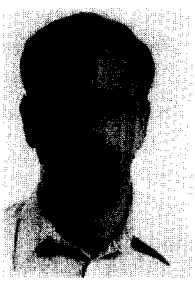
"Supernodes" in Networks Employing Spread Spectrum with Code Division Multiple Access *

Yitzhak BIRK
Computer Science Department, IBM Research Division, Almaden Research Center, 650 Harry Road., San Jose, CA 95120, U.S.A.

Fouad A. TOBAGI
Computer Systems Laboratory, Electrical Engineering Department, Stanford University, Stanford, CA 94305, U.S.A.

A spread-spectrum channel can accommodate several concurrent successful transmissions, and a single-transceiver node can thus utilize only a small fraction of the channel's capacity. As a result, the maximum network throughput is much lower than this capacity whenever a single node, such as a gateway or a file server, must carry a large fraction of the traffic. In order to allocate the appropriate fraction of capacity to a "busy" node, we propose to equip it with several transmitters and receivers, thereby turning it into a "supernode". Several architectures and operation policies for supernodes are suggested and compared. It is shown, for example, that an M-receiver supernode can significantly outperform M independent conventional nodes. In a slotted system with packet lengths of one slot, this is achieved by special routing of the supernode's inbound traffic. In an unslotted system, it is achieved by appropriate code assignment policies. Packet-radio networks with half-duplex nodes, as well as networks with full-duplex nodes, are considered.

Keywords: Communications Networks, Packet Radio Networks, Multiaccess Protocols, Channel Access Protocols, Spread Spectrum Multiple Access, Code Division Multiple Access, Multi-transceiver Stations.



Yitzhak Birk received the B.Sc. (cum laude) and M.Sc. degrees from the Technion – Israel Institute of Technology in 1975 and 1981, respectively, and a Ph.D. degree from Stanford University in 1987, all in Electrical Engineering. His doctoral dissertation was entitled "Concurrent Communication among Multi-Transceiver Stations over Shared Media".

From 1976 to 1980, he was project engineer in the Israel Defense Forces. In December 1986, he joined the International Business Machines Corporation and is presently a Research Staff Member in the Computer Science Department at IBM's Almaden Research Center in San Jose, California. His current research interests include computer communications, distributed systems and multiprocessor architectures. Dr. Birk is a member of the Institute of Electrical and Electronics Engineers.

* Portions of this paper were presented at INFOCOM '86, Miami, F.L. This work was supported in part by the Defence Advanced Research Projects Agency under contract MDA903-84-K-0249, monitored by ONR. Yitzhak Birk was supported by an IBM graduate student fellowship.

Fouad A. Tobagi received the Engineering Degree from Ecole Centrale des Arts et Manufactures, Paris, France, in 1970 and the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles, in 1971 and 1974, respectively.

From 1974 to 1978, he was a Research Staff Project Manager with the ARPA project at the Computer Science Department, University of California, Los Angeles, and engaged in modeling, analysis, and measurements of packet radio systems. In June 1978, he joined the faculty of the School of Engineering at Stanford University, Stanford, CA, where he is currently Professor of Electrical Engineering. His current research interests include packet switching in ground radio and satellite networks, high-speed fiber optics local area networks, integrated services broadband networks, modeling and performance evaluation of computer communications systems, and VLSI implementation of network components.

Dr. Tobagi is a Fellow of the Institute of Electronics and Electrical Engineers for his contributions to the field of Computer Communications and Local Area Networks. He is the winner of the 1981 Leonard G. Abraham Prize Paper Award in the field of Communications Systems for his paper "*Multiaccess Protocols in Packet Communications Networks*" and co-winner of the IEEE Communications Society 1984 Magazine Prize Paper Award for the paper "*Packet Radio and Satellite Networks*". He has served as Associate Editor for Computer Communications in the IEEE Transactions on Communications for the period 1984–1986, and Editor for Packet Radio and Satellite Networks in the Journal of Telecommunications Networks for the period 1981–1985. He was Co-editor of the special issue on Local Area Networks of the IEEE Journal on Selected Areas in Communications (November 1983) and the special issue on Packet Radio Networks of the Proceedings of the IEEE (January 1987). He recently co-edited *Advances in Local Areas Networks*, a book in the Series *Frontiers in Communications* published by the IEEE Press. Dr. Tobagi is also a member of the Association for Computing Machinery and has served as an ACM National Lecturer for the period 1982–1983.

