MULTIPLE ACCESS PROTOCOLS*

Simon S. Lam

TR-88 January 1979
Revised, Sept. 1980

Department of Computer Sciences

The University of Texas at Austin

Austin, TX 78712

* To appear as a chapter in Computer Communications, Vol I: Principles W. Chou (editor), Prentice-Hall, 1981. This work was supported by National Science Foundation under Grant No. ENG78-01803.

ABSTRACT

Broadcast networks are considered in which a population of distributed users are interconnected via a broadcast channel (e.g. satellite, radio, multipoint cable etc.). Of interest in this paper are multiple access protocols for sharing use of the broadcast channel by the users. The central problem is that of identifying users desiring channel access and resolving conflicts when more than one such users are present. A general model for characterizing user traffic is first presented, which is followed by a classification of multiple access protocols. Our emphasis is on packet-oriented protocols rather than the presently prevalent channel-oriented protocols. Among packet protocols we differentiate between those for passive users (the class of polling protocols) and active users (contention and reservation protocols). Attributes of the various classes are examined and illustrated with description of specific protocols. Their delay-throughput tradeoff performance characteristics are examined and compared under a variety of traffic assumptions.

TABLE OF CONTENTS

1. INTRODUCTION

Packet broadcasting networks may be defined to be packet switching networks in which the connectivity requirements of the network users are furnished by a broadcast medium (channel). Two examples of broadcast media are satellite and terrestrial radio channels. A less obvious example is that of a multipoint cable. Although multipoint networks have been in use for many years as broadcast networks, it was the ALOHANET using radio channels and a novel contention protocol [ABRA 70, BIND 75a] that first called attention to the concept of packet broadcasting as we know it now. This concept clearly emerged with the packet satellite project [ABRA 73, KLEI 73, ROBE 73, CROW 73, BUTT 74, JACO 77, JACO 78, WEIS 78] and the packet radio project [KAHN 77, KAHN 78] of the Advanced Research Projects Agency. Currently, packet broadcasting networks appear to be important in two areas. The first is satellite data networks. The rapid pace of development of commercial satellite systems in recent years has enabled substantial reductions in satellite system costs [ARBA 75]. Meanwhile, another market place is expanding rapidly: local area networks which, though outside the computer room, are confined to local environments e.g. office complex, manufacturing plant etc. Such networks based upon CATV technology are said to offer real advantages to computing facilities which serve many terminals and/or employ distributed processing [WILL 74, THOR 75, DEMA 76, METC 76, WEST 78]. An extensive bibliography of local computer networks can be found in [CLAR 78, THUR 79].

Of interest in this paper are protocols for sharing a single broadcast channel by a population of users. The users have random traffic requirements as well as delay constraints. The central problem is that of conflict resolution among users desiring channel access. The methods for conflict resolution will be referred to as multiple access protocols. We shall present a classification of such protocols, illustrate each class with description of specific protocols, and compare their performance under various traffic and channel assumptions. To delimit the scope of this paper, we shall consider only networks in which everyone in the population of users can "hear" everyone else over the broadcast channel i.e. we have a one-hop broadcast network.

The key measure of performance of a multiple access (MA) protocol is its channel throughput versus average delay tradeoff characteristic. The throughput of a channel is defined as follows. Let C be the channel transmission rate in bits per second (bps) and let there be on the average P bits in a transmitted block of data. The channel throughput S is defined to be the ratio of the rate of successfully transmitted data blocks to the rate C/P. Thus channel throughput is a normalized quantity between 0 and 1. It includes as useful throughput overhead bits contained in data blocks for bit synchronization, addressing, error control and other network control functions; these overheads however are not directly attributable to the MA protocol. Other overheads that are directly attributable to the implementation of an MA protocol are accounted for in the calculation of channel throughput; the maximum achievable channel throughput is thus less than one and is of interest as a gross measure of performance.

We shall next discuss briefly traditional multiple access protocols which are channel-oriented rather than packet-oriented (of interest here). A model for characterizing traffic sources is then presented. Following that, the characteristics of different catagories of multiple access protocols are surveyed. Specific protocols and their performance are examined in some detail in Sections 2,3 and 4.

1.1 Traditional Techniques

The problem of multiple access in the design of satellite systems, for example, has been solved in the past with voice communications in mind. The design objective is to maximize the number of (voice grade) channels for given constraints of power and bandwidth. The common satellite multiple access techniques are: frequency division multiple access (FDMA), time division multiple access (TDMA), and code division multiple access (CDMA) [SCHW 73]. CDMA is also called spread spectrum multiple access. It is by far the least efficient and has been mainly used for military systems with anti-jam and security requirements. TDMA is generally more efficient than FDMA; the price paid is an increase in the cost and complexity of the equipment of each user [PRIT 77].

Multiple access protocols have traditionally been channel-oriented. The transmission capacity available is subdivided into separate channels (with FDMA or TDMA). The basic unit for allocation is thus a channel. Channels can be either (i) fixed assigned, or (ii) demand assigned to users. With demand assignment, a channel needs to be set aside for signalling among users. (Access to the signalling channel is another multiple access problem! A typical solution for this is TDMA with fixed assignment.) Demand assignment can then be accomplished with either a central controller or distributed control algorithm [PRIT 77, PUEN 71].

The channel-oriented MA protocols are suitable for voice traffic and may also be suitable for some data traffic. Data communications in general, however, have very diverse requirements ranging from inquiry-response systems with intermittent traffic to file transfers with large volumes of data. In this paper, we are interested mainly in the class of new packet-oriented MA protocols for time-sharing a single broadcast channel. The broadcast channel under consideration for the shared use of a population of users may possibly have been derived at a higher level of resource allocation via FDMA, TDMA, or CDMA. (The allocation scheme at this level will be of no concern to us). We shall concentrate upon only the conflict resolution aspect of the multiple access problem and its performance in terms of throughput and delay. The other aspects of the multiple access problem (such as modulation, coding and clock synchronization) are classical problems and are beyond the scope of this paper; see, for example, [ELLI 73, JACO 74, GARD 80].

