

K. Prodromides and W. H. Sanders, "Performability Evaluation of CSMA/CD and CSMA/DCR Protocols under Transient Fault Conditions," to appear in IEEE Trans. on Reliability, March 1993.

PERFORMABILITY EVALUATION OF CSMA/CD AND CSMA/DCR PROTOCOLS UNDER TRANSIENT FAULT CONDITIONS*

K. H. Prodromides and W. H. Sanders

Department of Electrical and Computer Engineering

The University of Arizona

Tucson, AZ 85721 USA

(602) 621 - 6181

prodrom@ece.arizona.edu

whs@ece.arizona.edu

Keywords — CSMA/CD Protocol, Deterministic Collision Resolution (DCR), Performability, Stochastic Activity Networks (SANs), Stochastic Petri Nets.

Reader Aids —

Purpose: Describe a method for determining the performability of computer networks under transient fault conditions.

Special math needed for explanation: None

Special math needed to use results: None

Results useful to: Visualize and understand the effects of transient faults on two carrier sense multiple access protocols.

*This work was supported in part by the Digital Equipment Corporation Faculty Program: Incentives for Excellence.

Summary and Conclusions — CSMA/CD networks have gained widespread acceptance for use in many application areas. However, a barrier to their acceptance for industrial use has been the non-deterministic nature of the collision-resolution scheme employed in the standard variants (Ethernet and IEEE 802.3) of the protocol. Alternative schemes, which have a deterministically bounded collision-resolution time, have been proposed. These deterministic schemes typically work by partitioning the network nodes into groups, which may transmit during particular time periods after a collision is detected. The limit, in which each node is in its own group, has been implemented, but has not been evaluated in the presence of transient faults to determine if it offers any significant advantage over the standard exponential binary backoff algorithm. This is important in an industrial environment, where noise may significantly affect protocol operation. This paper presents the results of such an evaluation for the CSMA/CD protocol and a deterministic protocol under workloads anticipated in an industrial environment. Stochastic activity networks are used as the model type, and simulation is used as the solution method. The results show that the preferred resolution scheme depends on the level of workload anticipated and whether transient faults may occur.

I Introduction

Carrier sense multiple access protocols with collision detection (CSMA/CD) have been employed with success in many application areas. The common variants (Ethernet and IEEE 802.3) have very low delay under low loads and, due to the collision resolution algorithm employed, are still stable under relatively high loads. However, network stability is gained at the cost of delays at high loads, which is particularly undesirable in an industrial environment. An optimal protocol would have the low delay of CSMA/CD protocols at low loads and the low delay of token-passing-type protocols at high loads. CSMA/CD protocols with deterministic collision resolution (CSMA/DCR) are an attempt to achieve this goal.

There has been significant work directed specifically at adding deterministic collision resolution schemes to traditional CSMA/CD networks. Some notable examples include that by Takagi et al. (CSMA/CD with Deterministic Contention Resolution) [1]; Nute (Hybrid CSMA Protocol with High Utilization) [2]; Gburzynski and Rudnicki (Better-than-Token Protocol) [3]; Le Lann (802.3D Protocol) [4, 5]; Ciminiera et al. (Industrial IEEE 802.3 Networks) [6]; Capetanakis (Tree algorithms for packet broadcast channels) [7]; and, most recently, Chen and Li (Reservation CSMA/CD) [8]. These protocols seem to have advantages over traditional CSMA/CD-type protocols for industrial applications, under certain conditions [9]. However, there has been no detailed comparison of either protocol in an industrial environment when transient fault conditions may be present.

Transient fault conditions stem from a variety of sources, including electromagnetic pulses, radio-frequency interference, Electrostatic discharge, and electromag-

netic interference [10]. These sources of interference can be short lived, yet have long lasting ill effects on communication networks. In particular, short lived transient faults tend to last as little as 120 nanoseconds [10]. However, the effects on software (or firmware) that utilizes the interface hardware to the network can be significant. Specifically, a short lived transient fault can confuse the operation of a microcontroller that governs the interface to a network node. The effects are relatively long lived interruptions of network availability.

This paper will compare the performability [11] of the CSMA/CD protocol and a variant which employs a deterministic collision resolution, considering errors induced by transient faults. We will consider the long term effect that short lived transient faults may induce on the electronics which implement the protocols. We do this using stochastic activity networks, a stochastic extension of Petri nets. Stochastic activity networks permit us to accurately represent the detailed workings of both the CSMA/DCR and the traditional CSMA/CD protocols as well as the effects of transient faults.

II Overview and Assumptions

The CSMA/CD protocol considered compiles closely with the IEEE 802.3 standard [12], and hence will not be discussed in detail. The DCR protocol considered is an adaptation of the tree walk protocol, where a transition is immediately made to a leaf-level deterministic collision resolution upon a detected collision. This variant was chosen for evaluation due to the availability of a detailed specification [13].

In general, the leaf-level CSMA/DCR protocol behaves like the traditional CSMA/CD protocol with one notable difference: the protocol will revert to an implicit token-

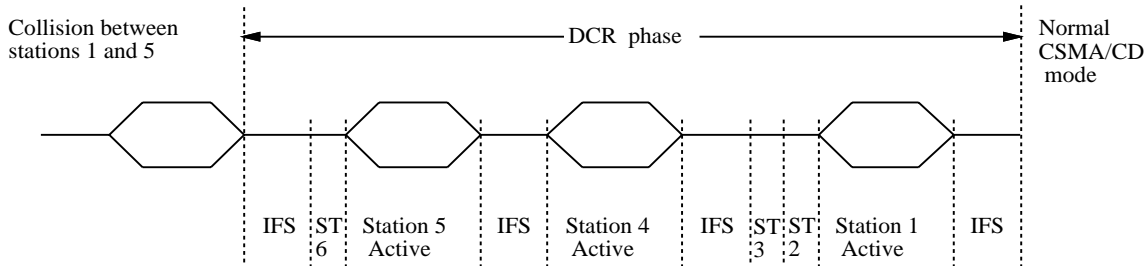


Figure 1: CSMA/DCR — Collision Resolution Phase Timing Diagram

passing algorithm when a collision is detected. The deterministic collision-resolution phase will involve all the stations, whether or not they participated in the collision [13, 14]. The resolution phase is divided into a fixed number of slots, with each station having a unique slot assignment. Each station is allowed to transmit in its specified time slot after one interframe space (IFS) or slot time (ST), as illustrated in Figure 1. If a station has nothing to transmit, its time slot will be forfeited after one slot time. The time slot forfeiture speeds up the resolution phase when only a few stations are involved in the collision.

As stated in the introduction, we assume operation in an industrial environment, where messages are typically short and may have varying priority levels. It is assumed that three types of data will be transmitted, as discussed in [15]. At the highest priority level (denoted as *type 3* data in the following), we assume the existence of a cyclic, repetitious communications as found in industrial local area networks [4, 5, 16]. The second type of data (*type 2*) is status and information sharing; it is given second priority. The last data type (*type 1*) is used for transmission of all other information and is given the lowest priority.

Since *type 3* data is cyclic in nature, the time between arrivals of this type of data

Table 1: Minimum Slot Time Computation

Parameter	Time (μs)	Equiv. # Bits
<i>Round Trip Propagation Time (RTPT)</i>	5.0	10
<i>Collision Detect Time (CDT)</i>	.5	1
<i>Jam Time (JAM)</i>	16	32
Total	21.5	43

is assumed to be normally distributed with a specified mean and very small (10% of the mean) variance. The time between arrivals of type 2 and type 1 data is assumed to be exponentially distributed with specified parameters. Furthermore, the fraction of the total system workload for these three types is assumed to be constant, with type 3 data comprising 60% of the total data, type 2 comprising 30%, and type 1 data comprising 10%.

Regarding message length, we assume that the length of data given to the Media Access Control (MAC) sublayer, per frame, is uniformly distributed between 1 and 128 bytes for all three data types. This is consistent with our assumed environment, where most messages would be short and of a control or status nature.

Regarding the network itself, we assume that all stations are identical, and the distance between adjacent stations is 10 meters. In addition, we assume that the bus has a raw speed of 2 Mbs with a maximum cable length of 500 meters (in IEEE terminology, a 2BASE5 system). This assumption was made for compatibility with the specification of [13, 14]; however, the results should scale to other media access bus speeds. Given this assumption for bus speed, it is appropriate to change certain parameters from their default values, as specified in [12]. In particular, the minimum

Table 2: MAC Sublayer Packet Format

Element	Bytes	Trans. Time (μs)
<i>Preamble</i>	7	28.0
<i>Start Frame</i>	1	4.0
<i>Destination Address</i>	2	8.0
<i>Source Address</i>	2	8.0
<i>Length</i>	2	8.0
<i>Data</i>	1 to 128	4.0 to 512.0
<i>Pad</i>	none	0.0
<i>CRC</i>	2	8.0
<i>Min. Packet Size</i> (MIN)	17	68.0
<i>Max. Packet Size</i> (MAX)	144	576.0

Table 3: 2BASE5 Network Parameters

Parameter	CSMA/CD	CSMA/DCR
<i>Jam Time</i> (JAM)	16 μ s	16 μ s
<i>Idle Time</i> (IDLE)	10 μ s	10 μ s
<i>Slot Time</i> (ST)	32 μ s	32 μ s
<i>Interframe Spacing</i> (IFS)	<i>n.a.</i>	10 μ s

slot time must be longer than the sum of the round trip propagation time, the collision detection time, and the jam time. For a 2BASE5 network, these times are given in Table 1. Table 2 defines the MAC sublayer packet format and Table 3 defines the protocol parameters used in the simulations.

We have made several assumptions regarding the occurrence of transient faults in the model. We assume that the transient fault enters the network at the physical connection between the station and the physical media, and that all stations on the network incorporate a microcontroller to implement the desired network communication protocol. Hence, as cited in [10], short-lived transient faults can disturb the

normal operation of a microcontroller system. Furthermore, we assume that the microcontroller incorporates a time-out device to reset itself in the event of a long lived error condition. Time-out devices, commonly known as watchdog timers, typically have fixed intervals in which to reset a microcontroller in the event of an error condition. Therefore, we assume that transient faults, upon arrival, will always cause a microcontroller to induce an error condition on the network of 10 milliseconds in duration until the watchdog timer resets the offending microcontroller. The subsequent error condition will be perceived as a collision on the bus or have the effect of damaging data packets.

