PERFORMANCE MODELS OF TOKEN RING LOCAL AREA NETWORKS

Robert Berry and K. Mani Chandy
Department of Computer Sciences, University of Texas, Austin, TX 78712

1. Introduction

This paper presents a simple heuristic analytic algorithm for predicting the "response times" of messages in asymmetric token ring local area networks. A description of the token ring and the model is presented in section 2 the algorithm is described in section 3 and the empirical results in section 4. The analytic results were compared against a detailed simulation model and the results are extremely close over a wide range of models.

Local area networks (or LANS) offer a very attractive solution to the problem of connecting a large number of devices distributed over a small geographic area. They are an inexpensive readily expandable and highly flexible communications media. They are the backbone of the automated office - a significant component of the office of the future.

This importance of LANS in the future of applied computer science has resulted in a tremendous burst of interest in the study of their behaviour. There are already many different LAN architectures proposed and studied in the literature [Tropper 81] [Tannenbaum 81] [Babic 78] [Metcalfe 76] [Clark 78] One LAN architecture is significant for several reasons. This architecture is the *token ring* [Carsten 77]. It has attracted interest because of its simplicity fairness and efficiency. The interest it has generated has resulted in the proposal of several different versions. This paper concentrates on one of these versions - the *single token* token ring protocol as described in [Bux 81]. This particular version is attractive because of its overall simplicity and reliability. This paper presents an algorithm for predicting response times in a token ring with the single token protocol.

1.1. Related work

In [Bux 81] Bux presents an exact solution for predicting mean response times in a *symmetric* token ring. This work was derived from work of Konheim and Meister [Konheim 74]. However the assumption of identical traffic at all stations on a ring is considered too restrictive for practical application.

In [Carsten 77], Carsten et al investigated an asymmetric Newhall Loop with exhaustive service. They derived expressions for the mean and variance of scan times (the time it takes for a token to travel around the loop). This work was continued in [Carsten 78], where means and variances of response times were derived. The results obtained in that work do not apply here, as the rings studied in [Carsten 77] and [Carsten 78] are of the exhaustive service type. In an exhaustive service protocol a station, once given permission to transmit, may transmit all of its waiting messages. The token ring studied in this paper is believed to be a more practical protocol. It is much fairer than exhaustive service because a station may transmit only one message before passing permission along to another station.

In [Kuehn 79], Kuehn presents a very accurate algorithm for nonexhaustive service in cyclical service queueing systems with overhead. When applied to token rings, the algorithm works well for small rings with limited asymmetry of load and high overhead. The algorithm next presented in this paper is accurate for large rings with unlimited asymmetry of load and low overhead. We are interested in rings having these characteristics because we expect them to be most frequently enountered in practice.

* This work was supported by a grant from IBM.

2. The Token Ring

The operation of the token ring is now presented in more detail. The protocol is described here at the hardware level. Aspects of higher level protocols, such as packet assembly/deassembly, are not discussed.

A token ring consists of a communication line (a cable) configured as a closed loop. Data is transmitted in a single direction, bit serially, around this ring. There are N stations (indexed 0,...,N-1) on a token ring (see fig. 2-1). Message packets arrive at each station and are enqueued at the station in first come first served (FCFS) manner. Each packet contains a destination address (i.e. station index). This address indicates the index of the station to receive the packet, once successfully transmitted onto the ring.

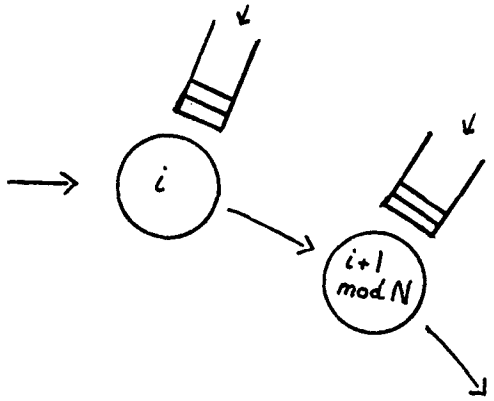

Figure 2-1: View of token ring data flow

A station with waiting packets must wait for permission to transmit onto the ring. This permission takes the form of a special sequence of bits, called the *token* (or, *free token*). All waiting stations constantly monitor the ring and watch for this special sequence. When such a station recognises the token sequence it alters it by changing the last bit of that sequence. This action removes the token from the ring, and creates a new bit sequence called a *connector token* (or, *busy token*). The station seizing the token in this way may now start to send out the message packet at the head of its queue.