1.2 Traffic Model

The following model will be used to represent the traffic characteristics and transmission requirements of users of a message-(or packet-) oriented communication network. Examples of users are: human operators of computer terminals, computer programs interacting with such human operators or with other programs/databases in other machines, data concentrators for a multiplicity of such traffic sources, etc. We shall view a user simply as a traffic source that can be modeled as a random point process with instants of message arrivals being the points of interest. A message is defined to be a block of data that has a time delay constraint associated with it for delivery to a destination user. (In a packet-oriented network, a message may be transported in one or more packets.) The above definition is quite general. For example, a message may be a computer data file that needs to be delivered within a period of hours. It may be a line of characters in an inquiry-response system that needs to be delivered in a fraction of a second. It may be a digital voice sample (8 bits) that has a delay constraint dictated by the real-time voice sampling rate (e.g., 8000 samples per second).

Computer data traffic sources are often described as "bursty." Traffic burstiness is an important characteristic that influences the design of packet communication systems. This concept is briefly reviewed and a quantitative measure of burstiness, called bursty factor, as defined in [LAM 78a] is presented below.

The bursty nature of a data traffic source stems from more than just the randomness in message generation time and size. The user-specified message delay constraints to be met for these traffic sources are actually the single most important factor in determining if data traffic sources behave in a bursty manner. Suppose we are given a traffic source with

T = average interarrival time between messages

and

δ = average message delay constraint

where δ can be estimated in practice from the performance specifications of the intended network users. Generally, a user-specified source-destination delay constraint can be broken up into several parts, each part becoming a constraint for a segment of the communication path. δ, is defined here to be the constraint for the message transmission time plus any necessary conflict resolution and queueing delays; it excludes, however, propagation delays through the network as well as message processing delays at the source and destination.

The bursty factor β of the traffic source is defined to be

$$\beta = \delta/T \tag{1}$$

Note that β depends only upon δ and T which are inherent user characteristics. (The delay constraint δ is indeed an inherent source characteristic in the eyes of the network designer. Failing to satisfy it means that the user will take his business elsewhere!) Next consider a traffic source that is formed by merging together N sources with different statistics and delay constraints. The bursty factor of the aggregate source is defined to be the sum of the bursty factors of the individual sources, namely

$$\beta = \beta_1 + \beta_2 + \ldots + \beta_N \tag{2}$$

The usefulness of β is due to the following observation [LAM 78a]. Suppose a communication channel is dedicated to a traffic source with bursty factor β and all delay constraints are met. Then the resulting ratio of peak to average channel data rates (PAR) satisfies

$$\text{PAR} \geq 1/\beta \tag{3}$$

and the channel throughput S satisfies

$$S \leq \beta \tag{4}$$

In other words, β gives an upper bound on the duty cycle of a traffic source. This information is useful to the network designer and is available independent of the communication system eventually provided. A traffic source with a small $\beta (<<1)$ is said to be <u>bursty</u>.

The above result also says that for bursty users, channel-oriented MA protocols using either fixed assignment or demand assignment (over a period of time $>>\delta$) are going to be very inefficient ($S << 1$). To improve the throughput of a broadcast channel shared by users with random bursty traffic, it is desirable to dynamically allocate transmission capacity on a per message (or packet) basis. This benefits from both the multiplexing effect of Eq. (2) as well as the statistical averaging effect of the law of large numbers [KLEI 76, section 5.1]. The key to realizing these gains is to design MA protocols for resolving channel access conflicts without excessive overhead.

We note that the definition of bursty factor in Eq.(1) does not involve the average length of messages generated by a traffic source. Thus it is possible for a traffic source to generate very long messages but is still considered to be very bursty by definition. This average message length parameter, we shall see, will also strongly impact the performance of MA protocols.

1.3 Network Assumptions

We consider a population of N users sharing use of a single broadcast channel. The ith user has a message generation rate of λ_i messages per second. Each message may give rise to one or more fixed length packets with a mean number of L. Each packet carries a destination address so that when the packet is transmitted over the channel, with no interference from another user, it will be received by the proper addressee(s). Errors due to channel noise will be ignored. Such errors affect MA protocols equally and their effect can be evaluated separately. Notice that the specific connectivity requirements of the population of users are relevant only indirectly through the resulting set of message rates. Given the set of λ_i, it does not matter, for example, whether users want to communicate with each other or they all want to talk to a specific central site. Thus, our only concern is the access problem of the broadcast channel.

Each user is capable of sending and receiving data at the channel transmission rate of C bps. In a number of MA protocols, the users are time synchronized so that the channel can be viewed upon as a sequence of time slots (just as in TDMA). Each time slot can accommodate one data packet. Minislots may also be interleaved with the data slots to accommodate small control packets.

A user when given access to the broadcast channel can send data to any other user. Thus, the MA protocol is simply an algorithm (possibly distributed as well as non-deterministic) for determining the channel access rights of the users. In some protocols, the access right is not uniquely determined and it is possible for packet transmissions from different users to "collide" in the channel. It is assumed that the collisions are always destructive and none of the packets involved in a collision can be correctly received. Each packet contains parity bits for error detection.

Suppose it takes R seconds following a packet transmission for a user to find out whether it suffered a collision. R will be referred to as the collision detection time. In many systems [ABRA 73, METC 76], the user can find out the outcome of a transmission for himself by monitoring the broadcast channel. In this case, R is approximately equal to the propagation delay of the channel. (The propagation delay is always taken to be the maximum value between any pair of sender and receiver in the network.) If the above collision detection mechanism is not possible, then R corresponds to the time-out period of a positive acknowledgment protocol [ABRA 70, BIND 75a].

