OPTIMAL DELAYS FOR RETRANSMISSION IN MULTI-ACCESS COMMUNICATION SYSTEMS

Frits C. SCHOUTE

Philips Telecommunicatie Industrie, Hilversum, Netherlands
University of California, Berkeley, USA.

ABSTRACT

In the context of a small unslotted carrier sense multi-access system we want to answer the following questions: a) Does it make a difference whether retransmission delays are drawn from an uniform or an exponential distribution? b) Is there a simple way to implement exponentially distributed delays? and c) Should one double the mean delay on subsequent retransmissions of the same packet (binary backoff)? We find the following:

a) A very simple two station simulation model shows an advantage for exponentially distributed delays over uniformly distributed delays.

b) In fixed point binary notation, the bits of an exponential random variable are independent binary random variables. This leads to an algorithm that is simple to implement on a microprocessor without logarithm capabilities.

c) An analytical model which gives an adequate approximation of a multi-access system with states (a next packet needs to be rescheduled) can be used to optimize the mean delay for each state.

1. INTRODUCTION

Ever since the implementation of the Aloha system at the University of Hawaii [1], similar forms of multi-access communication systems are enjoying great interest in academia for many of the challenging theoretical and practical questions and in practice for the flexibility of implementation. One of the well known earlier publications that addresses the issue of control of retransmissions is [2]. It is for me impossible to give even a rudimentary overview of what has been published on this subject since. Rather, I would like to address some related questions, which have been raised by implementors and for which, to my knowledge, the answers are still unknown.

The multi-access systems addressed here are small-scale and simple. Small-scale means e.g. a mobile radio system with a few tens of users, an office system with, say, ten stations or maybe a home system with, say, five stations. Simple means that one has very limited amounts of hardware and software. Typically such a system is a non-slotted carrier sense multi-access system. The questions that we want to answer for such a system are: a) Does it make difference whether retransmission delays are drawn from an uniform or an exponential distribution? b) Is there a simple way to implement exponentially distributed delays? and c) Should one double the mean delay on subsequent retransmissions of the same packet (binary backoff)?

Correspondingly the paper has the following sections:

Section 2 introduces a simple model of two stations or users that share a multi-access channel. Transmissions are to be rescheduled after a random delay when packets of the two users collide or when one of the users senses the channel busy. For a given set of parameter-

we compare, using discrete event simulation, the throughput of the channel for uniformly distributed and exponentially distributed retransmission delays.

In section 3 we show that in fixed point binary notation, the bits of a exponential random variable can be generated as the outcomes of independent Bernoulli experiments and for the less significant bits the success-probability is \( \frac{1}{2} \). This observation is used for an algorithm for exponential variates that can easily be implemented on a microprocessor.

In section 4 we consider a multi-access system where each station, when rescheduling transmission of a packet, enters a new state. Each state can have its own mean retransmission delay. The complexity of optimization of mean retransmission delays by simulation becomes prohibitive. We shall derive an approximating analytical model for which one can use standard optimization routines.

2. UNIFORM vs. EXPONENTIAL DISTRIBUTION

In a non-slotted carrier sense multi-access communication system collisions occur if two or more stations have scheduled their start of transmission less than \( \Delta \) apart. We assume \( \Delta \) to be a constant that incorporates processing delay between sensing the channel idle and start of transmission, as well as propagation delay. If a collision occurs then the start of transmission is rescheduled with a random retransmission delay. Rescheduling also takes place if the channel is sensed busy before an intended start of transmission. To investigate a possible difference in performance between using uniformly distributed retransmission delays or exponentially distributed retransmission delays, we consider an idealized model of a two station multi-access channel. The performance measure to be maximized, \( \eta \), is the utilization of the channel by successful transmissions.

To specify the model further, let us assume that the transmission time of one packet is constant and equal to one unit of time, that each successful transmission is followed by an exponentially distributed idle time \( V \), and that there are only 2 stations. When a transmission collides or when the channel is sensed busy before the intended transmission takes place, it will be rescheduled \( \Delta \) after the original (intended) start of transmission. Here \( D \) is the random retransmission delay and \( \Delta \) is added because packet transmission may be going on already this long before a collision occurs. At any point in time we let \( T_i \) give the time of the first upcomming start of transmission and \( T_j \) the time at which the other user intends to start transmission. Note that the indices do not correspond to the number of the station but to the order of the scheduled transmissions. We use ‘the symbol \( T_k \) and \( T_l \) to denote the next scheduled transmission of the station that was first and second, respectively. Depending on the relative distance of \( T_i \) and \( T_j \), we can distinguish the following three cases:

4.1A-2-1
In the last two cases (busy and succ) the first station has a successful transmission, and we collect one point. The utilization of the channel, \( \eta \), is the number of points collected per unit of time.