The connector token sequence is now followed by the bits of the station's packet. Once the station has finished sending out all of the bits of its packet it must give up the token. Giving up the token means recreating the unique token bit sequence, and ceasing all subsequent transmission until it again recognises the token.

At this point in the description we now indicate that there are several slightly different versions of the protocol. Each of these versions differs in the way that a station, once finished with the token, decides to put it back on the ring. The token may be sent out immediately following the last bit of the packet. This is called *multiple token* operation in [Bux 81]. Since a station follows the last bit of its message with the token, it is possible for an adjacent station to immediately seize the token (i.e. change it to a connector token), and start transmitting its packet. Thus it is possible to have multiple tokens on the ring at one time. In fact, if packet sizes are small enough it is possible to have several complete messages on the ring at a time, with multiple token operation. Multiple token operation is interesting because it seems to offer maximal ring utilisation and minimal delays for seizing the token.

Another alternative, *single packet* operation requires that the last bit of the packet sent out by the station be received by that station (and removed from the ring) before that token can be recreated. This is the most conservative protocol, and only allows the bits of one message, and one token to be on the ring at any one time. For reliability and recovery this is the most attractive choice, as only one message is in jeopardy if the ring should fail. However, in a large ring it might take a long time for the last bit of the message to get back to that station. This creates a lot of "dead time" on the ring - time that stations with waiting packets could use.

Yet another alternative, *single token* operation is to release the token when the connector token is received back by the sending station. Single token operation is the intuitive compromise between the other two alternatives. It has the reliability and recovery advantages of single packet operation, yet approaches the ring utilisation of multiple token operation for large message packets. For these reasons, single token operation has received special interest, and so has been selected for study in this paper.

In addition to looking for the token, stations are also constantly monitoring the ring for packets addressed to them. When a station recognises its address in a packet it starts to copy the subsequently arriving bits into a local buffer. On recognition of the end of a packet sequence the station stops copying from the ring. The reception of a packet from the ring is non-destructive in that the packet is still on the ring after the destination has received it. It is the responsibility of the station sending a packet to purge that packet from the ring when the station starts to receive the packet itself. This behaviour facilitates broadcasting of messages - many stations can listen-in, not just one. It also increases the ring efficiency and reliability - the receiver of a packet can change a bit in that packet to indicate to the sender

that the packet was received; further, the sender can compare what it sent out against what it received back, and look for errors.

In order to recognise bit sequences on the ring every station buffers at least one bit of information. Thus, with only limited memory requirements, a station can recognise passing bit sequences (see fig 2-2).

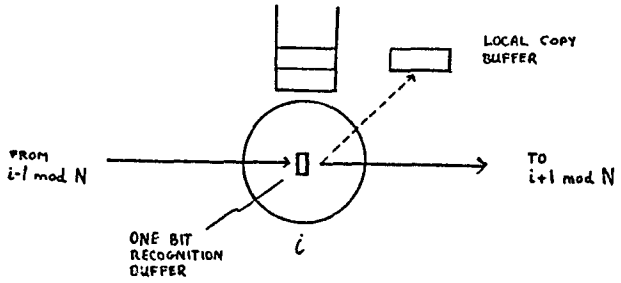

Figure 2-2: Token Ring Station with 1 Bit Buffer

For example, if the token is 8 bits long and has sequence 11111111, the token recognition logic need only count consecutive 1's. When 8 consecutive 1 bits have been counted by a station with a waiting packet, the station knows it has the token and may change the last bit of the token (while it still has it in its buffer) to a 0, thus giving the connector token the sequence 11111110.

3. The Algorithm

We now present the central contribution of this paper - an algorithm for predicting mean response times for stations on a token ring.

3.1. A Hypothesis - The Proportionate Error Hypothesis

This algorithm is based on an idea which was used in a different form in the LINEARIZER algorithm of Chandy and Neuse [Chandy 82]; we call the idea the *hypothesis of proportionate error* - we shall describe the general idea, and then the specifics of the algorithm.