1.4 A Classification of Protocols

The gamut of packet-oriented MA protocols can be classified as shown in Figure 1 and may be illustrated by the following analogy.
Consider a group of students in a classroom who want to say something. Assume that if two or more students talk at the same time, the resulting speech will be unintellibible.

Passive versus active talkers

The students may be "passive." A student will talk only when specifically asked to do so by a teacher acting as a central controller. The teacher determines who wants to talk by polling the students one by one. Or the students may be "active." Whoever has something to say will try to do something about getting the opportunity to talk. Among protocols for active talkers, we further differentiate between two categories as follows.

Contention versus reservation protocols

Under a contention protocol, there is no attempt to coordinate the talkers. Each student tries to talk whenever he has something to say. In doing so, he might exercise some caution to try to minimize interference with other talkers by observing past activities in the room. The alternative is to use reservation protocols which may use one of the following two types of control.

Centralized versus distributed control algorithm

Under a reservation protocol, each student who wants to talk raises his hand to signal a request to talk. A teacher may be present to serve as a central controller to determine who should talk next. Alternatively, each student monitors the requests of all students and exercises a distributed algorithm to determine who should talk next.

In the next three sections, we shall illustrate each class of protocols introduced above with descriptions of specific protocols. Performance implications of various channel and traffic parameters are discussed. In particular, a number of the protocols rely upon a short channel propagation delay for their efficiency and are thus not suitable for satellite channels.

2. PROTOCOLS FOR PASSIVE USERS

This class of protocols is commonly known as polling protocols. The presence of a central controller is required. Users are passive in the sense that they may access the channel only when specifically polled by the central controller. The controller has two functions: (1) to identify users with data to send (the ready users), and (2) to schedule usage of the channel by these users.

2.1 Conventional Polling Protocols

In conventional polling protocols, the central controller performs the above two functions by polling the population of users one after the other. Upon the arrival of a polling message, the user polled transmits all messages accumulated in his buffer. The time needed to poll every user once and transmit their accumulated data is said to be the polling cycle time t_c. This is just the time between successive polls of a specific user.

A very important parameter determining the efficiency of polling protocols is the total "walk time" in a polling cycle. This is the portion of cycle time attributable to necessary overheads such as: channel propagation delay, transmission time of polling and response messages, modem synchronization time, etc. A detailed breakdown of time elements in a polling cycle can be found in [MART 72].

Minor implementation variations of the above description exist, e.g. prioritized polling list, some users polled more than once in a cycle, the amount of data that a user can transmit at a time is limited, and others [MART 72]. These will not be considered.

The two conventional polling protocols most widely used are: roll-call polling and hub polling [SCHW 77, Chapter 12]. In roll-call polling, the central controller sends a polling message to the user polled. The user transmits his waiting data (if any) and passes control back to the central controller before the next user is polled. In hub polling, the central controller initiates a polling cycle by polling the user at the top of its polling list. The user transmits his waiting data (if any) and passes the polling message on to the next user directly. The user at the bottom of the list passes control back to the central controller to complete a polling cycle. The obvious advantage of hub polling over roll-call polling is that the average walk time $\bar{w}$ between users is typically less. However, the individual users need to be more intelligent and reliable than those of roll-call polling.

To illustrate the key performance tradeoffs in the design of polling protocols, we consider here some results from a queueing model of N identical users [KONH 74]. The same model applies to both roll-call and hub polling. (A detailed analysis of polling which takes into account various system features and response time elements can be found in [CHOU 78]).

Suppose each user has an arrival rate of λ messages/second and an average message transmission time of $1/\mu$ seconds.

Define

$$\rho = \lambda/\mu.$$

Let $\bar{w}$ be the average walk time between users and $\bar{t}_c$ be the average cycle time. Then it can be shown that [KONH 74]

$$\bar{t}_c = \frac{N\bar{w}}{1-N\rho} \tag{5}$$

The waiting time of a message is defined here to be the interval from its time of arrival to the start of its transmission. An expression for the average waiting time W is derived in [KOHN 74]. It can be easily shown that

$$W \geq \frac{\bar{t}_c}{2}(1-\rho)$$

Thus the simple expression in Eq. (5) can be used as an indirect measure of the average delay of a message which is

$$D = W + (1/\mu).$$

Eq. (5) has the usual queueing theory form with $N\rho$ being the traffic intensity of the broadcast channel. We have in the limit $\bar{t}_c \uparrow \infty$ as $N\rho \uparrow 1$. Observe that $\bar{t}_c$ is also directly proportional to the total walk time $N\bar{w}$ in a polling cycle. Consequently the delay performance may be greatly degraded if $\bar{w}$ is large (e.g. satellite channel) or N is large.

We next consider a performance booby trap of polling protocols. Recall that one of the reasons for employing packet- or message-oriented protocols in the first place is to take advantage of the statistical averaging effect of the law of large numbers [KLEI 76, section 5.1]. Suppose N is increased but a higher speed channel is used such that the channel utilization $N\rho$ remains the same. This change will tend to decrease the average delay D as a result of the statistical averaging effect. However, with a conventional polling protocol, the same change will affect D adversely as well. Increasing N while keeping $N\rho$ constant increases the $\bar{t}_c/2$ portion of the average delay due to the increased polling overhead $N\bar{w}$!

So far we have assumed users with plenty of buffer capacity (i.e. infinite waiting room for queues). Under this assumption the maximum channel throughput of polling is 1 in the limit of infinite queues.

Consider now users which have limited buffering capacity so that each user can transmit at most 1 message at a time. One consequence is that $\bar{t}_c$ is always finite. In addition, the channel throughput S can be easily shown to be bounded as follows.

$$S \leq \frac{1/\mu}{\bar{w} + (1/\mu)} \tag{6}$$

The upper bound in Eq. (6) is the maximum channel throughput of polling for such users. In this case, polling becomes extremely inefficient when the traffic consists of very short messages, e.g., terminals doing character-at-a-time transmission [WEST 72].