In addition, the time between the error episodes is assumed to be exponentially distributed. The net effect due to error conditions will effectively reduce bus availability. Since little is known about the frequency of transient faults on local area networks, we varied widely their rate of occurrence. Results obtained in earlier studies [9] indicate significant variation in performance over a 10% increase in bus utilization. Therefore, the effects between 1% and 10% reduction of bus availability will be considered. Given that the error condition will last 10 milliseconds implies that the average time between error episode will be varied between one second and $\frac{1}{10}$ seconds.

Furthermore, we assume that each station will transmit with equal probability, and that each receiving station has an equal probability of being the destination node. With these assumptions in mind, consider the time-space diagram in Figure 2, which illustrates the propagating effect transient faults may have on the network. In Figure 2, the horizontal axis represents the physical placement of stations on the cable. The vertical axis represents time, increasing from top to bottom. As time increases, the

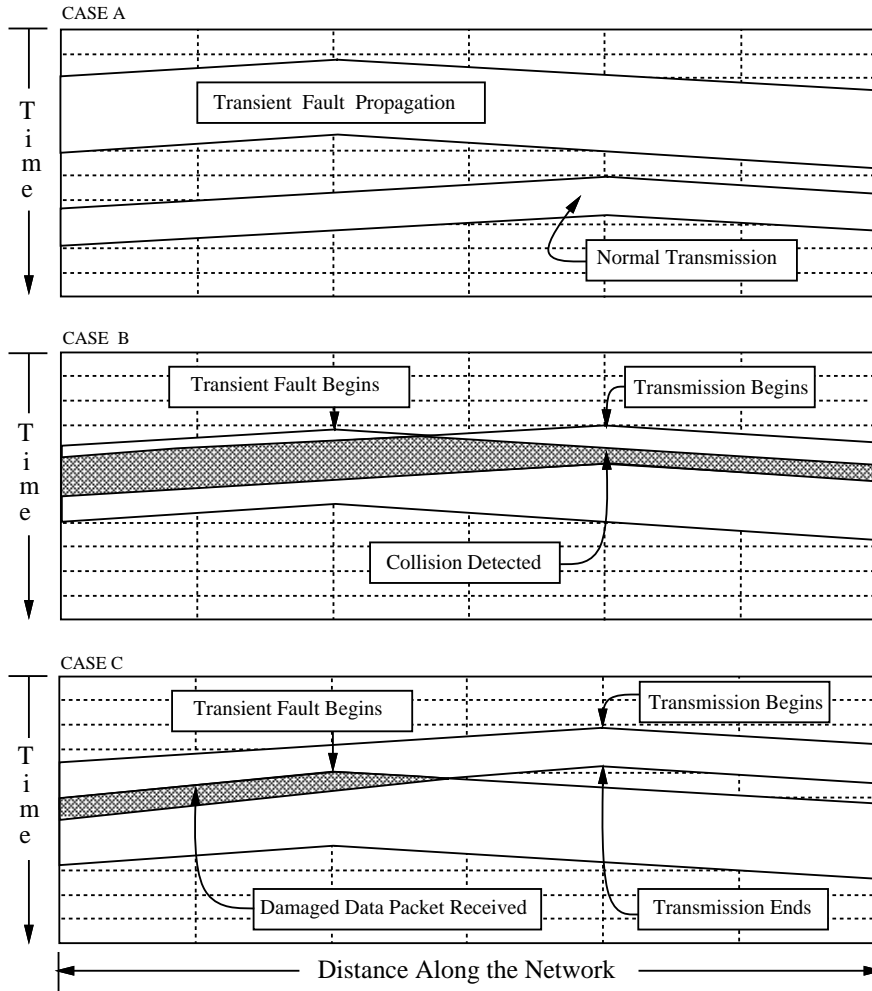


Figure 2: Time-Space Diagram of Transient Fault Scenarios

transmission of data, or error episode, will propagate to the adjacent stations.

During the propagation of signals, there are three situations which may occur. The first situation, labeled “CASE A” in Figure 2, illustrates a transient fault condition on an idle bus. A fault of this type will not effect the functioning of the network, since the subsequent error expires prior to the transmission of a data packet. Case B illustrates the most common effect when a transient fault damages a data packet. In this situation, the transmitting station perceives the error condition as a collision, and

merely schedules the packet for re-transmission, as defined for that specific protocol. Case B is thus modeled as a jam condition on the bus in both the CSMA/CD and CSMA/DCR protocols.

The final situation, case C, is a scenario where the transmitting station has completed its transmission prior to the impending error episode. This will occur if a transient fault corrupts a packet, but the station ceases transmission (hence will not detect the error) before news of the error propagates back to the station. In this case, the transmitting station will not be influenced by the impending error episode, and the receiving station detects the error as a fragmented packet or polynomial code error. This is shown in Figure 2, where an adjacent station is influenced by a transient fault causes an error condition which propagates along with the message intended for the receiving station. Therefore, case C is modeled as though the receiving station has detected a protocol error. If the data type is of type 1 or 2, then an upper layer protocol timer will time out (i.e., connection oriented) and request retransmission of the packet.

III SAN Models

As stated in the introduction, we use “stochastic activity networks,” (SANs) [17, 18, 19, 20] a stochastic extension to Petri nets, as our model representation. Due to space limitations, a description of stochastic activity networks is not presented in this paper. Instead, the reader is urged to consult the relevant references for additional information.

A CSMA/CD Network Model

Figure 3 illustrates a stochastic activity network model of a single CSMA/CD network node. The activity parameters, output gate functions, and input gate functions for the CSMA/CD protocol model is listed in Tables 4, 5, and 6 respectively. As discussed in a previous section, packets of three priorities are considered, representing the three data types (real-time, status information, and non-real-time). In the SAN model, the input queue for each data type is represented by a place (denoted as Q_T1_N , Q_T2_N , and Q_T3_N). Tokens within each of the places represent packets ready to be transmitted on the network. Place Q_T3_N represents queued packets of the highest priority (real-time data), while Q_T1_N represents queued packets of the lowest priority (non-real-time). The arrival of packets of each data type is modeled by timed activities (Arr_T1_N , Arr_T2_N , and Arr_T3_N). Each of the input gates associated with these activities ($Block_T1_N$, $Block_T2_N$, and $Block_T3_N$), represents the finite nature of each queue, by disabling the corresponding activity when the queue size is reached.

The scheduling of packets (according to their priorities) to be transmitted is done using the input gate Q_Sel_N . The predicate of this gate is such that it holds if there is a packet in any of the three queues, and the function is such that it removes a packet from the queue of highest priority. Activity Q_load_N is attached to Q_Sel_N , and allows the data packet of highest priority into the internal queue ($Queue_N$). It is enabled if the channel ($Channel_N$) is idle, the internal queue ($Queue_N$) is empty, and there is at least one data packet in the arrival queues (Q_T1_N , Q_T2_N , or Q_T3_N).

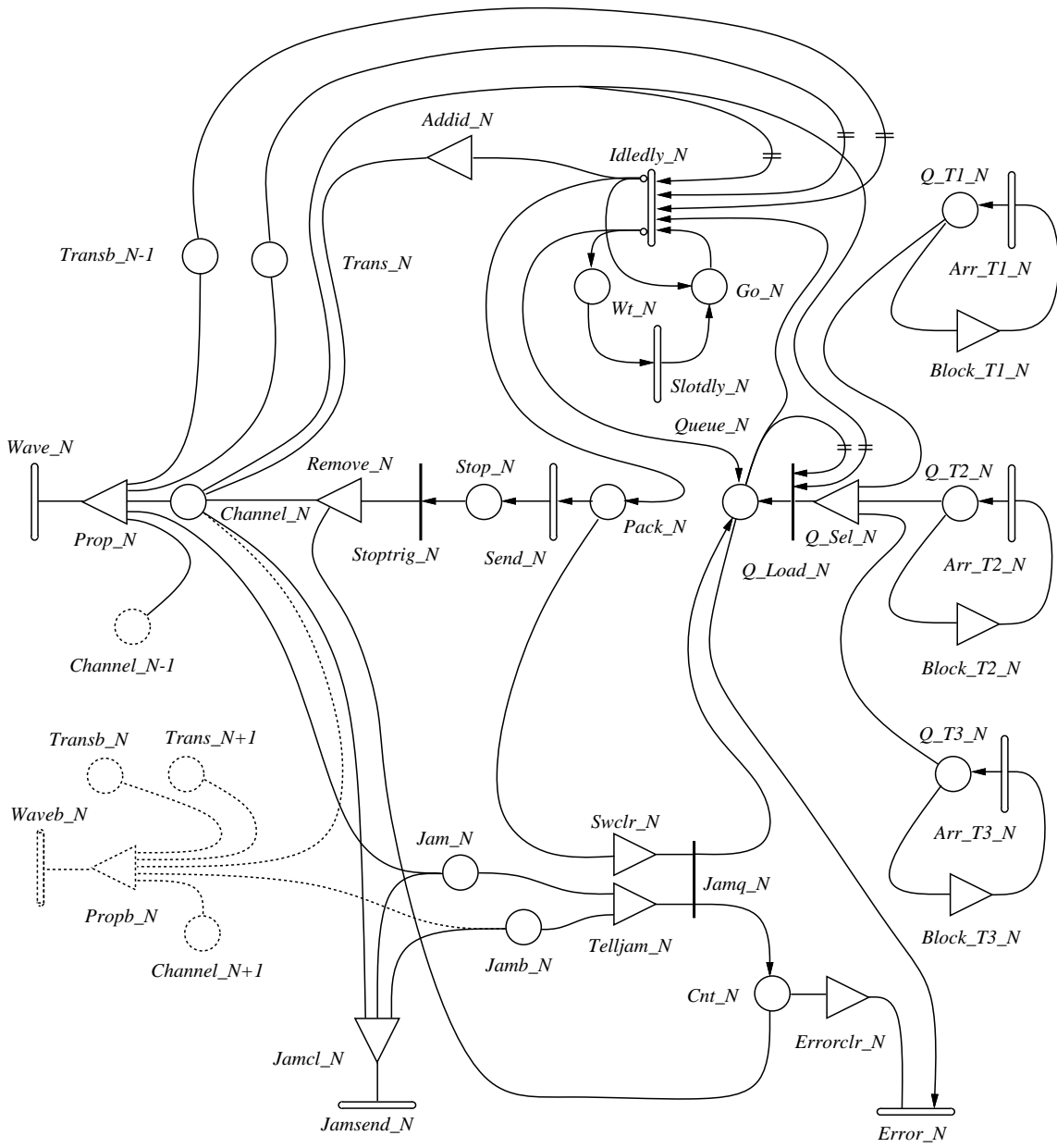


Figure 3: CSMA/CD SAN Model of a Single Station

After a packet enters the internal queue, it can begin transmission. The process by which this is done depends on whether or not the station has just been involved in a collision, and is governed by the standard binary backoff algorithm. The binary backoff algorithm is modeled, in part, by three places (*Queue_N*, *Wt_N*, and *Go_N*). In addition, two timed activities are required (*Slotdly_N* and *Idledly_N*). The timed activity *Idledly_N* performs the function of delaying until the channel becomes idle. The cases associated with *Idledly_N* determine whether the station will attempt to capture the bus (the top case) or will defer for a slot time (the bottom case) as modeled by the timed activity *Slotdly_N*. The probabilities are determined by the number of previous collisions that the station had encountered. The probability that a station will attempt to capture the bus is one if there are no tokens in the place *Cnt_N*. Otherwise, the probability is reduced by powers of two as specified by the binary backoff algorithm.