Most realistic models of computer and communication systems are intractable because the state space is too large. Therefore, approximate techniques are used to estimate performance metrics; due to the approximate nature of the analysis, these estimates are erroneous. Our goal is to reduce the amount of error by devising an algorithm to correct (some of) the error. We hypothesise that when the same approximate algorithm is used to estimate several metrics, the error is consistently in the same direction for all metrics, i.e. the estimates are all too large, or all too small. We further hypothesise that the magnitude of error of an estimate is (roughly) proportionate to the true value of the metric being estimated. This hypothesis, called the proportionate error hypothesis, may not hold for all models, however, we have searched for algorithms to correct the error assuming that the hypothesis does hold.

The particular form that the algorithm takes depends on the problem at hand. The LINEARIZER algorithm takes one approach in correcting the error for analysing product-form queueing networks, and we shall take a very different approach in this paper for studying token rings. Both approaches, however, are based on the same hypothesis. In this paper we argue that if the proportionate error hypothesis holds true, then the *ratios* of the estimates should be almost exact. Therefore, we search for a normalising constant which, when multiplied by the (almost exact) ratio, yields an accurate estimate of the performance metric: this approach is discussed in detail next.

3.2. Notation and Assumptions

- Packets arrive at station i in a Poisson manner with mean arrival rate of λ_i.
- Packets at all stations have the same size distribution. This distribution is described by random variable X. The mean and second moment of packet length are denoted as E[X] and $E[X^2]$, respectively.
- The bandwidth of the ring is expressed as μ bits per second. The time to transmit a packet of length a bits is then a/μ seconds. The time spent by a station transmitting a packet is a random variable S, with mean $E[S] = E[X]/\mu$, and second moment $E[S^2] = E[X^2]/\mu^2$. The coefficient of variation of the transmitting time is denoted by k, and is determined as $k = \sigma_S/E[S]$, where σ_S is the standard deviation of S.
- We let the queue length of message packets at each station be described by random variable Q_i. The mean of this variable is $E[Q_i]$.
- Let the waiting time for a packet at station i be W_i. We shall derive the mean of W_i, $E[W_i]$.
- We define the utilisation of the ring by station i to be the product of average packet size and packet arrival rate at station i. We denote this as ρ_i. (So, $\rho_i = \lambda_i * E[X]/\mu$). The utilisation of the entire ring, ρ, is then $\rho = \Sigma_i \rho_i$, ignoring overhead.
- In this discussion we assume the ring is reliable and free of physical and logical

faults. (The single token protocol was chosen for study partially because of its reliability, but the issue of reliability is not of concern in the remainder of this paper).

- The overhead delay is denoted as d, and is expressed in seconds. This represents the physical ring propogation delay plus the data holding time (data buffering time) for a station (recall that a station must buffer at least 1 bit to allow for token recognition and capture). Rings are usually short (e.g. 1km) so propogation delay is minor. Station delays are small when compared with the mean packet transmission time ($E[X]/\mu$ seconds) because stations buffer very few bits (1 to 4 bits), and hence, station delays are of the order of $1/\mu$ to $4/\mu$ seconds.

3.3. Motivation for the Algorithm

3.3.1. The token ring is an M/G/1 system

If we consider the token ring just described as a single server the entire ring satisfies the assumptions of the Polloczek-Khintchine formula [Kleinrock 75]. This formula gives the average waiting time in a queueing system under the assumptions of a single server with Poisson job arrivals, where arriving jobs request service times from a general service time distribution; the service order is independent of service time request, there is no preemption of service and no overhead.

This type of queueing system is called an M/G/1 queue. The mean queue length for this system, E[Q], excluding the customer (if any) in service, is given by the Polloczek-Khintchine formula as

$$E[Q] = \rho^2 * (1 + k^2) / (2 * (1 - \rho)) \quad (1)$$

where ρ is the server utilisation, and k is the coefficient of variation of service time.