For given \( \Delta \) and mean idle time \( E[V] \), \( \eta \) depends on the probability distribution of the retransmission delay. We analyzed this dependence with a small simulation program which, each cycle, updates the values of \( T_1 \) and \( T_2 \) according to

\[
T_1 := \min(77, 78); \quad T_2 := \max(77, 78)
\]

The symbol \( "\lt=" \) is to be read as "gets the value". The values of \( T_1 \) and \( T_2 \) for the next cycle are again determined depending on the case call, busy or succ from the list above.

Figure 1 summarizes the the difference in utilization of the channel for uniform and exponential distribution of retransmission delays as function of mean retransmission delay. For those graphs we have taken \( \Delta=0.2 \) and \( E[V]=1 \). The maximum attainable utilization of the channel is larger with exponentially distributed retransmission delays than with uniformly distributed delays.

![Graph of Channel Utilization as Function of Retransmission Delay](image)

3. GENERATING EXPONENTIAL RANDOM VARIABLES

Suppose one has available a random number generator that generates \( U \), uniformly distributed in the interval \((0,1)\). Then, an easy way to generate an exponential random variable, \( X \), is to compute \( X = -\ln (U) \) [3]. However when implementing a multi-access algorithm on a microprocessor, the small advantage of the exponential distribution as outlined in the previous section may not be worth the expense of adding code to compute the natural logarithm. Therefore we want to show a way to generate exponential variables that is well suited efficient implementation on a microprocessor.

Consider an exponential random variable \( X \) with mean 1 (to change the mean, just multiply \( X \) with the desired mean). In binary fixed point notation \( X \) can be written as

\[
X = \cdots \times b_6 b_5 b_4 b_3 b_2 b_1\]

where bit \( b_i \) indicates whether the term \( 2^i \) is in \( X \). Define \( p_i = P[b_i=1] \). In appendix A we give an expression for \( p_i \) and we show that \( b_0 \) and \( b_j \) (\( i \neq j \)) are independent random variables. Before we compile a table of bit probabilities a choice has to be made as to which accuracy these probabilities will be specified. For our application 16 bit accuracy is more than sufficient. The probability that bit 4, or a more significant bit, is 1 is negligible \((<2^{-40})\). Therefore we start the table with \( p_4 \) and specify this probability with 16 bit accuracy. To match this accuracy for less significant bits we specify \( p_j \) with 13+j bit accuracy \((j=12, \ldots, 4)\).

![Probability Table](image)

Table 1. Probability \( p_i \) that bit \( i \) of an exponential random variable (with mean 1) is 1. The right hand column gives \( p_i \) in binary fixed point notation with 13+j bit accuracy. Trailing zeros are omitted.

Suppose that we have a source of random bits, e.g. from a Tausworthe generator [4], in the form of a coded function with the name RanBit. Upon each call, the function RanBit returns a "1" with probability \( \frac{1}{2} \) or else a "0". As an example of how a bit of \( X \) is generated, we give below the flow diagram of the code to generate \( b_3 \). Each line corresponds to one digit of \( p_{-3} \) binary.

![Flow Diagram](image)

To understand how this works note that \( b_{-3} \) will only get the value "1" if the assignment of line 2 or 3 or 4 or 5 is executed. The probability that this happens is \( \frac{1}{8} \), which is the 13-3 bit approximation of \( p_{-3} \). For \( b_3 \) downto \( b_{-3} \) we have similar flow diagrams, and for the low order bits we only need to copy random bits, i.e. \( b_i = \text{RanBit}(i=-6, \ldots, -12) \). These flow diagrams are most suitable for implementation in
machine code. What makes the code efficient is its linearity (no loops) and the fact the 105 bits of the column ‘p, binary’ are directly expressed in the code. Although it seems that for the high order bits there are many calls to RanBit, it can be seen from the example flow diagram that as soon as RanBit returns a ‘1’, no more calls to RanBit are made for that bit of X. So the average number of calls to RanBit is less than 2 for each of the 16 bits of X.

