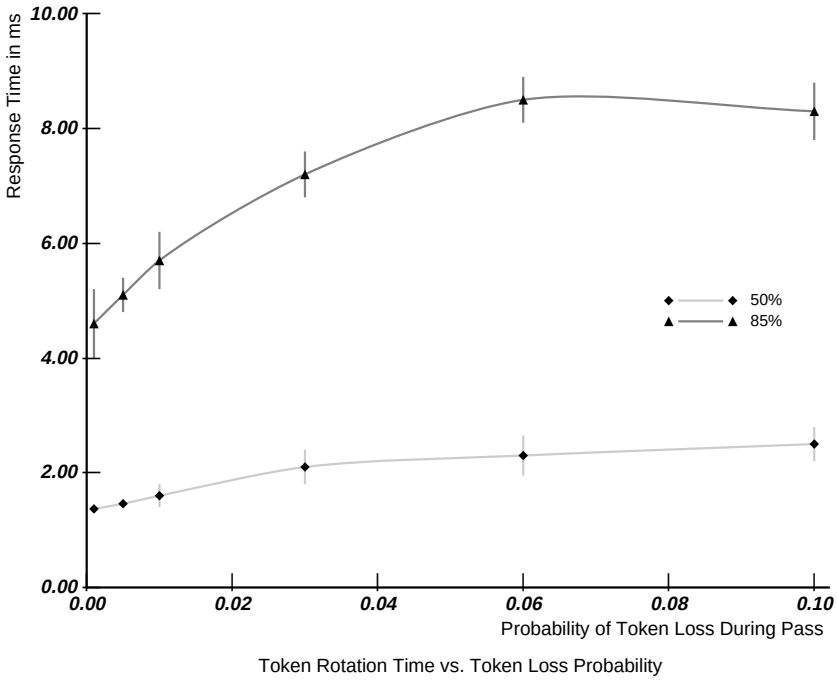


Figure 10: Token Rotation Time vs. Token Loss Probability (Data Transmission Time .8 ms)

and performability allow us to take these effects into account in an explicit and systematic manner.

The results obtained in this study provide important information regarding the behavior of IEEE 802.4 token bus networks. Noise bursts and random token losses have been shown to have minimal impact on network performance in the case of moderate (50%) network loading. In addition, it has been shown that even severe noise bursts and token losses have a manageable impact on token bus networks operating at very high (85%) loading.

REFERENCES

[1] D. Ferrari, *Computer Systems Performance Evaluation*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[2] J. Laprie (ed.), *Dependability: Basic Concepts and Terminology*, Springer-Verlag, Wien, 1992.

[3] A. Avizienis and J. C. Laprie, "Dependable computing:From concepts to design diversity," *Proc. of the IEEE*, vol. 74, no. 5, pp. 629–638, May 1986.

[4] *CCITT Red Book, Fasc. III.1: General Characteristics of International Telephone Connections and Circuits*, ITU, Geneva, 1985.

[5] M. K. Molloy, *Fundamentals of Performance Modeling*, MacMillan, New York, NY, 1989.

[6] J. F. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Trans. Comput.*, vol. C-22, pp. 720–731, Aug. 1980.

[7] J. F. Meyer, "Performability evaluation of telecommunication networks," in *Proc. 12th Int. Teletraffic Conf.*, Torino, Italy, June 1988.

[8] B. R. Borgerson and R. F. Fretitas, "A reliability model for gracefully degrading and standby-sparing systems," *IEEE Trans. Comput.*, vol. C-24, no. 5. pp. 517–525, May 1975.

[9] V. Bonaventura, S. Cacopardi, M. Decina, and A. Roveri, "Service availability of communication networks," in *Proc. 1980 Nat. Telecommun. Conf.*, 1980, pp. 15.2.1–15.2.6.

[10] D. F. Lazaroiu and E. Staicut, "Congestion-reliability-availability relationship in packet-switching networks," *IEEE Transactions on Reliability*, vol. R-32, no. 4, pp. 354–357, Oct. 1983.

[11] V. O. Li and J. A. Silvester, "Performance analysis of networks with unreliable components," *IEEE Trans. on Comm.*, vol. COM-32, no. 10, pp. 1105–1110, Oct. 1984.

[12] S. Natkin, "Reseaux de petri stochastiques," PhD thesis, CNAM-PARIS, 1980.

[13] M. K. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Trans. Comput.*, vol. C-31, pp. 913–917, Sep. 1982.