2.2 Adaptive Polling

With conventional polling, when the network is lightly loaded, it is still necessary to poll every one of the users although few users are ready (have data to send). The mean delay D in this situation is mainly determined by the polling overhead and not by the channel traffic intensity!

For a lightly loaded network, an adaptive polling protocol has been proposed by Hayes [HAYE 78] which significantly reduces the polling overhead. The key idea introduced is called probing. When a group of (2 or more) users are probed, a polling message is sent by the central controller to all users in the group. Any users with data to send respond by transmitting some signal in the channel. In order to avoid cancellation by interference, the signal can be some noise energy. Note that if a group of users is probed and none responds, the whole group can be ignored until the next polling cycle. Thus when the network is lightly loaded, significant polling overhead reduction results through probing groups of users. The following implementation is suggested [HAYE 78].

Assume a total of $N = 2^n$ users. Each user is addressed by an n bit binary number. The central controller can probe a group of users by sending the common prefix of the addresses of the users. At the beginning of each polling cycle, the central controller divides the users into 2^{n-j} groups of 2^j users each. The determination of j will be discussed later. Each of the 2^{n-j} groups is probed separately. If probing a group produces a positive response, each group is divided into two groups which are then probed separately. This process is repeated until all ready users are identified and serviced.

A poll (of a single user) or a probe (of 2 or more users) will both be referred to as an inquiry. An example is illustrated in Fig. 2 with a group of 8 users. Only one of them (address 011) is ready. Starting with probing all users ($j=0$), a total of 7 inquiries (5 probes and 2 polls) are needed to complete the polling cycle.

Returning to our general problem, let us consider the following special cases.

Case 1. Pure polling, $j=0$.

The number of inquiries required is 2^n which is independent of the number of ready users.

Case 2. Pure probing, $j=n$.

Pure probing is excellent under very light loading. For instance, if there is exactly one ready user, the number of inquiries is $2n + 1$. However, pure probing is penalized under heavy loading. If all users are ready, the number of inquiries will be $2^{n+1} - 1$.

From these extreme cases, we see that the group size 2^j to be selected at the beginning of each polling cycle is crucial to the performance of the protocol. It should be chosen to minimize the expected number of inquiries in the current polling cycle. Let

$$p = \text{Prob[a user has data to send in the current polling cycle]}$$

$$\bar{I}(j) = E[\text{number of inquiries}/j]$$

We want to determine j^* such that

$$\bar{I}(j^*) = \min_{0 \le j \le n} \bar{I}(j)$$

Let us compare the probing of a group of 2^j users and probing 2 groups of 2^{j-1} users. If we probe 2^j users, the probability that none will respond to the probe is

$$\text{Prob[no response}/j] = (1-p)^{2^j}$$

In this event, the choice of j instead of j-1 saves one inquiry. On the other hand, in the event that there is a positive response from the group of 2^j users, the choice of j instead of j-1 spends one more inquiry. Thus, we have

$$\begin{aligned}\Delta\bar{I}(j) &= \bar{I}(j) - \bar{I}(j-1)\\ &= 2^{n-j}\,[(-1).(1-p)^{2^j} + (1 - (1-p)^{2^j})]\\ &= 2^{n-j}[1 - 2(1-p)^{2^j}]\end{aligned}$$

for j = 1,...,n. Note that the term $1 - 2(1-p)^{2^j}$ increases monotonically with j. Thus, $\bar{I}(j)$ is minimized by the largest j such that*

$$1 - 2(1-p)^{2^j} < 0$$

or

$$(1-p)^{2^j} > \frac{1}{2} \tag{7}$$

Equation(7) tells the central controller how to determine j at the beginning of each polling cycle as a function of p. An estimate of p is

$$p = 1 - e^{-\lambda t_c} \tag{8}$$

where λ is the Poisson rate of message arrivals to a user, t_c is the duration of the last polling cycle. Eq.(8) assumes that messages arriving during one polling cycle do not respond positively to an inquiry until the next polling cycle. (This gives a pessimistic view of the operation of the system.) From Eqs.(7) and (8), we have,

$$2^j < \frac{\ln 2}{\lambda t_c} \tag{9}$$

* This is a simpler proof than the one found in [HAYE 78]. It is interesting to note that in general if addresses use base m numbers so that each node of the tree in Fig. 2 has m branches, the optimality condition is $(1-p)^{m^j} > \frac{1}{m}$

At the beginning of each polling cycle, the central controller can determine j^* as a function of λ and t_c using Eq.(9).

Performance summary

The mean cycle time for the adaptive polling protocol is still basically Eq.(5) but with the polling overhead term $N\bar{w}$ considerably reduced, especially under light loading. For instance, if there is only one ready user in a polling cycle, the number of inquiries is $2 (\log_2 N) + 1$ instead of N. The adaptivity in the protocol assures that there is no penalty during periods of heavy load. The above observations are illustrated in Fig. 3 which compares the mean number of inquiries per polling cycle as a function of p for pure polling, pure probing and the optimum adaptive strategy using Eq.(7).

3. PROTOCOLS FOR ACTIVE USERS

We next consider two classes of protocols which require ready users to actively seek channel access instead of waiting to be polled. These are called contention and reservation protocols. Under contention protocols, there is no attempt to coordinate the ready users to avoid collisions entirely. Instead, each ready user makes his own decision regarding when to access the channel: he exercises caution however to minimize interference with other ready users as much as possible. Under reservation protocols, a reservation channel is provided for ready users to communicate among themselves such that only one ready user is scheduled for channel access at a time. Since users are geographically distributed, the multiple access problem has not really disappeared; it now exists in the access of the reservation channel for the transmission of small reservation requests.