We can apply Eqn. 1 to the token ring environment and derive the overall mean ring queue length, with $\lambda = \Sigma_i \lambda_i$. This tells us the total number of message packets, on average, that are waiting in the ring system, excluding the packet being transmitted. From Little's Law [Little 61],

$$E[W] = E[Q] / \lambda \quad (2)$$

were E[W] is the overall average wait time (excluding service) for all packets arriving at the ring, over all stations. Our goal is to determine $E[W_i]$, for all i, i.e. the mean waiting time for packets arriving at each station i. The ring is not symmetric, and the mean waiting time at each station may be different from the overall mean waiting time. Our problem is to derive individual station statistics from the overall ring statistics.

To derive approximate ring statistics from station statistics we shall partition the overall mean ring queue length E[Q] into the individual station components. Our approach is to derive approximate mean queue lengths at individual stations q_i' and divide E[Q] in proportion to these values so that

$$E[Q_i] = (q_i' / \Sigma_j q_j') * E[Q] \quad (3)$$

3.3.2. Determining individual station queue lengths

A packet P arriving at random at station j can expect certain delays before being transmitted onto the ring from j to its destination. P's waiting time is divided into two components (see fig 3-1) for convenience:

1. waiting for the token to reach station j from its location on the the ring at the instant at which P arrives.
2. waiting for the packets ahead of P in station j's queue to be sent, given that the token is now at station j.

For instance, consider a packet P arriving at random at station 10 (say). Suppose that when P arrives the token is at station 1, and there are already 2 packets ahead of P in station 10's queue. We partition the waiting time for P into a) the time required for the token to get to station 10 from its current location at

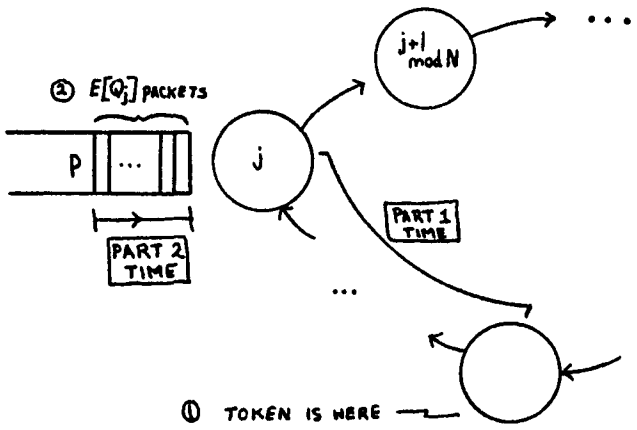

Figure 3-1: Waiting time components for a random arrival in a token ring

station 1, and b) the time remaining for the token to return to station 10 after making 2 more *complete* revolutions. (On the revolution in progress when P arrived, the packet at the head of station 10's queue will be transmitted when station 10 gets the token. On the next revolution the packet immediately ahead of packet P will be transmitted. Then at the end of the second revolution, packet P will be transmitted.) Our goal is to divide the number of revolutions a token must make

before P is transmitted into an initial partial revolution, followed by a number (0 or more) of whole revolutions.

These component delays are now discussed from the point of view of a randomly arrived packet at station j.

Waiting for the token to reach station j - the initial partial revolution

When a random packet arrives at station j, the token could be at any of the N stations (including j). The time for the token to reach station j will depend on

- the time for the present user of the token (say, station i) to complete transmission and release the token.
- the probability that any station between i and j will seize the token while on its way to j.

These considerations result in the definition of a random variable T_{ij} which describes the time for the token to reach station j given that it is held by station i. We are interested in the value of $E[T_{ij}]$, and compute it as follows (see explanation below):

$$E[T_{ij}] = E[S^2]/2*E[S] + d \quad \text{, if } i=\text{pred}(j) \quad (4)$$

$$E[T_{i\text{pred}(j)}] + \rho_{\text{pred}(j)}*E[S] + d \quad \text{, if } i\neq\text{pred}(j)$$

where pred(j) is the id of the station immediately preceding j on the ring.

Explanation for Eqn. 4, case 1: i=pred(j)

That is, if station i is immediately adjacent to j on the ring, then as soon as i's transmission completes it will give the token straight to j. $(E[S^2]/2*E[S])$ is called the mean *residual life* of the service time and represents the average remaining service time after a random "glance" at an active server. For more information about the theory of residual life, consult [Kleinrock 75]).

