A NOTE ON MULTIPLE-ACCESS PROTOCOLS

by

Zvi Rosberg

Technical Report #229

December 1981
A NOTE ON MULTIPLE-ACCESS PROTOCOLS

BY

Zvi Rosberg

ABSTRACT

A system of N transmission stations, transmitting messages via a common channel, is considered. A generalization of Gallager's multiple-access protocol is defined and analyzed by a very simple method. The maximal throughputs for different N are determined with the aid of the continuity principle in a Markov decision process; for infinite N the throughput for the optimal FIFS (Humblet-Mosely) protocol is readily obtainable. The protocols are adapted to slotted and mini-slotted channels, and a break-even point between them is given.
1. INTRODUCTION

Consider a system of $N$ transmission stations, transmitting messages via a common channel. Each station generates messages of fixed length, called packets, whose generation times form $N$ independent homogeneous Poisson processes with rates $\lambda_i$; $1 \leq i \leq N$; the process formed by the global generation times is obviously also Poisson, with rate $\lambda = \sum_{i=1}^{N} \lambda_i$.

The channel time is assumed to be slotted, i.e., divided into time segments (slots) corresponding in length to the packets. A slot may, without loss of generality, be taken as a unit of time. We also assume that the stations are synchronized with the channel in the sense that attempts to transmit packets are exactly time-aligned with the slots. While there is no restriction on the time of generation, that of transmission is dictated by the protocol (algorithm) to be used. Whenever one message is transmitted at a slot, it reaches its destination error-free; when there is no attempt to transmit a message at a given slot, we say that the slot is idle; finally, whenever two or more messages fall within the same slot, a conflict occurs.

In that event none of the messages reaches its destination and a conflict-resolution procedure has to be invoked.

At the end of each slot, every station learns whether zero, one or more than one messages were transmitted at that slot. What is sought, is a protocol which maximizes the throughput of the channel, i.e., the fraction of slots used for exactly one message.

This is equivalent to maximizing the arrival rate of new packets, $\lambda$, under which the underlying processes are ergodic (see Mosely (1979)). Since we are concerned with the maximal throughput rather than with any function of queue length, we may assume that the arrival rate is high; so whenever a packet is allowed to be transmitted on the basis of its genera-
tion time, it is already available (op.cit).

Numerous studies have been carried out on this model and on related ones; for a survey, the reader is referred to Massey (1980) and Lam (1979). The highest throughput known to-date (.48775) is associated with the optimal FIFS protocol due to Humblet (1980) and Mosely (1979), whereby messages are taken up for transmission under the exclusive criterion of their generation times. This may lead to transmission of more than one packet by the same station at the same slot. Such conflicts have probability zero when the number of stations is infinite, and positive when it is finite. Elimination of such conflicts from the protocol clearly increases the throughput in the finite case, as is shown in Sections 2 and 3 below. It is also shown that, even when the number of stations is very large, a throughput of more than .5 can be obtained when half the arrival rate is associated with a single station (a supervisor or a main computer station). Gallager's protocol, Gallager (1978), to which the Humblet-Mosely (H-M) protocol is closely related, is much simpler to analyze and yields the very close throughput of .48711. In Section 4, it is shown how successive refinements of Gallager's protocol improve the throughput to the same level as the H-M protocol without detriment to simplicity and implementation.

In Section 5, we consider a network with small propagation delay on which the interested reader can find a wide description in Franta (1981); for specific protocols and their performance analysis, see e.g. Chlamtac (1979) and Molle (1980). We use (in Section 5) a mini-slotted channel scheme and show how to adapt the optimal protocol in a slotted channel to a protocol in a mini-slotted channel, which is also optimal and avoids collisions.

A break-even point between a slotted and a mini-slotted scheme is also given.
2. TRANSMISSION PROTOCOLS

Let $\tau > 0$ be any positive number and $\mathbf{p} = (p_1, p_2, \ldots)$ be any sequence of probabilities. A Transmission protocol $\Pi = \Pi(\tau, \mathbf{p})$ will be defined below, by a semi-symbolic language, using the following local variables for every station $i$, $1 \leq i \leq N$.