3.1 Contention Protocols

Unlike polling protocols, the overhead incurred by contention protocols for assigning channel access to ready users is independent of N but instead is dependent upon the level of channel traffic. Thus, pure contention protocols are suitable for a large population of bursty users (i.e. $N \uparrow \infty$ as $\beta \downarrow 0$ for each user but $N\beta$ remains constant). For a lightly loaded network, the delay performance of contention protocols can be far superior to polling protocols. Below, we shall describe two pure contention protocols (ALOHA and slotted ALOHA), two contention protocols which include elements of reservation (R-ALOHA and CSMA), and an adaptive protocol (URN). We shall discuss, in the context of slotted ALOHA, the stability problem of contention protocols and the need for adaptive control in these protocols. We shall then describe a tree algorithm for resolving contention which guarantees channel stability. The algorithm is based upon essentially the same idea as that of the adaptive polling algorithm described earlier.

The ALOHA protocol [ARBA 70, BIND 75a]

Under the ALOHA protocol, users are not synchronized in any way. Each user transmits a data packet whenever one is ready. In the event that two or more packets collide, i.e. overlap in time, each of the users involved realizes this after R seconds (the collision detection time) and retransmits his packet after a randomized delay. As we shall discuss below, this randomized delay turns out to be crucial to the stability behavior and thus the throughput-delay performance of all contention-based protocols.

In [ARBA 70], Abramson first derived the maximum channel throughput of the ALOHA protocol in the limit of an infinite user population ($N \uparrow \infty$ and for each user $\beta \downarrow 0$). All messages consist of single packets. Hence, the aggregate packet "birth process" is a Poisson process at a rate of S packets per packet transmission time. Abramson also made the assumption that the sum of new transmissions and retransmissions in the channel (called channel traffic) can be approximated as a Poisson process at a rate of G packets per packet time. Furthermore, statistical equilibrium is assumed. The probability that a transmitted packet is successful is

$$\frac{S}{G} = e^{-2G} \qquad (10)$$

which is obtained from consideration of Fig. 4; each transmitted packet has a vulnerable period of 2 packet time duration. It will be successful only if no other packet begins transmission within the vulnerable period.

From the above equation, the maximum possible ALOHA channel throughput is obtained at $G = 0.5$ and for an infinite user population model (under the above assumptions) is

$$C_A = \frac{1}{2e} \tilde{=} 0.184 \tag{11}$$

The slotted ALOHA protocol [ROBE 72, ABRA 73, KLEI 73]

The slotted ALOHA protocol is just like ALOHA with the additional requirement that the channel is slotted in time. Users are required to synchronize their packet transmissions into fixed length channel time slots. By requiring synchronization of packet start times, packet collisions due to partial overlaps are avoided and the vulnerable period of a transmitted packet is just the duration of a time slot (see Fig. 4.) Under the same assumptions given above for ALOHA, we have

$$\frac{S}{G} = e^{-G} \tag{12}$$

where S is maximized at $G = 1$. The resulting slotted ALOHA maximum channel throughput for an infinite user population model is twice that of the unslotted case;

$$C_{SA} = \frac{1}{e} \tilde{=} 0.368 \tag{13}$$

Equations (10) and (12) are plotted in Fig. 5.

Two methods of scheduling retransmissions following the detection of a collision have been considered [METC 73, LAM 74]:

(1) Geometrically distributed delay—A collided packet is retransmitted in a time slot with probability $p < 1$. With probability $1 - p$, the packet is delayed and the Bernoulli trial is repeated in the next time slot.

(2) Uniformly distributed delay—A time slot is selected for retransmitting a collided packet from the next K time slots chosen equally likely.

For purposes of performance modeling, simulation studies indicate that in many cases the mean value $\bar{k}$ (instead of the exact probability distribution) of the retransmission delay is sufficient for predicting the behavior of a slotted ALOHA channel [LAM 74].

Performance considerations

The above analysis of ALOHA and slotted ALOHA was based upon three assumptions: (A1) statistical independence of channel traffic, (A2) infinite user population, and (A3) statistical equilibrium. More detailed studies of slotted ALOHA investigated the validity of these assumptions. In particular, it was shown that a necessary condition for (A1) is that the mean randomized delay $\bar{k}$ for retransmissions must be large ($\bar{k} \to \infty$). However, simulations showed that the maximum channel throughput results given by Eqs. (11) and (13) are robust; they are already quite accurate if $N \geq 10$ and $\bar{k} \geq 5$ [KLEI 73, LAM 74]. In these same references, the delay performance of slotted ALOHA was first studied for finite values of $\bar{k}$. When the traffic distribution is unbalanced with a mixture of high-rate as well as low-rate users, it was shown that the slotted ALOHA maximum channel throughput of 1/e can be considerably improved [ARBA 73].

The validity of assumption (A3) was also examined. It was shown that the slotted ALOHA protocol, without adaptive control, is potentially unstable [METC 73, LAM 74, KLEI 74, KLEI 75a]. (We might have deduced this from the curves in Fig. 5, which show two equilibrium values of G for each value of S!) Statistical fluctuations may cause the channel to drift into a saturation state—the channel is filled up with collisions resulting in zero throughput. For an unstable channel, equilibruim conditions assumed earlier exist only for a finite period of time before channel saturation occurs. A Markov chain formulation of the infinite population slotted ALOHA model shows that it is always unstable, in the sense that a stationary probability distribution does not exist [LAM 74, KLEI 75a, FAYO 77].

Fortunately, N must be finite in a real network. In this case, slotted ALOHA channels may exhibit stable or unstable behavior depending upon the parameters N, $\bar{k}$ and S (channel input rate). In [LAM 74, KLEI 74, KLEI 75a] a coherent theory of channel behavior is formulated. Specifically, a method for characterizing stable and unstable channels and a quantitative measure of instability for unstable channels are introduced. A theoretical treatment of adaptive control of unstable channels using a Markov decision model can be found in [LAM 75a]. Various heuristic control algorithms and their performance are presented in [LAM 75b, LAM 79a]. Several effective feedback control algorithms are proposed in [GERL 77a]. Simulations showed that these techniques are effective means of achieving stability, for initially unstable channels, at the expense of a small amount of delay-throughput performance degradation relative to lower bound values.