Explanation for Eqn. 4, case 2: i≠pred(j)

The time for the token to travel from station i to station j is the sum of the time taken for the token to travel from station i to station pred(j), and the time taken by station pred(j) to pass the token to station j. The probability that station pred(j) will have a packet for the ring is assumed to be $\rho_{\text{pred}(j)}$ - the utilisation of that station. If station pred(j) does have a packet, then this will delay the time for j to get the token by E[S] - a complete mean packet service time.

Using P_K for the probability that station k has a packet for the ring is a simplification. An early version of the algorithm used $1 - e^{-\lambda_K(T_{KK}-E[S^2]/2*E[S])}$ for this probability. This value is the probability of at least one arrival in the time for the token to reach status k, after k releases it. We found, however, that this made little difference to the results, and in fact that using p_K provided slightly better results. We suspect that proportional error contributes to the accuracy of the more simple approximation.

It is possible for the token to be in transit between stations when P arrives. This possibility was considered in our initial models, but was later discarded because it did not make the model more accurate for the parameter ranges of interest.

Now that we know the time for the token to reach j, given that it is at i, we can derive the mean time for the token to reach station j, regardless of where the token is. We describe this time with random variable K_j. The token will be at station i with probability ρ_i (station i's utilisation), so we may compute $E[K_j]$ as:

$$E[K_j] = \Sigma_i \rho_i*E[T_{ij}] \quad (5)$$

Waiting for packets ahead of P to leave - the succeeding whole revolutions

When the token has arrived at station j (the initial partial revolution is over), packet P must still wait until all packets ahead of it in j's queue have been sent. In this token ring protocol, a station may only send one packet before passing the token, so after each of the packets in j's queue is sent, the token will have to be released.

So, when one of the packets in j's queue is sent, and the token released, the remaining stations on the ring will get an opportunity to send a packet. Station j will send a packet, then station j+1 mod N may send one, j+2 mod N may send one, and so on, until station j gets the token back again. Then j sends another packet and the other stations get a similar chance. This behaviour continues until all of the packets originally ahead of packet P in j's queue are sent. This packet's wait is now over, and so it may begin transmission.

This situation is best illustrated by an example. Consult figure 3-2. Station 1 has just received packet P. The token has reached station 1, and now packet P finds itself waiting at station 1 for the three packets ahead to be sent. After packet A is sent, station 2 will send packet D, followed by station 3 sending packet F. The sequence continues, G-B-E-H-C-I, and then station 1 can send packet P. So, from P's point of view, it had to wait for 9 packets to be sent before it could be sent. Had any arrivals into stations 2 or 3 occured *before* packet C could have been sent this would have lengthened the sequence, and so the delay to packet P. Notice that packet J, and all subsequent arrivals at station 4 will not affect in any way the time for packet P to get onto the ring.