Given that a transmission is attempted, its time must be modeled. The transmission time for a packet is modeled by the timed activity *Send_N*. As per our assumptions, the activity time of *Send_N* is uniformly distributed between the minimum packet size and the maximum packet size (see Table 2).

Upon completion of the activity *Send_N*, a token is placed in *Stop_N*, indicating the completion of the transmission for that packet. Termination of a successful transmission is accomplished by the output gate denoted as *Remove_N*. *Remove_N* has the function of removing the tokens representing the packet from the place *Channel_N* and resets the count of transmission attempts (*Cnt_N*) to zero.

The status of the channel, as seen by the station, is determined by the marking of place *Channel_N*. If there are no tokens in *Channel_N*, then the station is idle. A

Table 4: Activity Parameters for CSMA/CD SAN Model

Activity	Rate	Probability	
		case 1	case 2
<i>Arr_T1_N</i>	$\exp(\lambda_1)$	-	-
<i>Arr_T2_N</i>	$\exp(\lambda_2)$	-	-
<i>Arr_T3_N</i>	$\text{normal}(\mu, \sigma)$	-	-
<i>Q_Load</i>	inst	-	-
<i>Idledly_N</i>	determ(IDLE)	binary backoff	binary backoff
<i>Slotdly_N</i>	determ(ST)	-	-
<i>Send</i>	uniform(MIN, MAX)	-	-
<i>Stoptrig_N</i>	inst	-	-
<i>Wave_N</i>	determ(0.05 μs)	-	-
<i>Waveb_N</i>	determ(0.05 μs)	-	-
<i>Jamq_N</i>	inst	-	-
<i>Jamsend_N</i>	determ(JAM)	-	-
<i>Error_N</i>	determ(≈ 0 μs)	-	-

Table 5: Output Gate Parameters for CSMA/CD SAN Model

Gate	Function
<i>Addid_N</i>	MARK(<i>Channel_N</i>) = Channel Identification
<i>Remove_N</i>	MARK(<i>Channel_N</i>) = 0; MARK(<i>Cnt_N</i>) = 0;

single token represents a collision. A marking of $N + 1$ tokens represents the channel identification N . The places $Trans_N$ and $Transb_N-1$ are used to model the propagation delay of the physical media. A collision is detected when the input gate $Prop_N$ detects a difference in the marking of $Transb_N-1$ and $Trans_N$ upon completion of the timed activity $Wave_N$. The input gate $Prop_N$ is a relatively complicated function and is used to model the propagation of the channel identification, collision, and idle condition. Upon detecting a collision, the input gate $Prop_N$ will replace the current marking of $Channel_N$ with a single token, indicating a collision. The single token will propagate to each of the adjacent channels via $Transb_N-1$ and $Trans_N$. The adjacent stations continue the propagation of the channel status in the same fashion.

When a collision is detected, input gate $Prop_N$ will place a token in the place Jam_N . Similarly, the adjacent station, after the physical layer propagation delay, will detect a collision and a token will be placed in $Jamb_N$. These two places (Jam_N and $Jamb_N$), start the collision recovery mechanization as described in the IEEE 802.3 standard. Specifically, the station detecting the collision will initiate the transmission of a jam signal, persisting for a time specified by the activity $Jamsend_N$. Upon completion of $Jamsend_N$, the input gate $Jamcl_N$ removes the tokens from the three associated places, leaving the channel idle. Furthermore, if this station was transmitting when the collision was detected, as indicated by a marking in the place $Pack_N$, the input gate $Swclr_N$ along with $Telljam_N$ will enable the instantaneous activity $Jamq_N$. $Jamq_N$ then places a token in the place denoted as Cnt_N . (Recall that the place Cnt_N is used to change the probability for this station's chance of attempting to capture the channel when the channel becomes idle.) In addition, the token in $Pack_N$ is transferred back to the internal queue denoted as $Queue_N$ for

re-transmission.

The stochastic activity network model for the entire network is constructed by connecting multiple copies of these models together. All stations are identical, except the two end stations, whose propagation functions differ slightly.

B CSMA/DCR Network Model

Figure 4 illustrates a CSMA/DCR stochastic activity network model of a single station. The CSMA/DCR SAN model closely resembles the CSMA/CD model as illustrated in Figure 3 and, hence, does not require the detailed discussion given to the first model. The activity parameters and gate functions that are similar to the CSMA/CD model apply to the CSMA/DCR model. Therefore, the lists of parameters in Tables 4, 5, and 6 are directly applicable to the CSMA/DCR model. We will, however, identify the areas of the CSMA/DCR model that differ from the CSMA/CD model including activities and gate functions as listed in Tables 7, 8, and 9.

The first area that differs from the CSMA/CD model is the timed activity denoted as *Idledly_N*. In this model, the probability of attempting a transmission is one, if the enabling conditions are true, since collisions are resolved using the deterministic collision resolution algorithm discussed earlier. The DCR portion is the collection of places, gates, and activities, illustrated at the bottom of Figure 4, that is different from those in the CSMA/CD model. To understand the functioning of this portion of the model, consider the case when the protocol is not in the DCR phase. In this situation, the markings of places *DCR_N*, *Slot_Cnt_N*, and *Serviced_N* are such that the input gate *DCR_Enable_N* holds and is an enabling condition for the timed activity *Idledly_N*. The protocol thus behaves, in this case, as a 1-persistent

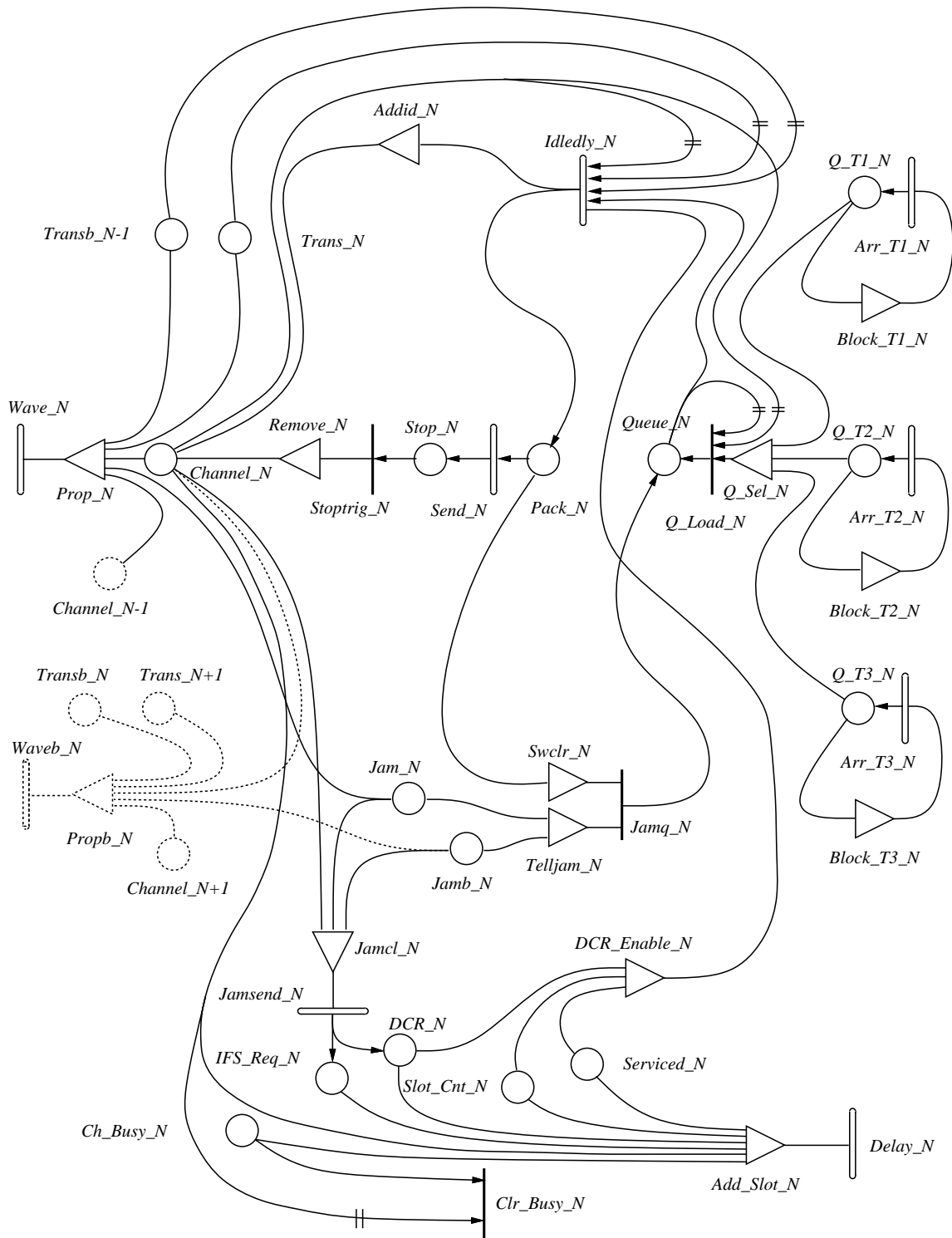


Figure 4: CSMA/DCR SAN Model of a Single Station

CSMA/CD protocol. If the channel is idle, then the probability of attempting to capture the channel is one. The 1-persistent protocol remains intact as long as there is no detected collision.