4. OPTIMAL MEAN DELAY FOR EACH STATE
In some systems, like Ethernet, rescheduling of a transmission puts the station into a next state. Here we shall address the question of the optimal mean delay for each state. The model is more elaborate than the model of section 2, because we want to consider the case of m > 2 states and moreover each station can be in one of the m states. So, even when ignoring the fact that the holding time of each state is not exactly exponential, we still have that the size of the state space is of the order m^m.

Given the complex constraint that only one station at a time can be successfully transmitting, there is not much hope for an exact analytical model. A simulation model that allows for different mean retransmission delays for each state, makes optimization by simulation, as we did in section 2, not very feasible. The approach that we want to take in this section is to derive an analytical model that is simplified in the sense that we ignore most necessary conditions for each step in the derivation. We verify the analytical model by comparing it with the outcome of a more realistic simulation model. The analytical model can then be used for optimization of mean retransmission delays.

For future reference we list below the most important variables used in the analytical model.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>number of stations</td>
</tr>
<tr>
<td>m</td>
<td>number of states</td>
</tr>
<tr>
<td>( \Delta )</td>
<td>min. separation of start of trans. for succ</td>
</tr>
<tr>
<td>i</td>
<td>transmission time of packet</td>
</tr>
<tr>
<td>d_i</td>
<td>mean retransmission delay state i</td>
</tr>
<tr>
<td>h_i</td>
<td>= d_i + \Delta (i &gt; 0) mean time spent in state i</td>
</tr>
<tr>
<td>h_0</td>
<td>= v + 1 mean time spent in state 0</td>
</tr>
<tr>
<td>f_i</td>
<td>probability station is in state i</td>
</tr>
<tr>
<td>p_i</td>
<td>succ probability from state i</td>
</tr>
</tbody>
</table>

In the state diagram of figure 2, each state corresponds to a possible state of a station. State 0 comprises the (successful) transmission time and the exponentially distributed idle time, \( V \), of a station.

![State Transition Diagram](image)

Fig. 2 State transition diagram for a station.

Let \( h_i \) be the mean time that a station spends in state i and \( f_i \) be the probability that a station is in state i at an arbitrary point in time, then we get from equating the probabilities of an upward transition and a downward transition across an imaginary boundary between state i and i + 1:

\[
(1-p_i) f_i = \sum_{j=i+1}^{m} p_j f_j h_j
\]

If we assume the success probabilities, \( p_i \), are given, then \( f_i \) i = 0, \ldots, m can be determined from the normalizing condition

\[
\sum_{i=0}^{m} f_i = 1
\]

The next step is to derive the success probabilities \( p_i \) assuming that \( f_i \) is given. At an arbitrary point in time a planned transmission will not be rescheduled if the channel is idle and if there is no interference from another transmission:

\[
P_{\text{succ}} = P_{\text{no interference}} P_{\text{channel idle}}
\]

The rate of transmission of a station in state i (i.e. the probability per unit of time that a start of transmission is planned) is 1/\( h_i \). Therefore the weighted rate is:

\[
f = \sum_{i=0}^{m} f_i h_i
\]

A station will have no interference if the n-1 other stations have not planned their start of transmission within \( \Delta \) of its start of transmission, hence:

\[
P_{\text{no interference}} = e^{-2\Delta (n-1) f}
\]

A station spends on average a fraction 1/(\( v + 1 \)) of state 0 in transmitting. Thus the probability that no other station is transmitting at an arbitrary point in time is:

\[
P_{\text{channel idle}} = 1 - \frac{n-1}{v+1} f
\]

An unsuccessful attempt that is repeated again after a very short time is likely to be unsuccessful again. We found that the 'steady state' value \( P_{\text{succ}} \) was approached exponentially with time constant \( \Delta/2 \). This gives us:

\[
p_i = P_{\text{succ}} [(1-e^{-2\Delta f/n})]/h_i
\]

The iterative procedure uses equations (2)-(3) to compute \( f_i \), then with \( f_i \) given, it uses (4)-(8) to compute \( p_i \). In the next iteration we start again at (2) with the new values of \( p_i \). For those values of \( d_i \) that correspond to a stable system, we experienced rapid convergence of the iterative procedure to a solution to the set of equations (2)-(8).