[14] J. P. Behr, N. Dahmen, J. Muller, and H. Rodenbeck, "Graphical modeling with FORCASD," in *Computer Applications in Production and Engineering*, North-Holland, Amsterdam, 1983, pp. 61–630.

[15] G. Chiola, "A software package of the analysis of generalized stochastic Petri net models," in *Proc. International Workshop on Timed Petri Nets*, Torino, Italy, July 1985, pp. 136–143.

[16] A. Cumani, "ESP - a package of the evaluation of stochastic Petri nets with phase-type distributed transition times," in *Proc. International Workshop on Timed Petri Nets*, Torino, Italy, July 1985, pp. 144–151.

[17] J. B. Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nicola, "Extended stochastic petri nets: applications and analysis," in *Performance 84*, North-Holland, Amsterdam, 1984, pp. 507–519.

[18] H. P. Godbersen and B. E. Meyer, "Function nets as a tool for the simulation of information systems," in *Proc. of the Summer Computer Simulation Conference*, Newport, CA, July 1978.

[19] M. A. Marsan, G. Balbo, and G. Conte, "A class of generalized stochastic Petri nets for performance evaluation of multiprocessor systems," *ACM Trans. on Computer Systems*, vol. 2, no. 2, pp. 93–122, May 1984.

[20] M. K. Molloy, "Discrete time stochastic Petri nets," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 4, pp. 417–423, Apr. 1985.

[21] A. A. Torn, "Simulation nets, a simulation modeling and validation tool," *Simulation*, vol. 45, no. 2, pp. 71–75, Aug. 1985.

[22] A. Movaghar and J. F. Meyer, "Performability modeling with stochastic activity networks," in *Proc. 1984 Real-Time Systems Symp.*, Austin, TX, Dec. 1984.

[23] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: structure, behavior, and application," in *Proc. International Workshop on Timed Petri Nets*, Torino, Italy, July 1985, pp. 106–115.

[24] W. H. Sanders, "Construction and solution of performability models based on stochastic activity networks," Computing Research Laboratory Technical Report CRL-TR-9-88, The University of Michigan, Ann Arbor, MI, August 1988.

[25] W. H. Sanders and J. F. Meyer, "METASAN: a performability evaluation tool based on stochastic activity networks," in *Proc. ACM-IEEE Comp. Soc. 1986 Fall Joint Comp. Conf.*, Dallas, TX, Nov. 1986.

[26] V. Chauhan and A. S. Sethi, "Performance studies of token based local area networks," in *Proc. 10th Conference on Local Computer Networks*, pp. 100–107, Minneapolis, MN, October 1985.

[27] J. Y. Chien, "Performance analysis of the 802.4 token bus media access control protocol," in *Proc. Factory Floor Communications Workshop,* pp. 1–39, General Motors Technical Center, Warren, MI, September 1981.

[28] D. Janetzky and K. S. Watson, Performance evaluation of the MAP token bus in real time applications, in *Advances in Local Area Networks,* pp. 411–425, New York: IEEE Press, 1987.

[29] *Token-Passing Bus Access Method, Std 802.4-1985*, Institute of Electrical and Electronics Engineers, 1985.

Activity	**Rate**	**Probability**	
		case 1	case 2
tx_frame_gen	exp(λ)	1	-
data_frame_delay	determ(800)	1	-
start_trans1	inst	1	-
token_hold_timer	determ(4000)	1	-
do_solicit_successor	inst	1	-
do_pass_token	inst	1	-
token_rotation_timer	determ(150000)	1	-
no_send	determ(0.1)	1	-
response_window_timer	if (*station_id*==*low_station_id*) determ(11.2) else determ(5.6)	1	-
ss_failed	inst	1	-
solicit_succ_frame_delay	determ(20.8)	1	-
token_frame_delay	determ(20.8)	1	-
start_trans2	inst	1	-
pass_successful	inst	1	-
token_pass_timer	determ(22.4)	1	-
get_token	determ(.1)	1 - *prob_loss*	*prob_loss*
bus_idle_timer	determ(224)	1	-
start_trans3	inst	1	-
claim_token_delay	determ(750.0*(*station_id*/*high_station_id*))	1	-
start_trans4	inst	1	-