Consider now what happens when a collision is detected. (The detection of a collision is identical to that in the CSMA/CD protocol.) The DCR phase starts by the completion of the timed activity *Jansend_N*, upon which a token is placed in *IFS_Req_N* and *DCR_N*. The input gate *DCR_Enable_N* no longer holds, disabling the timed activity *Idledly_N* from capturing the channel until the marking of the place *Slot_Cnt_N* is equal to the pre-assigned channel identification for that specific station. In addition, the marking of the place *Serviced_N* will hold a token if this station has sent a packet during its allocated slot time. The place *Serviced_N* controls the enabling of *DCR_Enable_N*, disallowing multiple packet transfers during the DCR phase by a single station.

Synchronization among stations is achieved by keeping track of the idle and busy periods observed during the collision resolution phase. More specifically, the timed activity *Delay_N* will delay for one of two times, depending on the marking of the place *IFS_Req_N*. If the place *IFS_Req_N* has a token, then the delay time will be the value of the interframe spacing as described earlier, and illustrated in Figure 1. Otherwise, the delay period is the required slot time. Upon the completion of the delay time (*Delay_N*), the place *Slot_Cnt_N* will be modified to indicate the next slot position or the conclusion of the DCR phase. In addition, the place *IFS_Req_N* is revised to indicate the appropriate delay time now required based on the last slot time channel activity. The interframe spacing is required only if the previous station had transmitted a packet, as indicated by the place *Ch_Busy_N* and the occurrence

of an idle channel. It should be noted that the DCR phase is subject to additional transient faults. Should a transient fault occur during the DCR phase, the protocol simply resets, and starts over with the first logical station receiving preference.

As with the previous protocol, the model for the entire network is constructed by connecting multiple copies of the station model together, except for the end stations whose propagation functions differ slightly.

C Transient Fault Generator

Figure 5 illustrates the SAN model for the transient fault generator. The figure can be described in two parts. The first part models the generation of the transient fault and the beginning of an error episode, as illustrated in the left portion of the figure. The second part is the representation of the three scenarios discussed earlier, as illustrated in the right portion of the figure. The activities, output gate functions, and input gate for the transient fault generator are listed in Tables 10, 11, and 12. The modeling of an error episode is done using the place denoted *Enable_tp_p* and the input gate *Enable_tf_i*. The time between error episodes is assumed to have an exponential distribution, as modeled by the timed activity *Tf_arrival*. Upon completion, the timed activity *Tf_arrival* will place a token in each place noted as *Tf_p*, *RR_p*, and *Tf_in_progress*. A token in *Tf_p* enables the timed activity *Tf_id_a*, which will ultimately determine the inflicted station. Specifically, there are n cases associated with the timed activity *Tf_id_a*. Each case has an equal probability of being selected. The n cases correspond to the n stations in the network to be modeled and there are n copies of the output gate function *Tf_id_n*¹.

¹In Figure 5, only case 1 is fully illustrated while cases 2 through n are partially illustrated.

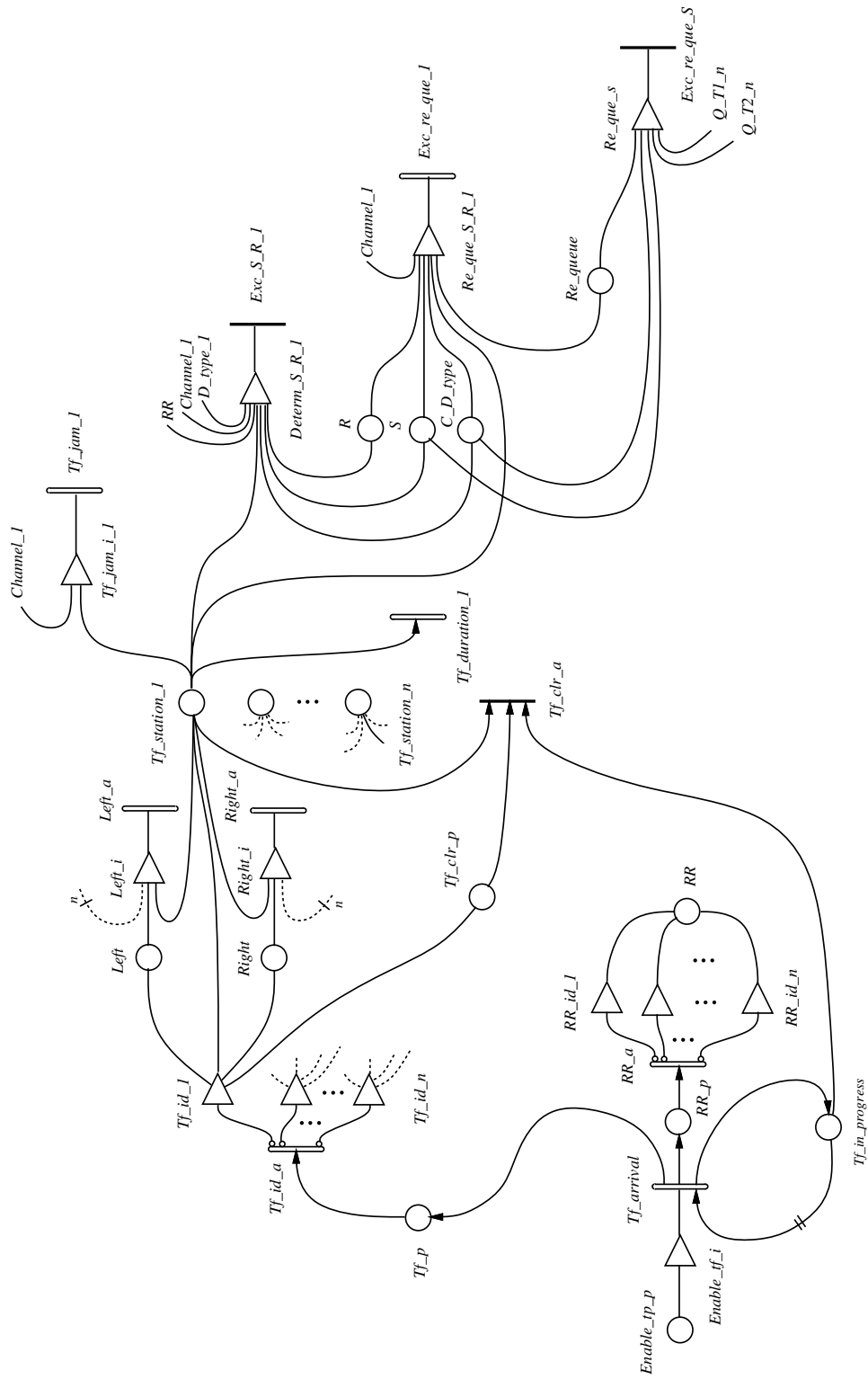


Figure 5: Transient Fault Generator

The output gate function associated with each case of the timed activity (Tf_id_a) changes the marking of the global places $Left$, $Right$, and Tf_clr_p . Initially, the marking of places $Left$ and $Right$ contains a number of tokens equal to the station number identifying the initial location of the error condition at the beginning of the error episode. The places $Left$ and $Right$ are used in conjunction with the input gates ($Left_i$ and $Right_i$) to propagate the error episode to each adjacent station. The propagation of the error episode is accomplished by the timed activities noted as $Left_a$ and $Right_a$, along with the associated input gates. The activity timed for both $Left_a$ and $Right_a$ is deterministic and has values equal to the propagation delay associated with the physical distance between stations.

Effectively, after the completion of the timed activity, the input gate $Left_i$ will propagate the error episode to the left by placing a token in the adjacent $Tf_station_n-1$ and removing a token from the place $Left$ ². The propagation continues in this manner until $Left$ no longer contains any tokens, indicating the end of the bus. Similarly, the propagation to the right is accomplished by the timed activity $Right_a$ and the input gate $Right_i$. The input gate $Right_i$ will propagate the transient fault to the adjacent $Tf_station_n+1$ after each propagation time period. The place $Right$ will receive another token until it has reached the maximum number of stations modeled in the network. Finally, the place noted as Tf_clr_p holds a token to indicate that an error episode is present on the network. Tf_clr_p , along with the instantaneous activity Tf_clr_a , is used to remove the token from the place $Tf_in_progress$ when all the error episode indicators ($Tf_stations_n$) have terminated.

Next, it is necessary to determine, relative to the transmitting station, which

²In Figure 5, there are n $Tf_station_n$; only the first is fully illustrated.

station is the destination station. Knowledge of the address of the destination station is required to model the condition described as case C in Figure 2. If a damaged data packet is detected by the destination station, it is necessary for the destination station to request a retransmission from the source station. This selection is modeled by the timed activity RR_a along with the associated places and output gates. Since we assume that each non-transmitting station is equally likely to be the destination node, the probability of each case of RR_a is equal. The selected case for the activity RR_a will permit the associated output gate function to mark the place RR with the proper number of tokens as an indication of the relative location of the destination station (the destination station is relative to the transmitting station).

In general, the right side of the transient fault model has n stations, each having a place noted as $Tf_station_n$. Each of the $Tf_station_n$ places is connected to replicate copies of the input gates $Tf_jam_i_n$, $Determ_S_R_n$, and $Re_que_S_R_n$ and the associated activities for the input gates. Figure 5 fully illustrates the first set of input gates and activities, $Tf_jam_i_1$, $Determ_S_R_1$, and $Re_que_S_R_1$. (Note that the suffix 1 identifies the specific gate or activity associated with place $Tf_station_1$.) The remaining places $Tf_station_2$ through $Tf_station_n$ and the associated input gates 2 through n are not illustrated in Figure 5.