- **CONFLICT (i)** equals unity if conflict resolution is in progress; zero otherwise;
- **S (i)** counts the number of consecutive successful transmissions as follows: it increases by one for every successful transmission and drops to zero for every conflict;
- **j (i)** is an index associated with an element in the sequence $p_1, p_2, \ldots$;
- **SLOT - NO (i)** indicates the next slot to be used for transmission;
- **FLAG (i)** equals unity if the station contains a selected message to be transmitted at the next slot; zero, if it contains a message whose transmission is delayed for the moment; or 'blank', if it contains no message to be transmitted during the current 'life-cycle' (defined below);
- **TRANS-RESULT (i)** is updated after each transmission and equals 2, 1 or 0 depending on whether a conflict, a successful transmission or no transmission occurred;
- **t (i)** denotes time.
Although, the protocol is decentralized, all local variables, except \( \text{FLAG (i)} \), have the same values at every station, so the index \( i \) can be omitted from their notation. A life-cycle of a protocol \( r(i, P) \) consists of the following steps, starting from step 1 until reaching again to step 1:

**Step 0:** (initialization) \( \text{CONFLICT}=0; S=0; j=0; \text{SLOT-NO}=0; \text{FLAG (i)}=\text{'blank', } 1 \leq i \leq N; \text{TRANS-RESULT}=0; t=0. \)

**Step 1:** For every station \( i \), perform 1.1 through 1.2;

1.1 If there exists a message in the queue whose generation time falls in the interval \( (t, t + \tau) \) perform 1.1.1 through 1.1.3, otherwise go to 1.2;

1.1.1 \( \text{FLAG (i)} = 1; \)

1.1.2 Keep one message in the queue and associate it with \( \text{FLAG (i)} \) (choice effected, say, on a FIFO basis);

1.1.3 Delete the remaining of the messages generated during \( (t, t + \tau) \);

1.2 End Step 1;

**Step 2:** For every station \( i \), perform 2.1 through 2.4;

2.1 \( \text{SLOT-NO} = \text{SLOT-NO} + 1; \)

2.2 If \( \text{FLAG (i)} = 1 \) transmit the message at the next slot;

2.3 Update \( \text{TRANS-RESULT} \);

2.4 End Step 2;

**Step 3:** For every station \( i \), perform 3.1 through 3.3;

3.1 If \( \text{TRANS-RESULT} = 2 \), perform 3.1.1 through 3.1.4;

3.1.1 \( \text{CONFLICT} = 1; \)

3.1.2 If \( \text{FLAG (i)} = 0 \), set \( \text{FLAG (i)} \) to 'blank' and delete the message associated with it;
3.1.3 \( S = 0 \);
3.1.4 Go to Step 4;
3.2 If \( \text{TRANS-RESULT} = 0 \) go to Step 5;
3.3 If \( \text{TRANS-RESULT} = 1 \) go to Step 6;
3.4 End Step 3;

**Step 4:** For every station \( i \) perform 4.1 through 4.3;

4.1 \( j = j + 1 \);
4.2 If \( \text{FLAG}(i) = 1 \) perform 4.2.1, otherwise go to 4.3;

\[ 4.2.1 \] Perform a Bernoulli trial with probability of success \( P_j \). If the trial is successful \( \text{FLAG}(i) = 1 \), otherwise, \( \text{FLAG}(i) = 0 \);
4.3 Go to Step 2;
4.4 End Step 4;

**Step 5:** For every station \( i \) perform 5.1 through 5.3;

5.1 If \( \text{CONFLICT} = 0 \), go to Step 7;
5.2 If \( \text{FLAG}(i) = 0 \), set \( \text{FLAG}(i) \) to 1;
5.3 Go to Step 4;
5.4 End Step 5;

**Step 6:** For every station \( i \) perform 6.1 through 6.6;

6.1 If \( \text{CONFLICT} = 0 \) then go to Step 7;
6.2 \( S = S + 1 \);
6.3 If \( S = 2 \), go to Step 7;
6.4 If \( \text{FLAG}(i) = 1 \), set \( \text{FLAG}(i) \) to 'blank' (message transmitted successfully);
6.5 If \( \text{FLAG}(i) = 0 \), set \( \text{FLAG}(i) \) to 1;
6.6 Go to Step 2;
6.7 End Step 6;
Step 7: For every station i perform 7.1 through 7.6:

7.1 \( t = t + 1 \);
7.2 \( j = 0 \);
7.3 \( S = 0 \);
7.4 \( \text{FLAG}(i) = \text{b'ank} \);
7.5 \( \text{CONFLICT} = 0 \);
7.6 Go to Step 1;
7.7 End Step 7;

At the end of each step the protocol proceeds to the next step unless it is directed otherwise. At step 1, each station selects a message for transmission according to its generation time. The transmission takes place, the result is recorded at step 2, and a conditioned branch is performed, according to it at step 3. At step 4 the messages at each station are separated by independent Bernoulli trials into two sets, one of which is slated for transmission at the next slot and the other delayed and taken up for further separation (or transmission) in step 5 (6) accordingly as the occurrence was idle (or successful). Finally at step 7, the variables are readjusted for the new life cycle beginning again with step 1.

REMARKS

(1) During a life-cycle of the protocol, the messages involved in the first conflict are dynamically classified into three sets. The first is known to contain at least two messages; two of these are successfully transmitted (using the separation procedure) before the life-cycle is ended. The second is known to contain at least one message; these messages are transmitted without separation. The third is known to contain at least zero messages (i.e., no additional common information...
available to the stations); these messages are deleted.

(2) For each successive run of the separation procedure (step 4) in a life-cycle a different \( p_j \) is used for the Bernoulli trials. The reason being so is, that less and less stations with \( \text{FLAG} (i) = 1 \) are expected at each successive visit to step 4, which suggests to take different probabilities of success.

(3) For \( \lambda_i = \lambda/N, 1 \leq i \leq N; \ p_j = 1/2, j = 1,2, \ldots, \) and \( N \rightarrow \infty \), the stochastic evolution of the system is the same as under Gallager’s protocol which (like the H-M protocol) performs separation (here in step 4) by classifying the messages in two sets according to their generation time. When \( N = \infty \) and the arrival times of the messages form a homogeneous Poisson process, the effect of this separation procedure on the stochastic evolution is the same as that of Bernoulli trials. This is, however, not the case with a non-Poisson process (e.g. any renewal process) or when \( N < \infty \) and only the first message is considered at every station during a life cycle. Thus the Bernoulli trial technique can be analyzed in the same way as in this paper, under more general conditions.

(4) The generalization achieved when \( p_j \) is allowed to depend on the number of visits to step 4 instead of setting \( p_j = 1/2 \) for all \( j \) is similar to that achieved by the H-M protocol versus Gallager’s protocol. There, the splitting ratio of the interval containing the generation times of the collided messages depends on the length of the interval (the shorter the interval, the fewer messages are expected). Indeed, as can be seen in Section 4, the throughput of our protocol for \( p_j = 1/2 \) is the same as Gallager’s and the optimal throughput is the same as H-M’s.

For every policy \( \Pi(\tau,P) \), let \( u(\Pi(\tau,P)) \) be the long-run average number of successful transmissions per slot (the throughput of the system).
The stochastic evolution of the system under the $\tilde{n}(\tau, p)$ protocols defines a Markov decision process. We omit the formal definition, since it is done in a similar way as in Mosely (1979). Moreover, the stochastic evolution is the same throughout all life-cycles of the protocol, and the life-cycles are statistically independent. Therefore, the mean ergodic theorem implies that

$$u(\Pi(\tau, p)) = E(\text{Number of successful transmissions during a life-cycle})/E(\text{Number of slots used during a life-cycle}).$$

(1)

Let $M_n$, $n \geq 2$, be the expected number of messages successfully transmitted, and $L_n$, $n \geq 2$, that of slots used, in a life-cycle, given that $n$ stations were involved in the initial conflict. For a given protocol $\Pi(\tau, p)$, the following recursive equations are straightforward:

$$M_n^i = (p_{i+1}(0,n) + p_{i+1}(n,n)) M_{n-1}^i + p_{i+1}(1,n)(1 + M_{n-1}^i) + \sum_{k=2}^{n-1} p_{i+1}(k,n) M_k^i + 1; n \geq 2, i \geq 0,$$

$$M_1^i = 1, \quad i \geq 0,$$

$$M_n = M_n^0, \quad n \geq 1.$$  

(2)