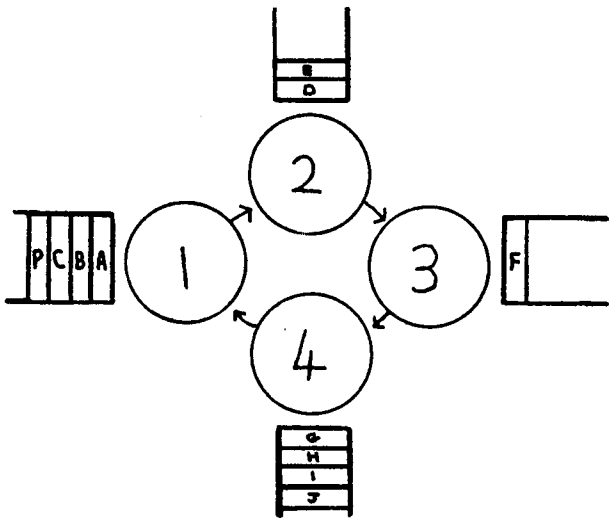

Figure 3-2: Example of packet waiting

We will call the time spent by a packet waiting for complete token revolutions (due to packets it finds ahead of it station j's queue), *work*. Let this work time spent by a packet be described by a random variable Z_j, with mean $E[Z_j]$. When a random packet, P, arrives at station j, each of the stations m (m ϵ 1..N) will contain $E[Q_m]$ packets in their queues, on the average. By the time that this packet at station j starts transmission (after $E[W_j]$) there will be, on average, $\lambda_m * E[W_j]$ new arrivals at each queue m. The $E[Q_j]$ packets at station j will have to compete for the ring with $E[Q_m]+\lambda_m * E[W_j]$ other packets at each station m. If j's queue is emptied first, it doesn't matter how big the other queues are because the waiting packet at j will now be at the head of the queue. Thus the maximum number of packets that any station i can send before P enters the ring is $E[Q_j]$. Hence we estimate the number of packets that enter at station i before P enters the ring as $\min(E[Q_j], E[Q_m]+\lambda_\mu * E[W_j])$. These considerations result in the following definition of expected work:

$$E[Z_j] = E[S] * (E[Q_j] + \Sigma_{m \neq j} \text{MIN}(E[Q_j], E[Q_m]+\lambda_m * E[W_j])) \quad (6)$$

The total mean wait time

The goal of the algorithm is to determine $E[W_j]$, the mean wait time at each station j. The last two sections discussed the determination of each of the two components of total wait time. The overall wait time for a packet is the sum of the two components described above. That is, the expected wait time for a packet at station j, is estimated as:

$$w'_j = E[K_j] + E[Z_j] \quad (7)$$

The result of these computations is a set of approximate mean wait times and queue lengths. We know, however, that the total queue length must sum to E[Q], and so use Eqn. 3 to arrive at the final estimates for wait times and queue lengths.

We now present the algorithm in stepwise form.

3.4. The Algorithm

The algorithm is iterative. New values for the queue lengths of message packets at individual stations, $E[Q_i]$, are computed in each iteration from the old values, $LAST\_E[Q_i]$, taken from the last iteration. At the end of each iteration, these two values are compared to check for iteration convergence.

Inputs: Message packet distribution moments - $E[X]$, $E[X^2]$
Ring bandwidth - μ

Number of stations - N

The arrival rates, for all i - λ_i

Initialisation: Initialise for all i

$E[W_i] = 0.0$
$E[Q_i] = 0.0$
$LAST\_E[Q_i] = 0.0$
$w_i' = 0.0$
$q_i' = 0.0$

Compute utilisations

$\rho_i = \lambda_i * E[S]$
$\rho = \Sigma_i \rho_i$

Compute coefficient of variation of service

$k = \sigma_S / E[S]$

Step 1: Compute E[Q], the overall queue length using Eqn. 1

Step 2: Compute $E[T_{ij}]$, for all i and j using Eqn. 4

Step 3: Compute $E[K_i]$, for all i using Eqn. 5

Step 4: Compute work, $E[Z_i]$, for all i using Eqn. 6

Step 5: Compute w_i', the approximate wait times, for all i using Eqn. 7

and q_i', the approximate queue lengths, using Little's Law.

Step 6: Save former values of $E[Q_i]$ in $LAST\_E[Q_i]$, for all i.

Compute $E[Q_i]$, the predicted queue length using Eqn. 3

and $E[W_i]$, the predicted wait times, using Eqn. 2

Step 7: Compare $E[Q_i]$ and $LAST\_E[Q_i]$. If the convergence criterion is met, terminate. If the convergence criteria is not met, go to step 4.

Convergence Criterion

The convergence criterion used is based on queue length. At the end of one iteration new values of $E[Q_i]$ are compared with old values ($LAST\_E[Q_i]$). Define MAXDIF to be the largest relative difference between the corresponding values of $E[Q_i]$ and $LAST\_E[Q_i]$. MAXDIF is compared with a small number (e.g. .0001). If MAXDIF is larger then .0001, we must continue. If MAXDIF is smaller than .0001, we may terminate the iteration.

4. Results, Validation and Discussion

A token ring was not available for instrumentation, so testing of the algorithm was done against simulation. A very efficient simulation model of the token was constructed in Fortran. This model included confidence interval estimates to facilitate simulation convergence determination, as well as to aid in the evaluation of the results of the algorithm.