We now consider the representation of the three effects an error may have on the network. Before doing this, it is important to understand that there are four places common to the replicate copies of the input gates $Determ_S_R_n$ and $Re_que_S_R_n$: R , S , C_D_type , and Re_queue . In addition, there is a global input gate noted a Re_que_s and associated instantaneous activity $Exc_re_que_s$. In the discussion which follows, the effects are described as seen by an arbitrary station n , since the mechanism

by which error episodes propagate has already been seen.

Case A, where an error episode does not affect a transmitting or receiving station, is modeled by the place $Tf_station_n$ and the timed activity $Tf_duration_n$. In the event that a station does not become active before the timed activity $Tf_duration_n$ completes, then a token will be removed from the place $Tf_station_n$, terminating the error condition for that station.

Case B, where an error episode causes, in effect, a collision on the bus, is modeled by the input gate $Tf_jam_i_n$ and the timed activity Tf_jam_n . If the station $Channel_n$ is transmitting, then the condition will be modeled as a collision. Essentially, the gate and timed activity model the collision-detection time for the transmitting station. The effect is that the place $Channel_n$ will be marked with a single token, representing a collision on the bus. The collision is then propagated normally, as modeled by the CSMA/CD and CSMA/DCR SAN models.

Case C, where an error episode damages a data packet after the transmitting station ceases transmission, is a relatively complicated part of the transient fault model. First, the receiving station must detect that the packet is damaged. In addition, if the damaged data packet is of type 1 or 2, the transmitting station will retransmit the packet after a higher-level protocol timer times out. This is modeled by having the transient fault model place a token in the appropriate queue of the transmitting station, corresponding to the packet to be retransmitted.

More specifically, in case C, the detection of a damaged packet is modeled, in part, by the input gate $Determ_S_R_n$ along with the instantaneous activity $Exc_S_R_n$. The value of the predicate to the input gate $Determ_S_R_n$ depends on the status of the channel, place RR , and place $Tf_station_n$. Effectively, the input gate will hold

while there is activity on the network, the current station is being affected by an error episode, and the station is the receiving station. Under these conditions, and upon the completion of $Exc_re_que_n$, the input gate will change the marking of R , S , and C_D_type as follows. First, the receiving station is identified by placing a number of tokens in R representing an identification number. Then, the sending station is indicated by placing a number of tokens in S representing the sending station's identification number. In addition, the data type (type 1, 2, or 3) is preserved by placing the appropriate number of tokens in C_D_type .

If it is determined that a data packet was damaged, then it is necessary to determine if the data packet is of type 1 or 2 (recall that type 3 data packets are not rescheduled and therefore are not requested for retransmission). The input gate $Re_que_S_R_n$ and the global place noted as C_D_type are used to determine if queuing is necessary. All data types, except for type 3, will be re-transmitted.

Finally, if it was determined that the data was of type 1 or 2, then the receiving station will request a retransmission (it is assumed that the request will be scheduled at the same priority as the damaged data packet). The input gate Re_que_s , along with the place Re_queue , is used to perform the rescheduling of the receiver's request and the queuing of the data packet to be re-transmitted by the original sender. Essentially, the place Re_queue contains the number of tokens that represents the receiving station's identification number. In addition, the place S contains the original transmitting station's identification number and C_D_type defines the data type. Therefore, at the completion of the instantaneous activity $Exc_re_que_s$, the input gate function Re_que_s will queue the appropriate stations with the appropriate data type, provided that the specified queue is not full.

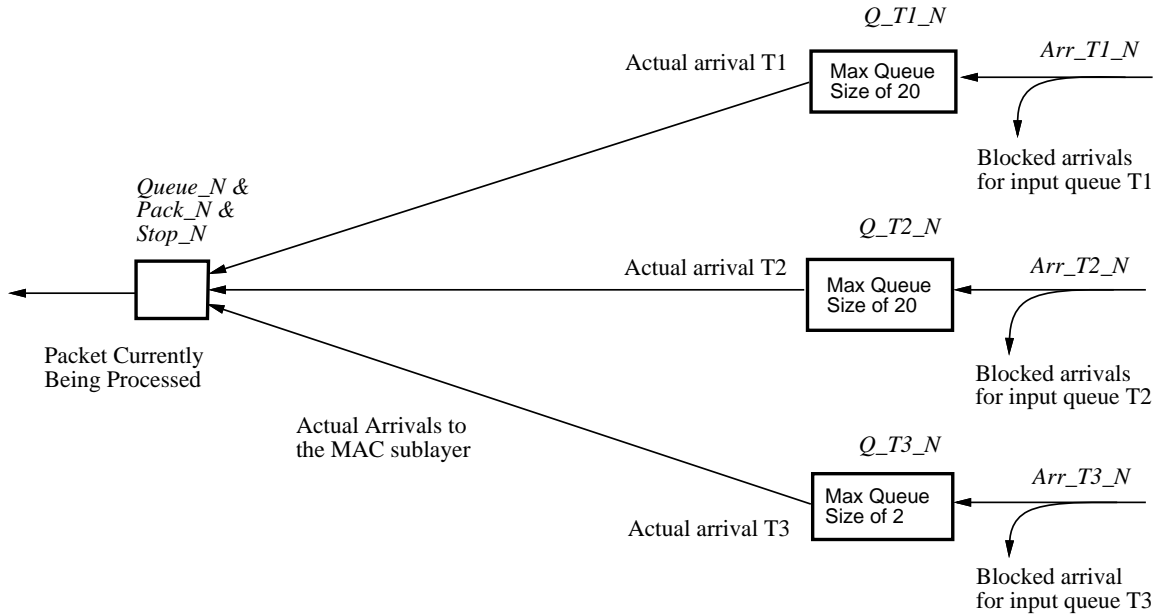


Figure 6: Packet Flow Through a Station

In summary, the transient fault model is used with both the CSMA/CD protocol and CSMA/DCR protocol for performability evaluation. The connection between the transient fault model and the two protocols being evaluated is defined by the specific places denoted as $Channel_n$, D_type_n , Q_T1_n , and Q_T2_n as illustrated in Figures 3, 4, and 5.

IV Performance Variables

In comparing the performability of the two models, we will use the expected response time as our metric. This delay is estimated using Little's result [21], after determining the blocking probabilities for each data type and the expected number of packets at each station. Figure 6 illustrates the flow of packets through a station, and shows how the expected response time is calculated. In this figure, the square

boxes represent queues of stated lengths, with the corresponding SAN model place names noted. Note that a packet is not removed from the final queue (notated “packet currently being processed”) until it is successfully transmitted. The flow of packets through a station is denoted by arrows. There are two ways a packet may leave a station, either by being rejected due to a queue being full, or being transmitted. To determine the expected delay time for each type of traffic, we first estimate the number of packets at each queue in the station, and the blocking probability at each input queue. We then use Little’s result to determine the expected time at each queue. Finally, for each type of traffic, we sum the expected delay times that are experienced at each queue visited to determine the total expected delay time for each data type.

V Discussion of Results

The results will be presented in two sections. The first section will present the results of a fault-free, 25 node network model. The second section will discuss the performability of the two protocols in the presence of transient faults. Simulation was used as the solution method, and done using a package known as METASAN³. All results are presented with confidence intervals at a 95% level of confidence.

A Performance Under Fault-Free Conditions

Figure 7 illustrates that there is a significant difference between the performance of the deterministic and non-deterministic collision resolution methods. Note that in a network of this size, the effects due to the queueing discipline are completely over-

³METASAN is a registered trademark of the Industrial Technology Institute.