Control algorithms for preventing channel saturation are based upon one or both of the following mechanisms: (i) reducing the probability of retransmitting a collided packet in a time slot (thus increasing the effective $\bar{k}$), and (ii) revoking the access right of some users for a period of time (thus reducing the effective N). These two mechanisms have been referred to as retransmission control and input control respectively [LAM 74, LAM 75a].

Recall that the slotted ALOHA channel throughput S is maximized when the channel traffic rate is one. The goal of most adaptive control algorithms is to achieve the $G = 1$ condition in the channel. The main difficulty is the acquisition of global network status information by individual users. If,for example, the total number n of ready users can be made known to individual users instantaneously, then an adaptive strategy for realizing the $G=1$ condition is to have each ready user transmit into the next time slot with probability $1/n$ (provided that $n \geq 1$).

Next we observe that the delay-throughput performance of contention protocols is not entirely independent of N as we said earlier. When N is increased (accompanied by a decrease in β), slotted ALOHA channels tend to be less stable [LAM 74, KLEI 75a], which impacts the delay-throughput performance. However, this is only a second order effect.

We have discussed the stability problem and adaptive control techniques within the context of slotted ALOHA, which is a pure contention protocol. However, the stability problem is present in all contention-based protocols (many to be described in this section as well as the next section on reservation protocols) and must not be forgotten.

<u>The R-ALOHA protocol</u> [CROW 73, LAM 78b, LAM 80c]

The traffic environement suitable for ALOHA and slotted ALOHA is that of a large population of low-rate bursty users with short messages (one packet per message). The R-ALOHA protocol was originally proposed by Crowther et al. [CROW 73] and is suitable for users who generate long multipacket messages or users with steady input traffic and queueing capability.

The R-ALOHA protocol requires, in addition to time slotting, that time slots are organized into frames. Time slots are identified by their positions in the frame. The duration of a frame must be greater than the maximum channel propagation time between any two users in the network. (This is of concern in practice only for a satellite channel.) Consequently each user is aware of the usage status of time slots in the previous frame. The network operates without any central control but requires each user to execute the same set of rules for transmitting packets into a time slot depending upon the outcome in the same time slot of the previous frame.

A time slot in the previous frame is <u>unused</u> if it was empty or contained a collision. Slots unused in the previous frame are available for contention by ready users in exactly the same manner as slotted ALOHA. A slot which had a successful transmission by a user in the previous frame is <u>used</u> and is reserved for the same user in the current frame. As a result, such a user now has the equivalent of an assigned TDMA channel for as long as he has traffic to send in it.

We further differentiate between two slightly different protocols depending upon whether an end-of-use flag is included in the header of the last packet before a user gives up his reserved slot:

(P1) end-of-use flag not included, and

(P2) end-of-use flag included.

Under (P1), a time slot is always wasted when a user gives up his reserved slot. The tradeoff for adopting (P2) is some additional packet processing overhead by each user.

Under the assumption of equilibrium conditions, the channel throughput S_{RA} of R-ALOHA can be expressed in terms of the slotted ALOHA throughput S_{SA} for the contention portion of the channel [LAM 80c]

$$S_{RA} = \frac{S_{SA}}{S_{SA} + \frac{1}{\bar{v}}} \qquad \text{under (P1)}$$

or

$$S_{RA} = \frac{S_{SA}}{S_{SA} + \frac{1 - S_{SA}}{\bar{v}}} \qquad \text{under (P2)} \tag{14}$$

where

$\bar{v}$ = average number of packets that a user transmits before he gives up a reserved slot.

Note that S_{RA} is a monotonic function of S_{SA} so that the maximum channel throughput of R-ALOHA is

$$C_{RA} = \frac{C_{SA}}{C_{SA} + \frac{1}{\bar{v}}} \qquad \text{under (P1)}$$

or

$$C_{RA} = \frac{C_{SA}}{C_{SA} + \frac{1 - C_{SA}}{\bar{v}}} \qquad \text{under (P2)}$$

Assuming that $C_{SA} = 1/e$, C_{RA} is plotted in Fig. 6 as a function of $\bar{v}$. Note that $\bar{v}$ ranges from 1 to infinity; thus, we have

$$\frac{1}{1+e} \leq C_{RA} \leq 1 \qquad \text{under (P1)}$$

or

$$\frac{1}{e} \leq C_{RA} \leq 1 \qquad \text{under (P2)}$$

Notice that the R-ALOHA protocol adapts itself to the nature of the traffic. At one extreme, it behaves like slotted ALOHA when the users are bursty ($\bar{v} = 1$). At the other extreme, the channel throughput of R-ALOHA approaches one in the $\bar{v} \rightarrow \infty$ limit; this is the case for high-rate users who can accommodate very long queues. The reader is referred to [LAM 78b, LAM 79a, LAM 80c] for a queueing and message delay analysis of R-ALOHA as well as simulation results.

<u>CSMA protocols</u> [TOBA 74, KLEI 75b, METC 76, HANS 77, LAM 79b, LAM 80a, TOBA 79]

For broadcast channels with a <u>short propagation delay</u>, collisions in the channel can be significantly reduced by requiring each user to sense the channel for the presence of any ongoing transmission before accessing it, hence the name, Carrier Sense Multiple Access (CSMA). A ready user (user with data to send) transmits his data into the channel only if the channel has been sensed idle. Let τ seconds be the amount of time from the start of transmission by one user to when all users sense the presence of this transmission. It is equal to the maximum propagation delay between two users in the network plus carrier detection time. (The latter depends upon the modulation technique and channel bandwidth. It was found to be neglibible relative to the propagation delay in some cases [KLEI 75b].)