$$L_n^i = 1 + (p_{i+1}(0,n) + p_{i+1}(n,n)) L_{n-1}^i + p_{i+1}(1,n)(1 + L_{n-1}^i) + \sum_{k=2}^{n-1} p_{i+1}(k,n) L_k^i + 1; n \geq 2, i \geq 0,$$

$$L_1^i = 0, \quad i \geq 0,$$

$$L_n = L_n^0,$$

where $p_i(k,n) = Pr(X=k)$ and $X \sim B(k, p_i)$. Clearly, if for some $i$, $p_i = p$, $i \geq I$, then $M_n^{i-1} = M_n^i$ for $i \geq I$, and the set of recursive equations is finite.

Let $q_i(n)$ be the probability of $n$ stations being involved in the initial conflict. We have
\[ q^*_\tau(n) = \sum_{k_1=0,1, \ldots}^{n} \prod_{i=1}^{N} \left(1-e^{-\lambda_i \tau}\right)^{k_i} \left(e^{-\lambda_i \tau}\right)^{-k_i}. \]  \hspace{2cm} (3)

By (1) to (3), we have

\[ u(t,p) := u(\tau(p),p) = \frac{e^{-\lambda \tau} \left( \sum_{i=1}^{N} (e^{\lambda_i \tau} - 1) \right) + \sum_{n=2}^{N} q^*_\tau(n) N_n}{1 + \sum_{n=2}^{N} q^*_\tau(n) L_n}. \]  \hspace{2cm} (4)
3. OPTIMIZATION-RESULTS

From Kolonko's study, Kolonko (1980), on the continuity of Markov decision processes with average reward, it follows that $u^* = \max_{\tau, P} u(\tau, P)$ can be approached by $u^*(I) = \max_{\tau, P} \{u(\tau, P) | P_1 = P, i=I\}$, when $I \to \infty$. Since the improvement over the case $I=1$ is insignificant, we present the result for $I=1$ only.

Let $P = (P_1, P_2, P_3, \ldots)$. For every given number of stations $N$ and rates $(\lambda_1, \lambda_2, \ldots, \lambda_N)$, $u(\tau, P)$ (given in (4)) is readily maximized using a simple computer program. In Table 1 are given the optimization results for different numbers of stations and four classes of rates $\lambda_1, \ldots, \lambda_N, \sum \lambda_i = 1$, namely (i) $\lambda_i = \lambda/2$ and $\lambda_1 = \lambda/2(N-1)$ if $i = 2, 3, \ldots, N$; (ii) $\lambda_1 = \lambda_2 = \lambda/4$ and $\lambda_i = \lambda/2(N-2)$ if $i = 3, 4, \ldots, N$; (iii) $\lambda_1 = \lambda_2 = \lambda_3 = \lambda/6$ and $\lambda_i = \lambda/2(N-3)$, if $i = 4, 5, \ldots, N$; (iv) $\lambda_i = \lambda/N$, if $i = 1, 2, \ldots, N$.

Classes (i)-(iii) are typical of the case where some of the stations are computer systems and the others users.

REMARKS

(1) For all $N$, the lowest throughput is obtained when the stations are symmetric (i.e., have the same $\lambda_i$).

(2) The throughput increases as the number of station decreases; for $n \leq 4$, the throughput always exceeds .25.

(3) For all $N$, the throughput increases as the stations becomes "less symmetric".