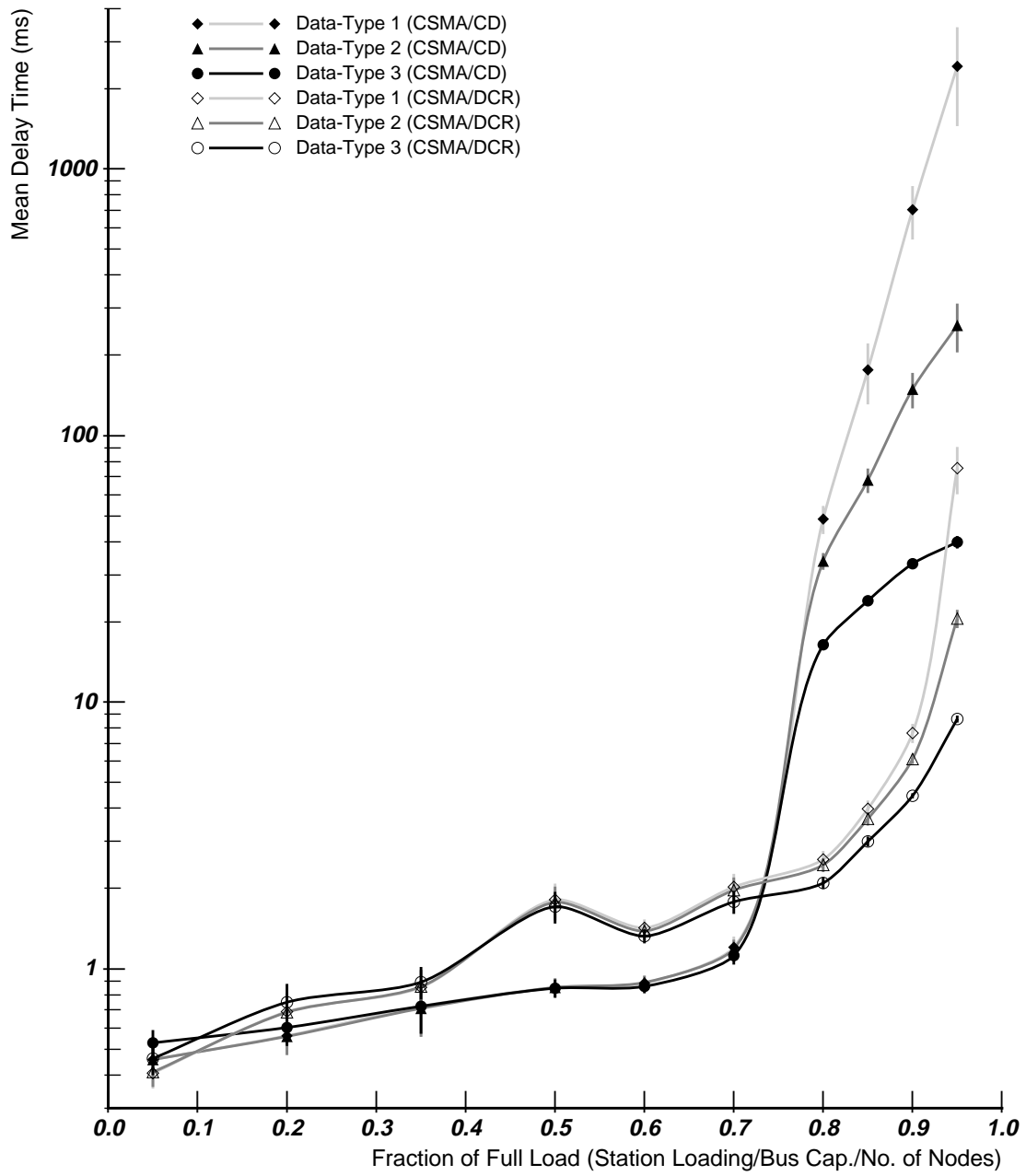


Figure 7: Expected Response Time for the 25-Node Networks

shadowed by the collision resolution method, and that there is a distinct region where each scheme performs better than the other. More specifically, the non-deterministic resolution method performs better when the offered load is less than 70%, while the deterministic resolution scheme performs better when the loading is higher than 75%. The degree of difference is more important; the non-deterministic scheme only offers a several millisecond advantage for low loads, while the deterministic scheme offers an advantage of *hundreds* of milliseconds at high loads. Also note that the non-deterministic scheme exhibits significant blocking of all data types at high loads, while the deterministic scheme exhibits no significant blocking. It is important to observe the effect of the non-zero blocking probability of the three data types in the CSMA/CD networks. The effect manifests itself as a rolloff of expected response time for the three data types around 80 to 95% workload. This is not a signal that the protocol is performing “better” at higher loads, only that the increase in delay time for transmitted packets as offered load increases is not as great. It is at this point that a significant number of the packets are being rejected at the input queues, and never receive service. This is clearly an undesirable situation.

Also note that there is a bulge (relative to the confidence interval width) in the delay time curve for the CSMA/DCR protocol around the 50% workload. This bulge may be related to the fact that the protocol is switching between deterministic and non-deterministic (CSMA/CD) modes, and that neither is efficient at this loading. This suggests that an immediate transition to “leaf mode” may be too abrupt, and that an adaptive tree walk, as proposed by some, may perform better.

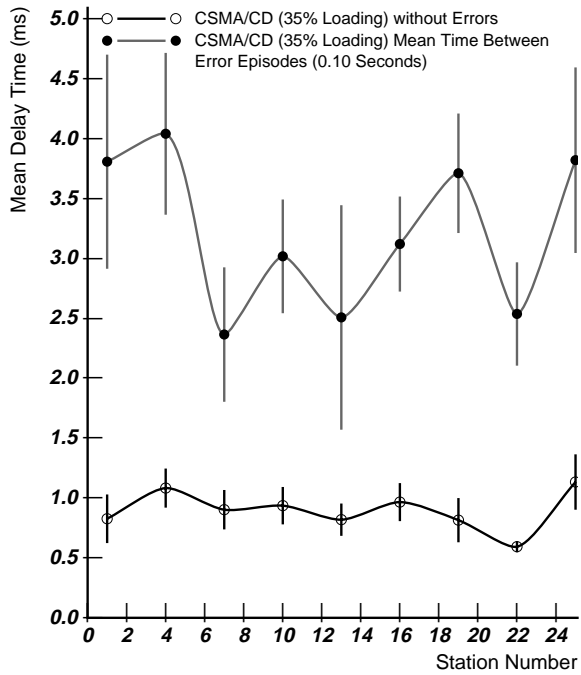
B Performability Under Transient Fault Conditions

To investigate the performability of the two protocols, we estimated the expected response time of the two protocols at two workloads (35% and 70% full load) as the expected time between error episodes was varied. As seen in the previous subsection, the effect of the queuing discipline is relatively insignificant at 35% and 70% loading. Therefore, only results for type-3 (real-time) data is presented here.

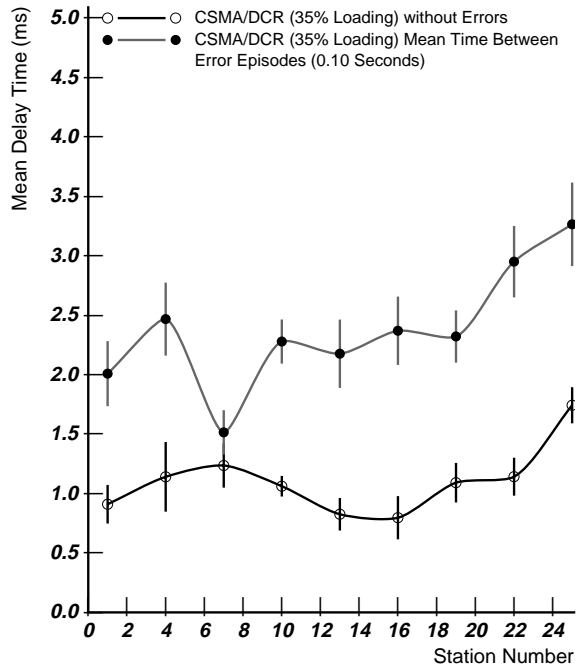
We have varied the mean time between error episodes from 1, $\frac{1}{3}$, $\frac{1}{7}$, to $\frac{1}{10}$ of a second, in order to investigate the effects of reduced bus availability of 1%, 3%, 7%, and 10% respectively. We considered these conditions in light of Figure 7 and the dramatic effects observed between 70% and 80% of full load for the CSMA/CD protocol.

Furthermore, we postulated that a variation in the expected response time for each of the stations in the network may exist. Therefore, the expected response time was estimated for every third station. Figure 8 gives the results for both protocols at both 35% and 70% workloads, under the maximum fault-level studied.

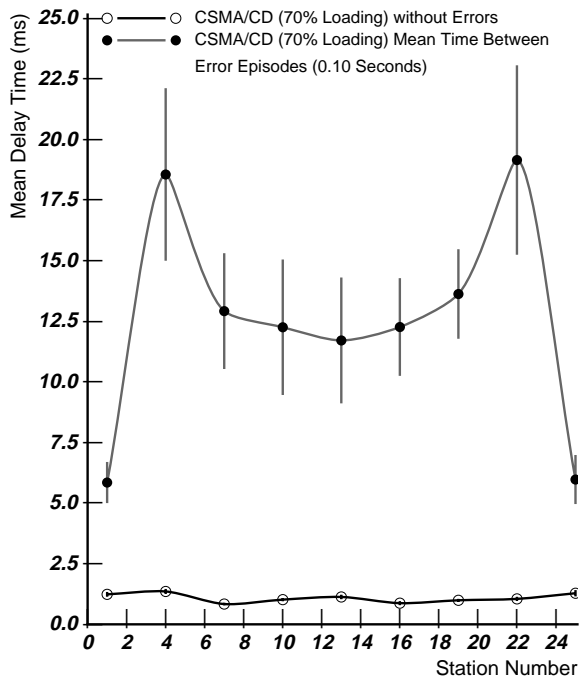
In the top-left graph, it is clear that the response time, when there are no transient faults, is the same, to the level of confidence reached, for all stations. As expected, when the mean time between error episodes increases to $\frac{1}{10}$ of a second, a measurable increase in the expected response time is observed. It would appear that there is a variation in this time for the various stations under these extreme error conditions. However, relative to the estimated confidence interval, the shape of the curve is not sufficiently significant to draw any conclusions. It should be noted that, relative to the confidence interval, the difference between the two curves is important.



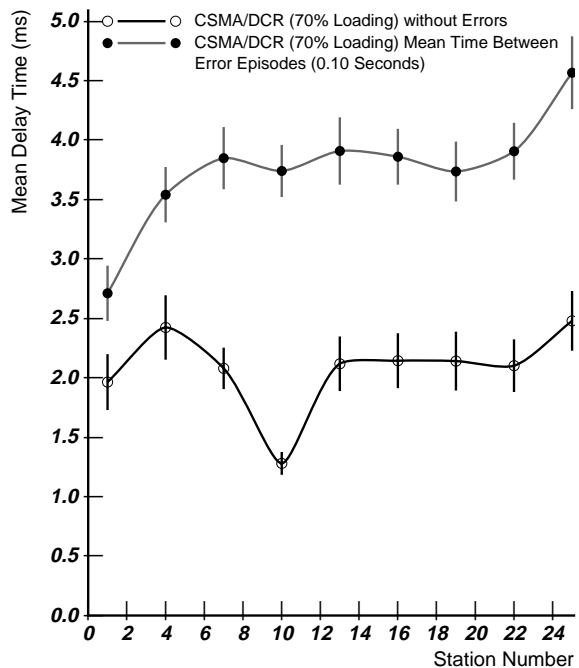
Mean Delay Time for CSMA/CD at 35% Full Load



Mean Delay Time for CSMA/DCR at 35% Full Load



Mean Delay Time for CSMA/CD at 70% Full Load



Mean Delay Time for CSMA/DCR at 70% Full Load

Figure 8: CSMA/CD and CSMA/DCR at 35% and 70% Loading - Expected Response

The top-right graph presents the results of the CSMA/DCR protocol at a workload of 35% full load, in both the fault and fault free cases. The interesting point to note about these two curves is that there appears to be a fairly constant offset between the fault and no-fault results, with the exception of one data point. However, relative to the estimated confidence interval and the precision of the simulation, the two curves do not provide strong evidence that a variation in expected response time exists between stations.

The lower-left graph presents the results of the CSMA/CD protocol for a workload of 70% full load⁴. Again, the response time, under no-fault conditions, is relatively constant for all stations. However, the instability of the CSMA/CD protocol is apparent at this loading and fault level. In addition to the extremely high expected response time experienced by stations 4 and 22, these stations also experienced a significant probability of blocking of type-3 requests. The large width of the confidence interval associated with the points on this curve makes it difficult to draw any conclusions from its shape. It is, however, evident that a significant difference exists between the expected delay time for the fault and no-fault cases.