Table 3: Activity Parameters for Network Node

Gate	Enabling Predicate	Function
$G1$	MARK($P1$)≥1 and MARK($P2$)==0 and MARK($P4$)≥1	MARK($P4$)=MARK($P4$)-1;
$G2$	(MARK($P2$)≥1 or MARK($P1$)==0) and MARK($P4$)≥1	if (MARK($P2$)≥1) MARK($P2$)=MARK($P2$)-1; MARK($P3$)=0; MARK($P4$)=MARK($P4$)-1;
$G3$	(MARK($P6$)==1 or MARK($P7$)>0) and MARK($P5$)≥1	MARK($P5$)=MARK($P5$)-1; if (MARK($P7$)≥1) MARK($P7$)=MARK($P7$)-1; if (MARK($P6$)≥1) MARK($P6$)=0;
$G4$	MARK($P5$)≥1 and MARK($P7$)==0 and MARK($P6$)==0	MARK($P5$)=MARK($P5$)-1; MARK($P6$)=0;
$G9$	MARK($P10$)==1 and MARK($P9$)==0	MARK($P10$)=0;
$G10$	MARK($P10$)==1 and MARK($P9$)==1	MARK($P10$)=0; MARK(P9)=0;
$G18$	$station\_id$==MARK($pbus$)	MARK($pbus$)=$idle$;
$G23$	1	if ($station\_id$==$high\_station\_id$) if (MARK(pxx)≠1) MARK($pbus$)=$idle$; else MARK($pbus$)=$corrupt\_carrier$;
$G31$	1	if (MARK($pbus$)==$corrupt\_data$) if (MARK(pxx)≠1) MARK($pbus$)=$idle$; else MARK($pbus$)=$corrupt\_carrier$; else MARK($pbus$)=$idle$; MARK($P1$)=MARK($P1$)-1;
$G33$	1	if (MARK(pxx)≠1) MARK($pbus$)=$idle$; else MARK($pbus$)=$corrupt\_carrier$;
$G35$	1	if (MARK($pbus$)≠$corrupt\_protocol$) MARK($pbus$)=$next\_station\_id$; else if (MARK(pxx)≠1) MARK($pbus$)=$idle$; else MARK($pbus$)=$corrupt\_carrier$;
$G37$	MARK($pbus$)≠$next\_station\_id$ and MARK($pbus$)≠$idle$ and MARK($pbus$)≠$corrupt\_protocol$ and MARK($pbus$)≠$corrupt\_carrier$	identity
Gss_failed	MARK($pbus$)==$corrupt\_carrier$ or MARK($pbus$)==$corrupt\_protocol$	identity
$Gclaim$	MARK($pbus$)==$idle$)	identity

Table 4: Input Gate Parameters for Network Node

Gate	Function
$G5$	MARK($P8$)=1; MARK($P9$)=1; MARK($P10$)=1;
$G6$	MARK($P9$)=0; MARK($P10$)=1; MARK($P8$)=1;
$G12$	if (MARK(pxx)$\neq$1) MARK($pbus$)=$token_frame$; else MARK($pbus$)=$corrupt_protocol$;

$G22$	if ($station\_id$==$high\_station\_id$)
	if (MARK(pxx)$\neq$1)
	MARK($pbus$)=$claim\_frame$;
	else MARK($pbus$)=$corrupt\_CF$;

$G25$	if ($station\_id$==$high\_station\_id$)
	MARK($P4$)=1; MARK($P3$)=1;
	MARK($P7$)=20;
	else
	MARK($P19$)=1; MARK($P7$)=20;

$G30$	if (MARK(pxx)$\neq$1) MARK($pbus$)=$data_frame$; else MARK($pbus$)=$corrupt_data$;
$G32$	if (MARK(pxx)$\neq$1) MARK($pbus$)=sol_succ_frame; else MARK($pbus$)=$corrupt_protocol$;
$G41$	MARK($P7$)=20;
$Gcancel\_pass$	MARK($P16\_previous\_station$)=0; MARK($P19\_previous\_station$)=1; MARK($Ptoggle\_previous\_station$)=0;

$Gpass\_failed$	if (MARK($Ptoggle$)==0)
	MARK($Ptoggle$)=1; MARK($P9$)=1;
	MARK($P10$)=1;
	else
	MARK($Ptoggle$)=0; MARK($P9$)=0;
	MARK($P10$)=1;

$Gtoggle$	MARK($Ptoggle$)=1;

Table 5: Output Gate Parameters for Network Node

Time Between Noise Bursts 400 ms, Token Loss Prob. .001				
Fraction of Full Load	data_on_bus (frac. of time)	idle_bus (frac. of time)	token_on_bus (frac. of time)	soll_succ_frame (frac. of time)
.05	.051 ± .002	.0228 ± .0004	.857 ± .003	.0331 ± .0006
.25	.250 ± .007	.0177 ± .0004	.679 ± .007	.0264 ± .0007
.50	.494 ± .009	.012 ± .0004	.457 ± .008	.018 ± .0007
.65	.646 ± .009	.0085 ± .0031	.319 ± .008	.0122 ± .0005
.85	.847 ± .006	.0037 ± .0002	.137 ± .005	.0052 ± .005