The experiments were sequenced in increasing order of variability of model parameters. That is, initial experiments considered only symmetric rings. From these, a large range of models were considered where the packet size was identically distributed over all stations, but the arrival rates were different. We now present a discussion of the results.

4.1. Symmetric Rings

The Konheim Meister [Konheim 74] result reported in [Bux 81] indicates that the expected wait time at a node in a symmetric exhaustive service token ring is

$$W = \rho*E[S^2] / (1-\rho)*2*E[S] + E[S] + N*d*(1-\rho/N) / 2*(1-\rho) + N*d / 2 \quad (8)$$

Numerous simulation experiments were conducted with symmetric rings of various sizes. The results obtained from the simulation experiments were all in agreement with Eqn. 8. The analytic algorithm presented in this paper gives results identical with Eqn. 8, except for the last two terms, which we consider insignificant. Notice that the first term of Eqn. 8 is exactly the Polloczek-Khintchine formula (see Eqn. 1).

The last two terms in Eqn. 8 are, under the assumptions considered in this paper, very small. For instance, on a 100 station, 4 Mbps ring, where each station imparts a 1 bit delay to the ring traffic, even if the ring utilisation is 90% the last two terms are less than .0002 seconds. Of far more significance are the first two terms.

4.2. Asymmetric Rings

We conducted experiments with rings of various sizes and degrees of asymmetry. The state space of possible token ring configurations and arriving packet characteristics is enormous, so we attempted to cover a manageable, but useful portion of it.

Ring sizes were partitioned into three groups - small (up to 5 stations), medium - (6 to 11 stations), and large (12 to 21 stations). Larger ring configurations are certainly possible, but we expect that because of performance, reliability and cost, very large ring systems will be partitioned into rings of rings, and so on.

Within each of the three ring sizes different asymmtric topologies were considered. Type I asymmetry considered rings where one of the stations had an arrival rate greater than all the other stations. Each of the other stations had the same arrival rate. Type II asymmetry considered rings where there were stations having larger arriva rates than the rest of the stations. These two stations each had the same arrival rate and were adjacent on the ring. Type III asymmetry was the same as Type II except the two stations with high arrival rates were on opposite sides of the ring. With rings exhibiting Type I, II and III asymmetry we were also interested in studying the effect of position on the ring on individual station response times.

Two packet size distributions were considered for the tests. A few of the tests used exponentially distributed packet sizes. Most of the experiments considered only constant packet size distributions because it was more realistic.

A further parameter of concern was the ring utilisation. Earlier studies [Tropper 81]resulted in models that were accurate only for low utilisations. We felt that , in order to be useful, our algorithm should work for high utilisations as well. Our experiments allowed the ring utilisation to range over 20%-90%.

A sample of the results of the study is shown in figure 4-1.

4.3. Accuracy

The validation metric used in the study was the relative deviation of the wait time predicted by the algorithm from that produced by the simulation. The predicted mean response time for station i is the sum of the mean wait time at station i ($E[W_i]$) and the mean packet transmit time (E[S]). In all configurations studied, including

- overall utilisations ranging from 20%-90%
- asymmetry Types I, II, and III
- rings with 5, 11, and 21 stations
- constant message packet size distribution

the algorithm has consitently predicted individual station response times within 10% of times produced by the detailed simulation model. Over 270 such individual station response times were so predicted during this study. It is worthy of note that the algorithm is *consistently* accurate for the entire range of utilisations, from low to high.

Medium Ring

N = 11 stations
μ = 1 Mbps
E[X] = 10000 bits; $E[X^2]$ = 100000000 $bits^2$
(constant distribution)
Type I asymmetry: $\lambda_{1,1\neq1}$ = 1/s;
λ_1 = 10/s, 30/s and 80/s
Note: 90% confidence intervals are shown where wide enough to be visible on the graph