The lower-right graph presents the results for the CSMA/DCR protocol at a workload of 70% full load. As in the other curves, the expected response time is relatively constant for each station, except for station 10. It is not clear why the dip at station 10 exists, and if the variation is significant. The curve also suggests a trend unique to the CSMA/DCR protocol. Relative to the confidence interval width, station 1 has a lower expected response time than station 25. One might expect such behavior with the CSMA/DCR protocol studied, which uses an implicit token-passing

⁴Note the change in the scale of the Y-axis in this graph, relative to the others on this page.

mechanism to resolve collisions. Recall that this version of the CSMA/DCR protocol reverts to a CSMA/CD protocol when a perceived collision occurs, followed by a jam signal, due to an error episode. The stations with lower addresses are favored under extremely high fault conditions, since the occurrence of a second transient fault while in the deterministic collision resolution mode will cause the network to abort the current resolution cycle. After this occurs, the network will again begin to offer service to each station, starting with the station with the lowest address.

Finally, Figure 9 provides a comparison of the two protocols with one another, when the expected time between the error episodes is varied. For a given fault level, protocol, and loading, the point plotted is the expected response time, averaged across all stations. The figure clearly demonstrates that the CSMA/CD protocol becomes unstable at a loading of 70%, when many error episodes occur. In contrast, the DCR protocol is relatively insensitive to increases in the expected time between error episodes, even at a very high loading. The CSMA/DCR protocol thus demonstrates superior stability over the CSMA/CD protocol, even under very heavy transient fault conditions.

VI Conclusions

The study presented 1) illustrates the utility of stochastic activity networks in representing both performance and dependability related behaviors in a single model and 2) provides significant insight into the performability of the two protocols studied.

In regard to the modeling approach used, it was seen that stochastic activity networks permit the representation of a relatively complex fault model as well as normal protocol operations. This is important when doing performability evaluations

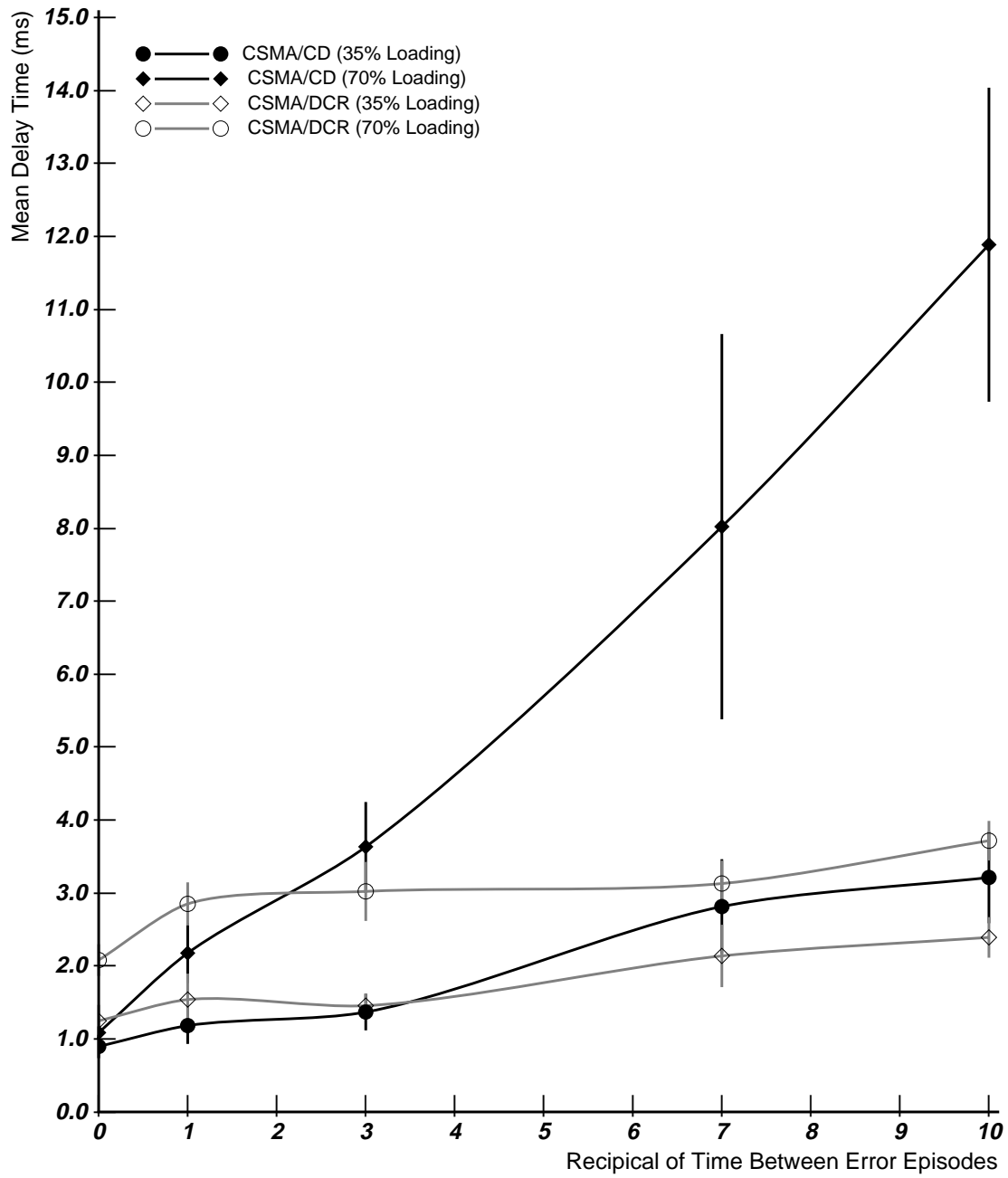


Figure 9: Expected Response Time versus Time Between Error Episodes

of this type. In addition, the fault model was built in a manner in which there was minimal interaction between it and the protocol models. The same fault model might thus be able to be used to investigate the performability of other protocols.

In regard the to the protocols themselves, the results of the study indicate that there are significant differences in the behavior of the deterministic and non-deterministic collision-resolution schemes, which depend on both the offered workload and rate at which transient faults occur. In the fault free case, there is a distinct region of offered workload in which each resolution scheme performs best. Thus, if the workload were relatively constant and known, there would be situations when each of the resolution schemes should clearly be chosen. However, if the offered workload for the application is not known or varies significantly, one should consider the absolute difference between the performance of the two collision-resolution schemes in each region. In this case, the non-deterministic scheme performs just slightly better than the deterministic scheme at low loads, while the deterministic scheme performs significantly better at high loads. Thus the deterministic resolution scheme offers real advantages when the offered workload is high, is not known, or varies significantly.

In the situation where transient fault conditions exist, the effects due to transient faults have a significant impact on the behavior of the non-deterministic collision-resolution scheme. The most pronounced effect is evident when the workload is heavy and the occurrence of transient faults is frequent. The non-deterministic collision-resolution protocol becomes unstable at loading of 70% of full load, when many transient faults occur. On the other hand, the deterministic-collision resolution protocol continues to perform in an acceptable manner, although there is some degradation in performance. Thus when transient faults are considered, the deterministic collision

resolution scheme clearly performs better than the non-deterministic scheme.

REFERENCES

- [1] A. Takagi, S. Yamada, and S. Sugawara, "CSMA/CD with deterministic contention resolution," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 877–884, Nov. 1983.
- [2] T. Nute, "HCSMA - A nearly conflict-free hybrid CSMA protocol with high utilization," in *IEEE 1984 Computer Networking Symposium*, (Los Angeles CA 90080), IEEE Computer Society, IEEE Computer Society Press, Dec. 1984.
- [3] P. Gburzynski and P. Rudnicki, "A better-than-token protocol with bounded packet delay time for Ethernet-type LAN's," in *1987 Simulation of Computer Networks*, (Los Angeles CA 90080), IEEE Computer Society, IEEE Computer Society Press, Aug. 1987.
- [4] G. Le Lann, "The 802.3D protocol: A variation on the IEEE 802.3 standard for real-time LAN's," tech. rep., INRIA-BP 105, F-78153 Le Chesnay Cedex, France, Oct. 1987.
- [5] G. Le Lann, "A deterministic 802.3 protocol for industrial local area networks," tech. rep., INRIA-BP 105, F-78153 Le Chesnay Cedex, France, 1987.
- [6] L. Ciminiera, C. Demartini, and A. Valenzano, "Industrial IEEE 802.3 networks with short delivery times for urgent messages," *IEEE Transactions on Industrial Electronics*, vol. 35, pp. 18–25, Feb. 1988.
- [7] J. I. Capetanakis, "Tree algorithms for packet broadcast channels," in *IEEE Transactions on Information Theory, Vol IT-25, No. 5*, Sep. 1979.

- [8] J. S. J. Chen and V. O. K. Li, "Reservation CSMA/CD: A multiple access protocol for LAN's," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 202–210, Feb. 1989.
- [9] W. H. Sanders and K. H. Prodromides, "A comparison of collision resolution schemes for carrier sense multiple access protocols," Technical Report CERL-TR-90-06, University of Arizona, Tucson, Arizona, July 1990.
- [10] T. Williamson, "Designing microcontroller systems for electrically noisy environments," Tech. Rep. 231792-002, Intel - Application Notes, Santa Clara, CA, Nov. 1988.
- [11] J. F. Meyer, "On evaluating the performability of degradable computing system," *IEEE Trans. Comput.*, vol. C-22, pp. 720–731, Aug. 1980.
- [12] ANSI/IEEE, *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification*. 345 East 47th Street, New York NY 10017: The Institute of Electrical and Electronic Engineering, Inc., 1985.
- [13] The Intel Corp., "83C152 universal communication controller users manual," Users Manual 270243-002, Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051, May 1987.
- [14] D. N. Atallah, "Peer-to-peer facilitates real-time communications," *Electronic Design News (EDN)*, vol. 33, pp. 179–185, Aug. 1988.