(4) When one of the stations produces one-half of the messages on the average, the throughput exceeds .5.
<table>
<thead>
<tr>
<th>$N, \lambda_1, \lambda_2, \ldots, \lambda_N$</th>
<th>$P$</th>
<th>$\tau$</th>
<th>$u^*$</th>
<th>$N, \lambda_1, \lambda_2, \ldots, \lambda_N$</th>
<th>$P$</th>
<th>$\tau$</th>
<th>$u^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N=2; \lambda_1=\lambda_2 = 1/2$</td>
<td>$.480$</td>
<td>$1.60$</td>
<td>$.5512$</td>
<td>Category (iii)</td>
<td>$.480$</td>
<td>$1.32$</td>
<td>$.4982$</td>
</tr>
<tr>
<td>$\lambda_1=1/3, \lambda_2=2/3$</td>
<td>$.475$</td>
<td>$1.37$</td>
<td>$.5409$</td>
<td>&quot; (iv)</td>
<td>$.475$</td>
<td>$1.28$</td>
<td>$.4899$</td>
</tr>
<tr>
<td>$\lambda_1=1/4, \lambda_2=3/4$</td>
<td>$.475$</td>
<td>$1.43$</td>
<td>$.5585$</td>
<td>&quot; (i)</td>
<td>$.485$</td>
<td>$1.40$</td>
<td>$.5202$</td>
</tr>
<tr>
<td>$N=3; category (iv)$</td>
<td>$.490$</td>
<td>$1.50$</td>
<td>$.5375$</td>
<td>&quot; (ii)</td>
<td>$.480$</td>
<td>$1.34$</td>
<td>$.5032$</td>
</tr>
<tr>
<td>$N=4; category (iv)$</td>
<td>$.485$</td>
<td>$1.43$</td>
<td>$.5223$</td>
<td>&quot; (iii)</td>
<td>$.480$</td>
<td>$1.32$</td>
<td>$.4979$</td>
</tr>
<tr>
<td>$N=5; category (i)$</td>
<td>$.490$</td>
<td>$1.46$</td>
<td>$.5317$</td>
<td>&quot; (iv)</td>
<td>$.475$</td>
<td>$1.28$</td>
<td>$.4887$</td>
</tr>
<tr>
<td>&quot; (ii)</td>
<td>$.485$</td>
<td>$1.40$</td>
<td>$.5154$</td>
<td>N=200; &quot; (i)</td>
<td>$.485$</td>
<td>$1.40$</td>
<td>$.5200$</td>
</tr>
<tr>
<td>&quot; (iv)</td>
<td>$.485$</td>
<td>$1.39$</td>
<td>$.5142$</td>
<td>&quot; (ii)</td>
<td>$.480$</td>
<td>$1.34$</td>
<td>$.5030$</td>
</tr>
<tr>
<td>$N=10; &quot; (i)</td>
<td>$.485$</td>
<td>$1.43$</td>
<td>$.5249$</td>
<td>&quot; (iii)</td>
<td>$.480$</td>
<td>$1.32$</td>
<td>$.4977$</td>
</tr>
<tr>
<td>&quot; (ii)</td>
<td>$.480$</td>
<td>$1.36$</td>
<td>$.5074$</td>
<td>&quot; (iv)</td>
<td>$.475$</td>
<td>$1.28$</td>
<td>$.4881$</td>
</tr>
<tr>
<td>&quot; (iii)</td>
<td>$.480$</td>
<td>$1.34$</td>
<td>$.5024$</td>
<td>N=500; &quot; (i)</td>
<td>$.485$</td>
<td>$1.40$</td>
<td>$.5198$</td>
</tr>
<tr>
<td>&quot; (iv)</td>
<td>$.480$</td>
<td>$1.33$</td>
<td>$.4999$</td>
<td>&quot; (ii)</td>
<td>$.480$</td>
<td>$1.34$</td>
<td>$.5029$</td>
</tr>
<tr>
<td>$N=20; &quot; (i)</td>
<td>$.485$</td>
<td>$1.41$</td>
<td>$.5222$</td>
<td>&quot; (iii)</td>
<td>$.480$</td>
<td>$1.31$</td>
<td>$.4976$</td>
</tr>
<tr>
<td>&quot; (ii)</td>
<td>$.480$</td>
<td>$1.35$</td>
<td>$.5048$</td>
<td>&quot; (iv)</td>
<td>$.475$</td>
<td>$1.27$</td>
<td>$.4877$</td>
</tr>
<tr>
<td>&quot; (iii)</td>
<td>$.480$</td>
<td>$1.32$</td>
<td>$.4995$</td>
<td>N=1000; &quot; (i)</td>
<td>$.485$</td>
<td>$1.40$</td>
<td>$.5198$</td>
</tr>
<tr>
<td>&quot; (iv)</td>
<td>$.480$</td>
<td>$1.30$</td>
<td>$.4935$</td>
<td>&quot; (ii)</td>
<td>$.480$</td>
<td>$1.34$</td>
<td>$.5028$</td>
</tr>
<tr>
<td>$N=50; &quot; (i)</td>
<td>$.485$</td>
<td>$1.41$</td>
<td>$.5207$</td>
<td>&quot; (iii)</td>
<td>$.480$</td>
<td>$1.31$</td>
<td>$.4976$</td>
</tr>
<tr>
<td>&quot; (ii)</td>
<td>$.480$</td>
<td>$1.34$</td>
<td>$.5036$</td>
<td>&quot; (iv)</td>
<td>$.475$</td>
<td>$1.27$</td>
<td>$.4876$</td>
</tr>
</tbody>
</table>