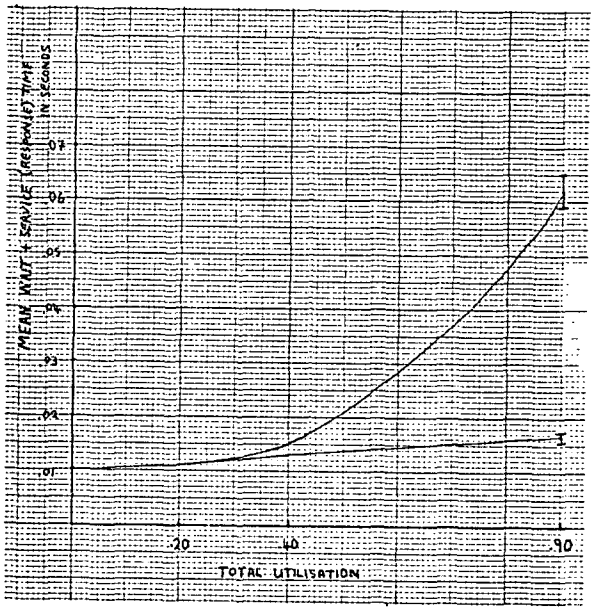

Figure 4-1: Example plot of respone time

5. Conclusion

We have presented a heuristic based on the hypothesis of proportionate error to analyse token ring local area networks. The heuristic is rapid and accurate. We were fortunate in that the M/G/1 formula gave us an accurate estimate of the overall behaviour of the system; the M/G/1 estimate of overall queue length multiplied with accurate estimates (based on the proportional error hypothesis) of the *ratios* of individual station queue lengths to the overall queue lengths, gave accurate estimates of performance metrics. Applying the proportionate error hypothesis to other problems may not be equally straight-forward.

The arguments used in developing this heuristic apply to rings using multiple token and single packet protocols. However, validation has been carried out only with the single token protocol. We plan to carry out validation studies with the other protocols.

Models of local area networks which incorporate higher-level station-to-station protocols, such as HDLC, have not appeared in the literature. We feel that the proportionate error hypothesis applies to such models as well.

6. Acknowledgement

We would like to thank Fred May of IBM Austin, for his advice and support. We also want to thank Bill McCallum, Rick Gimarc, Charlie Sauer and Jim Markov of IBM for their help.

References

[Babic 78] Babic, Gojko A. *Performance Analysis of the Distributed Loop Computer Network.* PhD thesis, Ohio State University, 1978.

[Bux 81] Bux, Werner. Local-Area Subnetworks: A Performance Comparison. *IEEE Trans. on Comm* (10):1465-1473, October, 1981.

[Carsten 77] Carsten, Ralph T., et. al. A Simplified Analysis of Scan Times In an Asymmetrical Newhall Loop with Exhaustive Service. *IEEE Trans. on Comm* (9):951-957, September, 1977.

[Carsten 78] Carsten, Ralph T. and Posner, M.J. Simplified Statistical Models of Single and Multiple Newhall Loops. *Proceedings of the National Telecomm. Conf.* :44.5.1-44.5.7, 1978.

[Chandy 82] Chandy, K.M. and Neuse, D.M. Linearizer: A Heuristic Algorithm for Queueing Network Models of Computing Systems. *CACM* 25(2):126-134, February, 1982.

[Clark 78] Clark, David D., et. al.
An Introduction to Local Area Networks.
Proceedings of the IEEE 66(11):1497-1517, November, 1978.

[Kleinrock 75] Kleinrock, Leonard.
Queueing Systems, Volume 1: Theory.
Wiley-Interscience, New York, New York, 1975.

[Konheim 74] Konheim, Alan G. and Meister, Bernd.
Waiting Lines and Times in a system with Polling.
JACM 21(3):470-490, July, 1974.

[Kuehn 79] Kuehn, P.J.
Multiqueue Systems with Nonexhaustive Cyclic Service.
Bell Systems Technical Journal 58(3):671-698, March, 1979.

[Little 61] Little, J. D. C.
A Proof of the Queueing Formula $L=\lambda W$.
Operations Research 9:383-387, 1961.

[Metcalfe 76] Metcalfe, Robert M. and Boggs, David R.
Ethernet: Distributed Packet Switching for Local Computer Networks.
CACM 19(7):395-404, July, 1976.

[Tannenbaum 81] Tannenbaum, Andrew S.
Computer Networks.
Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[Tropper 81] Tropper, Carl.
Local Computer Network Technologies.
Academic Press, New York, New York, 1981.