Given that a user sensed the channel to be idle and began a transmission, the vulnerable period of that transmission to collisions is 2τ seconds (see Fig. 7). In a short propagation delay environment, this vulnerable period is significantly smaller than those of ALOHA and slotted ALOHA. The channel may also be slotted into minislots of duration τ seconds each, thereby reducing the vulnerable period further to just τ seconds. (Note, however, that τ is the absolute minimum time slot duration. A larger value is necessary in practice. The reader is referred to [ELLI 73, GARD 80] for a discussion of some techniques for time synchronizing distributed users.)

Suppose $\bar{x}$ is the average duration of a successful transmission. The maximum channel throughput of a CSMA protocol depends strongly upon the ratio

$$\alpha = \tau/\bar{x}$$

Specifically, we find out below that C_{CSMA} goes up to one if α is decreased to zero (by either letting $\tau \downarrow 0$ or $\bar{x} \uparrow \infty$; see below).

CSMA protocols have been studied extensively in the past within a packet radio network environment by Kleinrock and Tobagi [TOBA 74, KLEI 75b] and later by Hansen and Schwartz [HANS 77]. CSMA protocols were also implemented and analyzed within a multipoint cable network environment [METC 76, WEST 78, LAM 79b, LAM 80a, TOBA 79]. The main difference between the two environments is as follows. In cable networks, collisions in the broadcast channel can be easily detected. Thus, users involved in a collision can abort their transmissions immediately upon detecting the collision. Mechanisms for detecting collisions and aborting transmissions have been implemented in at least two networks [METC 76, WEST 78]. It appears, however, that this "collision abort" capability is not as easily implementable in packet radio networks of interest in [TOBA 74, KLEI 75b].

We present next the CSMA protocol defined and analyzed in [LAM 80a] This particular version of CSMA clearly shows the relationship of CSMA to the slotted ALOHA protocol (in pretty much the same manner as R-ALOHA is related to slotted ALOHA). The analysis of this CSMA protocol is fairly complete. In addition to maximum channel throughput, explicit formulas for average message delay as well as the moment generating function of the number of ready users are available.

A description of the protocol follows; see also Fig. 8. Network users are time synchronized so that following each successful transmission, the channel is slotted in time. In order to implement the collision abort capability discussed above, the minimum duration of a time slot is 2τ so that within a time slot if a collision is detected and the collided transmissions are aborted immediately, the channel will be clear of any transmission at the beginning of the next time slot. (In practice the duration of a time slot needs to be larger than 2τ. The slotted channel assumption is made mainly to facilitate the analysis. In a real network, either a slotted or unslotted channel may be implemented.) The protocol is defined by the two possible courses of action by ready users:

(C1) Following a successful transmission, each ready user transmits with probability one into the next time slot.

(C2) Upon detection of a collision, each ready user exercises an adaptive algorithm for selecting its transmission probability (less than one) in the next time slot.

The adaptive algorithm in (C2) is not specified. However it should clear at this time that the contention problem in the minislots is exactly the slotted ALOHA problem. An adaptive control algorithm (such as those considered for slotted ALOHA) is needed to guarantee that a successful transmission occurs within a finite number of slots following a collision. The first successful transmission terminates the contention period. (See Fig. 8.)

The maximum channel throughput of the CSMA protocol defined can be obtained as a function of the equilibrium slotted ALOHA throughput S_{SA}. Let C_{SA} be the maximum achievable value of S_{SA}, it is shown in [LAM 80a] that

$$C_{CSMA} = \frac{C_{SA}}{2\alpha + C_{SA}(1+\alpha)} \tag{15}$$

We note that C_{CSMA} is equal to one in the $\alpha \downarrow 0$ limit. Recall that $\alpha = \tau/\bar{x}$. In both ground radio and cable environments, τ is typically very short, say, 0.001 to 0.1 of a packet transmission time. Furthermore, if we consider users who are capable of accommodating long queues, so that $\bar{x}$ is now the average time to empty a user's queue instead of a single packet transmission time, the channel throughput will be one in the $\bar{x} \to \infty$ limit.

The CSMA protocol defined above has the desirable property that when the channel is lightly utilized, the delay in identifying and assigning channel access to a ready user is extremely short and is independent of N (unlike polling and probing considered earlier also for a short propagation delay environment). In particular, when there is exactly one ready user, the delay is zero.

In Fig. 9, we plot the fraction of transmissions that incur zero delay in gaining channel access (given that the channel is free) as a function of α and channel throughput. These are analytic results obtained with $S_{SA} = 1/e$ in [LAM 80a] where a queueing and message delay analysis of the above CSMA protocol can be found. A comparison of the delay-throughput performance of CSMA and polling is shown in Fig. 14 and will be discussed in Section 4 below. The stability problem and adaptive control of CSMA have also been addressed [TOBA 77, HANS 77]. An alternative method of solution to these problems is to view the contention periods as a slotted ALOHA channel; this is the approach taken here as well as in [METC 76].

The URN protocol [KLEI 78, YEMI 78]

The slotted ALOHA protocol is an important component of both R-ALOHA and CSMA protocols. We also learned earlier that the slotted ALOHA protocol needs to be adaptively controlled. Furthermore, the goal of most adaptive control algorithms is to achieve the $G = 1$ condition in the channel. Let us assume for the moment that the number n of ready users at any time is in fact known to individual users instantaneously. (We shall discuss how to estimate n later.) Then an adaptive strategy for achieving the $G = 1$ condition is to have each ready user transmit into the next time slot with probability $1/n$, where $n \geq 1$.

Kleinrock and Yemini [KLEI 78] proposed an alternative "pure" strategy for ready users to determine whether or not to transmit in the next time slot: the probability of transmission is either 1 or 0. In other words, some users have full channel access rights while others have none. (Consequently, the URN protocol is said to be asymmetric.) A user who has channel access right and is also ready transmits into the next time slot. It is possible to prove that optimal strategies are always pure strategies and therefore asymmetric [YEMI 78].