- [15] F. H. Phail and D. J. Arnett, "In-vehicle networking - serial communication requirements and directions," Tech. Rep. 860390, SAE Technical Paper Series, 1986.
- [16] P. Pleinevaux and J. D. Decotignie, "Time critical communication networks: Field buses," *IEEE Networks*, vol. 2, pp. 55–63, May 1988.
- [17] A. Movaghar and J. F. Meyer, "Performability modeling with stochastic activity networks," in *IEEE 1984 Real-Time Systems Symposium*, (Los Angeles CA 90080), IEEE Computer Society, IEEE Computer Society Press, Dec. 1984.
- [18] 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), pp. 106–115, July 1985.
- [19] W. H. Sanders, "Construction and solution of performability models based on stochastic activity networks," tech. rep., The University of Michigan, Ann Arbor, MI, Aug. 1988.
- [20] J. A. Convillion et al., "Performability Modeling with UltraSAN," *IEEE Software*, vol. 8, pp. 69–80, Sep. 1991.
- [21] J. D. C. Little, "A Proof of the Queueing Formula $l = \lambda w$," *Operations Research*, vol. 9, no. 5, pp. 383–387, 1961.

AUTHORS

Kevin H. Prodromides was born in Pocatello Idaho in 1954. After receiving his BS in engineering in 1978 from Idaho State University, he joined United Technologies as a system design engineer and was primarily involved in designing full authority electronic jet engine fuel controls. Kevin further received two degrees; A MSEE in 1981 and a MSCS in 1991 from Rensselaer Polytechnic Institute during which time his professional career lead him to Allied-Signal Corporation as a system/project engineer in the Monitoring and Advanced Products group. Kevin's current responsibilities include communication networks for embedded controllers and monitoring equipment.

William H. Sanders (S'81-M'88) received the B.S.E. degree in Computer Engineering in 1983, the M.S.E. degree in Computer, Information and Control Engineering in 1985, and the Ph.D. degree in Computer Science and Engineering in 1988, all from the University of Michigan.

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Arizona. His areas of research are performance/dependability evaluation, computer networks and protocols, fault-tolerant computing, and manufacturing systems. Prior to joining the faculty at the University of Arizona, he was a Research Associate at the Industrial Technology Institute. His work at ITI included the development of a software package known as METASAN. More recently, he lead a team at the University of Arizona in research leading to the development of *UltraSAN*, a X-window based tool for evaluating systems represented as stochastic activity networks.

Dr. Sanders is a member of Sigma Xi, Eta Kappa Nu, The Institute of Electrical and Electronics Engineers, and the Association for Computing Machinery. In 1989, 1990 and 1991 he was the recipient of a Digital Faculty Award, as part of Digital's Faculty Program: Incentives for Excellence.

Table 6: Input Gate Parameters for CSMA/CD SAN Model

Gate	Enabling Predicate	Function
$Block_T1_N$	$MARK(Q_T1_N) \leq MAX_SIZE_T1$	NULL Function
$Block_T2_N$	$MARK(Q_T2_N) \leq MAX_SIZE_T2$	NULL Function
$Block_T3_N$	$MARK(Q_T3_N) \leq MAX_SIZE_T3$	NULL Function
Q_Sel_N	$MARK(Q_T1_N) > 0$ or $MARK(Q_T2_N) > 0$ or $MARK(Q_T3_N) > 0$	if $MARK(Q_T3_N) > 0$ $MARK(Q_T3_N) - -$ else if $MARK(Q_T2_N) > 0$ $MARK(Q_T2_N) - -$ else $MARK(Q_T1_N) > 0$ $MARK(Q_T1_N) - -$
$Prop_N$ Internal nodes End nodes require special consideration	$(MARK(Channel_N) == 1$ and $MARK(Jam_N) == 0)$ or $(MARK(Channel_N) > 1$ and $MARK(Trans_N) == 0)$ or $(MARK(Channel_N) == 0$ and $MARK(Trans_N) == 1)$	Controls propagation forward on the network based on channel status. $MARK(Channel_N) == 0$; Idle $MARK(Channel_N) == 1$; Coll. $MARK(Channel_N) > 1$; Busy
$Propb_N$ Internal nodes End nodes require special consideration	$(MARK(Channel_N) == 1$ and $MARK(Jamb_N) == 0)$ or $(MARK(Channel_N) > 1$ and $MARK(Transb_N) == 0)$ or $(MARK(Channel_N) == 0$ and $MARK(Transb_N) == 1)$	Controls propagation backward on the network.
$Jamcl_N$ Internal nodes	$MARK(Channel_N) > 0$ and $MARK(Jam_N) > 0$ and $MARK(Jamb_N) > 0$	$MARK(Channel_N) = 0$; $MARK(Jam_N) = 0$; $MARK(Jamb_N) = 0$;
$Swclr_N$	$(MARK(Pack_N) > 0$	$MARK(pack_N) = 0$
$Telljam_N$ Internal nodes	$MARK(Jam_N) > 0$ and $MARK(Jamb_N) > 0$	NULL Function
$Errorclr_N$	$MARK(Cnt_N) \geq CNT$	$MARK(Cnt_N) = 0$;

Table 7: Activity Parameters Unique to the CSMA/DCR SAN Model

Activity	Rate	Probability	
		case 1	case 2
<i>Idledly_N</i>	determ(IDLE)	-	-
<i>Clr_Busy_N</i>	inst	-	-
<i>Delay_N</i>	determ(ST or IFS)	-	-

Table 8: Output Gate Parameter Unique to the CSMA/DCR SAN Model

Gate	Function
<i>Remove_N</i>	MARK(<i>Channel_N</i>) = 0;

Table 9: Input Gate Parameters Unique CSMA/DCR SAN Model

Gate	Enabling Predicate	Function
<i>DCR_Enable_N</i>	MARK(<i>DCR_N</i>) == 1 (or Zero) and (MARK(<i>Slot_Cnt</i>) == My_Station and MARK(<i>Serviced_N</i>) ≠ 1)	If MARK(<i>DCR_N</i>) == 0 <i>Serviced_N</i> = 0 Else If MARK(<i>Slot_Cnt_N</i>) == My_station <i>Service_N</i> = 1
<i>Add_Slot_N</i>	MARK(<i>Ch_Busy_N</i>) == 0 and MARK(<i>DCR_N</i>) == 1	If MARK(<i>Slot_Cnt</i>) ≥ MAX_STATIONS { <i>Slot_Cnt_N</i> = 1; <i>DCR_N</i> = 0; <i>Serviced_N</i> = 0 } else if MARK(<i>Channel_N</i>) == 0 { <i>Slot_Cnt_N</i> ++; <i>IFS_Req_N</i> = 0 } else if MARK(not <i>Channel_N</i>) { <i>Ch_Busy_N</i> = 1; <i>IFS_Req_N</i> = 1 }

Table 10: Activity Parameters for Transient Fault Model

Activity	Rate	Probability	
		case 1	case n
$Tf_arrival$	$\exp(\lambda)$	-	-
Tf_id_a	determ(ϵ)	$\frac{1}{n}$	$\frac{1}{n}$
RR_a	determ(ψ)	$\frac{1}{n-1}$	$\frac{1}{n-1}$
$Left_a$	determ(DIST)	-	-
$Right_a$	determ(DIST)	-	-
Tf_jam_n	determ(φ)	-	-
$Tf_duration_n$	determ(10 ms)	-	-
$Exc_S_R_n$	inst	-	-
$Exc_re_que_n$	determ(ξ)	-	-
$Exc_re_que_S$	inst	-	-
Tf_clr_a	inst	-	-

Table 11: Output Gate Parameters for Transient Fault Generator

Gate	Function
Tf_id_n	$MARK(Left) = n$ $MARK(Right) = n$ $MARK(Tf_station_n) = 1$ $MARK(Tf_clr_n) = 1$
RR_n	$MARK(RR_n) = n;$

Table 12: Input Gate Parameters for Transient Fault Generator

Gate	Enabling Predicate	Function
<i>Enable_tf_i</i>	MARK(<i>Enable_tf_p</i>) == 1	NULL Function
<i>Left_i</i>	MARK(<i>Left</i>) > 0	SWITCH(<i>Left</i>) case <i>n</i> : <i>Tf_station_n</i> = 1; <i>Left</i> - -;
<i>Right_i</i>	MARK(<i>Right</i>) > 0	if (<i>Right</i>) > MAX_STATIONS <i>Right</i> = 0 else SWITCH(<i>Right</i>) case <i>n</i> : <i>Tf_station_n</i> = 1; <i>Left</i> + +;
<i>Tf_jam_i_n</i>	MARK(<i>Channel_n</i>) == <i>n</i> and MARK(<i>Tf_station_n</i>) == 1	<i>Channel_n</i> = 1; Collision
<i>Determ_S_R_n</i>	MARK(<i>RR</i>) == <i>n</i> and MARK(<i>Channel_n</i>) > <i>n</i> MARK(<i>Tf_station_n</i>) == <i>n</i>	<i>RR</i> = 0; <i>R</i> = <i>n</i> ; <i>S</i> = <i>Channel</i> ; switch (<i>S</i>) case <i>n</i> : <i>C_D_type</i> = <i>D_type_n</i> ; BREAK;
<i>Re_que_S_R_n</i>	((MARK(<i>R_N</i>) == <i>n</i>) and ((MARK(<i>Channel_N</i>) == 0) or (MARK(<i>Channel_N</i>) == 1))	if (<i>Channel_n</i> == 1) <i>R</i> = 0; <i>C_D_type</i> = 0; else switch (<i>C_D_type</i>) case 1: switch (<i>R</i>) switch (<i>R</i>) case <i>i</i> : if (<i>q_t1_n</i> < 20) <i>q_t1_n</i> ++; <i>R</i> = 0; <i>Re_queue</i> = 1; case 2: switch (<i>R</i>) switch (<i>R</i>) case <i>i</i> : if (<i>q_t2_n</i> < 20) <i>q_t2_n</i> ++; <i>R</i> = 0; <i>Re_queue</i> = 1; case 3: (<i>R</i> = 0; <i>C_D_type</i> = 0;)
<i>Re_que_s</i>	MARK(<i>Re_queue</i>) > 0 and	switch (<i>C_D_type</i>) case 1: switch (<i>S</i>) switch (<i>S</i>) case <i>i</i> : if (<i>q_t1_n</i> < 20) <i>q_t1_n</i> ++; <i>S</i> = 0; <i>Re_queue</i> = 0; case 2: switch (<i>S</i>) switch (<i>S</i>) case <i>i</i> : if (<i>q_t2_n</i> < 20) <i>q_t2_n</i> ++; <i>S</i> = 0; <i>Re_queue</i> = 0; case 3: (<i>S</i> = 0; <i>C_D_type</i> = 0;)