Time Between Noise Bursts 400 ms, Token Loss Prob. .001				
Fraction of Full Load	corrupt_pro (frac. of time)	corrupt_carrier (frac. of time)	claim_frame (frac. of time)	cdata_on_bus (frac. of time)
.05	.000024 ± .000006	.00003 ± .000006	.031 ± .002	.00009 ± .00008
.25	.000023 ± .000008	.000020 ± .000007	.023 ± .001	.0002 ± .0001
.50	.000016 ± .000009	.000014 ± .000007	.016 ± .002	.0006 ± .0002
.65	.000009 ± .000005	.000009 ± .000004	.012 ± .001	.00007 ± .00002
.85	.000009 ± .000003	.000007 ± .000003	.0052 ± .0005	.0009 ± .0002

Time Between Noise Bursts 400 ms, Token Loss Prob. .001			
Fraction of Full Load	num_in_sys HS	num_in_sys LS	token_rotation (ms)
.05	.0026 ± .0007	.0022 ± .0007	.718 ± .002
.25	.013 ± .002	.016 ± .002	.90 ± .11
.50	.035 ± .004	.039 ± .005	1.37 ± .07
.65	.062 ± .007	.061 ± .006	1.96 ± .07
.85	.19 ± .01	.18 ± .01	4.6 ± .6

Table 6: Experiments with Load Varied

50% Full Load, Token Loss Prob. .001			
Time Between Noise Bursts (ms)	num_in_sys HS	num_in_sys LS	token_rotation (ms)
10	.042 ± .004	.047 ± .005	1.46 ± .3
20	.041 ± .004	.037 ± .004	1.38 ± .2
30	.036 ± .004	.042 ± .005	1.38 ± .3
40	.037 ± .004	.045 ± .005	1.39 ± .4
130	.038 ± .004	.037 ± .004	1.36 ± .2
220	.037 ± .003	.038 ± .003	1.34 ± .15
310	.039 ± .003	.038 ± .003	1.34 ± .15
400	.035 ± .004	.039 ± .005	1.37 ± .07

Table 7: Experiments with Noise Burst Rate Varied, 50% Load

85% Full Load, Token Loss Prob. .001				
Time Between Noise Bursts (ms)	num_in_sys HS	num_in_sys LS	num_in_sys MS	token_rotation (ms)
10	.39 ± .03	.39 ± .04	.37 ± .03	9.5 ± .9
30	.23 ± .02	.24 ± .02	.23 ± .02	5.6 ± .3
130	.17 ± .02	.20 ± .02	.19 ± .02	4.6 ± .3
310	.17 ± .02	.20 ± .02	.20 ± .02	4.4 ± .3
400	.19 ± .01	.18 ± .01	.19 ± .01	4.6 ± .6

Table 8: Experiments with Noise Burst Rate Varied, 85% Load

50% Full Load, Mean Time Between Noise Bursts 400 ms			
Prob. of Token Loss per Pass	num_in_sys HS	num_in_sys LS	token_rotation (ms)
.001	.035 ± .004	.039 ± .005	1.37 ± .07
.005	.038 ± .003	.040 ± .003	1.46 ± .01
.01	.040 ± .003	.042 ± .003	1.6 ± .2
.03	.044 ± .005	.051 ± .004	2.1 ± .3
.06	.051 ± .004	.054 ± .004	2.3 ± .35
.1	.069 ± .007	.066 ± .007	2.5 ± .3

Table 9: Experiments with Token Loss Probability Varied, 50% Load

85% Full Load, Mean Time Between Noise Bursts 400 ms				
Prob. of Token Loss per Pass	num_in_sys HS	num_in_sys LS	num_in_sys MS	token_rotation (ms)
.001	.19 ± .01	.18 ± .01	.19 ± .01	4.6 ± .6
.005	.202 ± .026	.194 ± .025	.213 ± .025	5.1 ± .3
.01	.217 ± .035	.24 ± .03	.23 ± .04	5.7 ± .5
.03	.23 ± .02	.24 ± .02	.30 ± .02	7.2 ± .4
.06	.33 ± .02	.35 ± .02	.57 ± .04	8.5 ± .4
.1	.47 ± .05	.54 ± .06	.97 ± .09	8.3 ± .5

Table 10: Experiments with Token Loss Probability Varied, 85% Load