The performance that we want to maximize is the number of successful transmissions per unit of time. Each successful transmission corresponds to one visit to state 0 followed by zero or more visits to higher states; we call this a cycle. Hence maximizing successful transmissions corresponds to maximizing the number of cycles per unit of time, which corresponds to minimizing the mean cycle time, \( t_{cyc} \). Since each cycle has one visit to state 0, it is not hard to see that \( f_0 t_{cyc} = h_0 \). or, \( t_{cyc} = h_0 / f_0 \)

Two cases of retransmission delays in state i are of special interest to us: (i) \( d_i \) = 24, corresponding to 'binary backoff' which is done in the widely implemented Ethernet and (ii) \( d_i \) = constant which turned out to minimize \( t_{cyc} \). For a system with \( n = 5 \) stations and \( m = 8 \) states and a mean idle time \( v = 8 \) packet lengths we simulated and computed the quantities of (2)-(8). The input for \( d_i \) was (i): \( d_i = 0.1, \ldots, d_i = 12.8 \) (8): \( d_i = 0.5 \) (i=1, \ldots, 8). In figure 3 we show the computed values and confidence intervals of the simulated value of \( t_{cyc} \), for 'binary backoff' and constant retransmission delay.

The approximate model is not exact in predicting the outcome of the simulation. However, varying the values for the retransmission delays, we found that the changes in \( t_{cyc} \) and other quantities of the analytical model paralleled the changes observed in the simulation model. Therefore the analytical model is very useful when searching for optimal retransmission delays. The search was done by numerical optimization. The outcome: to have the same retransmission delay for all states, was not what we had initially expected, but not so surprising afterwards, if one considers the equal roles that state 1 through 8 play in the model.
Fig. 3 Mean time between successful transmissions of the same station, $t_{get}$, for 'binary backoff' and constant retransmission delays.

5. CONCLUSIONS.

Investigation of questions on retransmission delays that were posed for a small unslotted carrier sense multi-access communication system led to the following results.

a) A higher throughput can be obtained with retransmission delays from the exponential distribution than with uniformly distributed delays.

b) Constructing an exponential random variable, from a source of random bits, can be done very well on a microprocessor without logarithm capabilities. From the point of view of programming such a routine in machine language, the code is straightforward and efficient.

c) From the point of view of maximizing throughput in a system with a given (small) number of stations and given equal mean idle time (think time) for each user, constant retransmission delays is better than binary backoff. However, we did not address aspects of stability of the system. Instability would show up in the analytical model as oscillations in the iterative procedure of section 4.1. Further research is still needed to know which set of retransmission delays makes the system robust.

REFERENCES


APPENDIX A

Let $X$ be an exponential random variable with mean 1. In fixed point binary notation it is written as

$$X = \ldots b_2 b_1 b_0 . b_{-1} b_{-2} b_{-3} \ldots$$

Bit $i$ has the value "1" if $X$ is in one of the intervals $[k2^i+1, (k+1)2^i)$ for $k=0,1,2, \ldots$, hence by integrating the exponential density over those intervals we get

$$P(b_i=1) = \sum_{k=0}^{\infty} \int_{k2^i+1}^{(k+1)2^i} e^{-x} dx$$

Using the notation $a_i = 2^i$ (note that $a_{i+1} = a_i^2$) this can be worked out as

$$P_i = \sum_{k=0}^{\infty} (a_i^{k+1} - a_i^k) \frac{a_i - a_i+1}{1 - a_i+1} = \frac{a_i}{1+a_i}$$

The next thing we want to show is that $b_i$ is statistically independent of $b_{i+1}, b_{i+2}, \ldots$. When $b_{i+1}, b_{i+2}, \ldots$ are given this means that we know in which interval of size $2^{i+1}$ the random variable $X$ is. Let this be the interval $[k2^i+1, (k+1)2^i)$. Now the conditional probability

$$Pr[b_{i+1} = 1 | b_i + 1, b_{i+2}, \ldots] = \frac{\int_{k2^i+1}^{(k+1)2^i} e^{-x} dx}{\int_{k2^i+1}^{(k+1)2^i} e^{-x} dx} = \frac{a_i - a_i+1}{a_i^{k+1} \cdot a_i^k}$$

turns out to be equal to the unconditional probability from which we may conclude independence of the binary random variables $b_i$ and $b_j$ for $i \neq j$. 

APPENDIX A-2-4