The URN protocol is described using the following URN model. Consider each user as a colored ball in an urn: black for ready, white for not ready. The access protocol is essentially a rule to sample balls from the urn. Let k be the number of balls drawn from the urn. The probability of a successful transmission (throughput) is that of getting exactly one black ball in the sample. This probability is

$$\text{Prob [throughput]} = \frac{\binom{n}{1}\binom{N-n}{k-1}}{\binom{N}{k}} \tag{16}$$

where N is the number of balls, n is the number of black balls. The above expression is maximized when $k = \lfloor N/n \rfloor$ where $\lfloor x \rfloor$ gives the integer part of x. Not only does this value of k maximize the probability of selecting exactly one black ball, but it also gives that the average number of black balls selected is equal to one (i.e. G = 1).

The URN protocol adapts smoothly to network load fluctuations. When the network is lightly loaded, a large number of users get channel access rights. For instance, n = 1 gives rise to k = N; all users get access rights, but only one (the lone ready user) is going to make use of it. As the network load increases, n increases and the number k of users getting access rights is reduced. When $n > \frac{N}{2}$ then k = 1 and the URN protocol becomes effectively a TDMA protocol (which is most suitable for a heavy load). The maximum channel throughput of URN is thus unity. In Fig. 10, simulated delay-throughput results of URN obtained in [KLEI 78] are shown illustrating the desirable traffic adaptivity of the protocol.

Two questions arise in the implementation of the URN protocol: How does an individual user obtain the up-to-date value of n? How does the protocol obtain coordination of the distributed decisions of individual users?

A solution for estimating n with high accuracy at the expense of a small overhead is proposed in the references. Briefly, it consists of a binary erasure reservation subchannel. An idle user who becomes ready (n increases by 1) sends a message of a few bits in the subchannel. When a ready user turns idle (n decreases by 1), the condition is detected by other users from examining his last packet or its positive acknowledgment in the broadcast channel. An erasure (collision) in the subchannel means two or more users become ready in the same time slot. In this case, the increase in n is assumed to be two (an approximation). The resulting error in the estimate of n was found to be neglibible since the probability of more than two users becoming ready in the same time slot is very small. Furthermore, the estimate of n is corrected every time the network goes idle (n = 0). Other heuristic algorithms for estimating n can be found in [LAM 74, LAM 75a, LAM 75b].

Implementation of the URN protocol also must insure that individual users agree upon k the number of users with access rights as well as their identity. The optimal $k = \lfloor N/n \rfloor$ to be used is determined by each user from n estimated as described above. The selection of which k users should get access rights may be achieved via identical pseudorandom number generators at individual users, or via a window mechanism as well as other methods. The reader should consult the references for details.

A tree algorithm for contention resolution [CAPE 77, CAPE 79]

A tree algorithm for scheduling retransmission of packets involved in a collision was proposed by Capetanakis [CAPE 77]. A very desirable property of the algorithm is channel stability . The key idea of the algorithm is similar to that of adaptive polling considered earlier. The difference between the two is in implementation. Adaptive polling relies on a central controller probing passive users for status information. On the other hand, implementation of the tree algorithm is distributed thus requiring users to be able to observe outcomes in the broadcast channel and make decisions. To do so, the channel needs to be time slotted. Furthermore, each algorithm step requires at least a channel propagation time plus two time slots to execute. The channel propagation time in each algorithm step is necessary to enable users to find out the outcomes of the previous algorithm step (see Fig. 11). For a satellite channel, some special technique is needed to avoid wasting the large propagation time interval between slot pairs. One method is to time multiplex the channel into subchannels which can be used independently by different user populations. For example, if P/C is the duration of a time slot, R is the channel propagation time, and m is an integer such that

$$2m(P/C) > R$$

then the satellite channel can be time multiplexed into $m + 1$ subchannels, each accessed by a population of users with the tree algorithm. For a high-speed satellite channel (m is large), much of the benefit from statistically averaging user demands discussed in Section 1.2 will be lost. Another idea suggested in the reference is to use a tree algorithm on $1/(m+1)$ of the channel to make reservations for the other $m/(m+1)$ of the channel. (See reservation protocols below.)

The binary tree algorithm is next described. Suppose that each user corresponds to a leaf in a binary tree, such as illustrated in Fig. 2 earlier (in the context of probing). Each user handles at most one packet at a time. The number of users can be finite or infinite. An infinite user population corresponds to a Poisson source of packet arrivals;

in this case, the binary tree extends to infinity. Time slots in the channel are paired into odd and even slots as shown in Fig. 11. To facilitate description of the algorithm, a last-in-first-out stack is assumed. (It is obviously not necessary.) Each algorithm step consists of the following three action steps. Initially the stack is empty and step (3) is executed.

(1) Remove the binary tree (or subtree) from top of stack and divide it into 2 subtrees. Users with access rights in the first subtree transmit in the odd slot; users with access rights in the second subtree transmit in the even slot.

(2) Observe outcomes in the two slots. For each slot, discard the subtree if the slot is empty or contains a successful transmission; put the subtree back on stack if the slot contains a collision.

(3) If stack is empty, give access rights to all new packets that have arrived in the meantime, put the entire tree on stack and go to (1); else go to (1).

Note that new packet arrivals are only given access rights when the stack empties, which marks the end of one "epoch" and the beginning of the next. Arrivals during one epoch must wait to be served in the next epoch.

We make two observations. First, the objective of the algorithm is to divide the population of users into partitions each containing at most one contending user. Therefore, the order of the tree search is not important. (In other words, the stack does not have to be last-in-first-out). Second, the assignment of binary addresses to users does not have to be predetermined. Specifically, following a collision, a contending user may with equal probability join any one of the two resulting subtrees. This random addressing scheme is equivalent to allowing each of the contending