Table 1.
4. THE CASE N = ∞

Assume now the limit case, $\Sigma \lambda_i = \lambda$, $N \to \infty$ and $\lambda_i \to 0$ (as in Mosely (1979) and Gallager (1978)). Note that in these circumstances, the probability of more than one message arriving at the same station in any finite interval of time, is zero. From Kolonko (1980), it follows that the optimal throughput under the set of protocols defined in Section 2, can be approached by optimizing on protocols $\Pi(\tau, P)$ with $P_i = P$ for $i \geq 1$ and $I \neq \infty$.

For $N = \infty$ and different values of $I$ we obtain from (4) the following values $u^* = \max_{\tau, P} u(\tau, P)$:

For $I = 1$, $u^* = u(1.272, P) = .48757$, where $P_1 = .476$, $i \geq 1$.

For $I = 2$, $u^* = u(1.272, P) = .48773$, where $P_1 = .46$, $P_2 = .485$, $i \geq 2$.

For $I = 3$, $u^* = u(1.275, P) = .48775$, where $P_1 = .46$, $P_2 = .48$, $P_3 = .495$, $i \geq 2$.

For $I = 1$, a simple generalization of Gallager's protocol is obtained by taking $P = .476$ instead of $P = .5$; for $I = 2$, we already have the throughput of H-M's protocol to the fourth decimal point.
5. THE MINI-SLOTTED CHANNEL

When the propagation delay is small, the throughput of the channel can be improved by using a carrier-sense with collision detect device and a mini-slotted channel. (see e.g. Molle (1980)). It is shown below that the maximal throughput of a mini-slotted channel can be found in the same method as of the slotted one; a break-even point between them is also given.

Let \( a \) be the maximal propagation delay between two stations. The following property is introduced by the carrier-sense with collision detect device: when station \( i \) transmits a message to station \( j \) at time \( t \), all stations are informed about it until time \( t + a \); moreover, every station can differentiate between a successful transmission, a conflict and an idle channel. The mini-slotted, as the slotted, channel is also assumed to be slotted, but with slot length of \( a < 1 \), rather than of unity (a packet length). This kind of slot is called mini-slot. In addition to regular packets (of length unity), each station, which has a regular packet in its queue, may transmit a 'contention bit' (of length \( a \)). These bits are used by the stations (as described below) to grant channel time for transmitting regular packets. Let \( \Pi(\tau,P) \) be any protocol given in Section 2. Each station, which has a regular packet in its queue, transmits 'contention bits' during the mini-slots according to protocol \( \Pi(\tau,P) \). Once a 'contention bit' is successfully transmitted by a station, it grants the channel for a unit of time starting from the next mini-slot. During this period it transmits a regular packet without interruption. Right after the transmission is over, the protocol \( \Pi(\tau,P) \) is resumed with 'contention bits'.
By the same arguments as in Section 2, it follows that every transmission of a regular packet is proceeded by \( a/u(\tau, P) \) mini-slots on the average. Thus, the throughput of the mini-slotted channel is

\[ \text{UMS}(\tau, P) = \frac{1}{1 + a/u(\tau, P)}. \]  

(5)

Let \( u^* = \max u(\tau, P) \) and \( \text{UMS}^* = \max \text{UMS}(\tau, P) \).

From (5) we have

\[ \text{UMS}^* = \frac{1}{1 + a/u^*}. \]  

(6)

From (6) we have the break-even point between the slotted and the mini-slotted channel which is given by (7) below.

\[ \text{UMS}^* > u^* \text{ if and only if } a < 1 - u^*. \]  

(7)
REFERENCES