North-Holland
Computer Networks and ISDN Systems 15 (1988) 341–357

1. Introduction

1.1. Concurrency in Communication over Spread-Spectrum Channels

In addition to improving immunity to noise and jamming, the use of spread spectrum also permits several concurrent transmissions to be received by collocated receivers. One manifestation of this is *time capture*; by using a spread-spectrum code with narrow (temporal) autocorrelation mainlobes and low sidelobes, replicas of the same code which are staggered in time are nearly orthogonal to each other, permitting several staggered transmissions on the same code to be received concurrently by different receivers. Concurrency can also be attained by *code-division multiple-access*, or CDMA, i.e. by using different, mutually orthogonal codes for different transmissions [1,2].

1.2. The Problem in Allocating a Spread-Spectrum Channel's Capacity

In a real network, certain nodes must often carry much more traffic than most other nodes; examples of such nodes are gateways and file servers in terrestrial networks, as well as the terrestrial hub of a 2-hop satellite network [3]. To make use of a network's capacity, a "busy" node must therefore receive a fraction of channel capacity which is much larger than those given to most other nodes.

The fact that the spread-spectrum channel can accommodate several ongoing transmissions complicates the task of nonuniform capacity allocation, since a single transmission uses only a fraction of the channel capacity. Assuming the use of standard equipment and codes of equal information-theoretic rates for all transmissions, extreme nonuniformity in capacity allocation can therefore only be achieved by equipping a "busy" node with several transmitters and receivers, thereby permitting it to engage in concurrent transmissions or receptions. Such a node will be referred to as a "supernode".

1.3. Goals of this Work

The purpose of this paper is to explore the potential performance advantages of a supernode with M transmitters and receivers over a collection of M independent, collocated conventional nodes. Since the problem in allocating capacity is common to all spread-spectrum channels, the emphasis here is on identifying and understanding issues which are valid regardless of the exact spread-spectrum channel characteristics. This is different from most spread-spectrum research, which has been directed at the detailed understanding of the spread-spectrum channel (e.g. [1,2]).

Specifically, we consider a single supernode, $\mathscr{S}$, which is surrounded by many conventional nodes, each of which carries a small fraction of the network traffic. The design goal is to increase $\mathscr{S}$'s throughput by proper architecture and operation. (Since $\mathscr{S}$ is assumed to constitute a throughput bottleneck, maximizing its throughput also maximizes the network throughput.) Also, for any given inbound throughput, it is desirable to maximize the efficiency of channel usage, which is defined to be the reciprocal of the number of times a packet must be transmitted until it is received successfully by its destination.

1.4. Outline of the Paper

In Section 2, we present a model for packet reception in the CDMA environment. In Section 3, we consider one of $\mathscr{S}$'s receivers; it is shown that in a slotted system with packet lengths of exactly one slot, routing all the traffic destined for this receiver via a subset of $\mathscr{S}$'s neighbors can substantially increase its inbound throughput. This is referred to as *link masking*, since the communication links from the other neighbors to $\mathscr{S}$ are effectively masked.

Section 4 addresses the design and operation of a supernode with multiple transmitters and receivers. We begin by summarizing results that were obtained in [4] for a slotted system, but are nevertheless valid in unslotted systems as well. Next, an unslotted model is used to further study the design and operation of $\mathscr{S}$'s receivers, focusing on time capture and on the resulting design tradeoffs. Several multi-receiver node architectures and code-assignment policies are proposed and compared. Since the most common use of spread-spectrum channels is currently in packet-radio networks [5], the analysis in Sections 3 and 4 assumes that a node cannot receive and transmit

concurrently (half duplex). In Section 5, the results of Sections 3 and 4 are adapted to the case of full duplex nodes, which applies to the use of spread-spectrum in local-area networks over low-attenuation media or in packet radio networks with transmitters and receivers that are not collocated. Section 6 summarizes the paper.

2. Model for Packet-Reception

A packet consists of two fields: (i) *preamble* of fixed length, and (ii) *data*. The reception of a packet consists of two phases:

(i) synchronization onto the preamble, and

(ii) reception of the data portion.

In the spread-spectrum environment, occasional contamination of the received signal is possible. Therefore, the data portion of the packet is sometimes encoded prior to transmission using *forward error correction* codes, or FEC [6]. Upon completion of its reception, a received packet is decoded by the recipient. If it is error-free, the reception is considered successful; otherwise, the packet is rejected. Packets that are not received successfully are lost and must be retransmitted at a later time. To facilitate analysis, we will distinguish between *raw throughput*, consisting of all received packets, and *error-free throughput*, consisting only of those packets that are received successfully. The latter is the true throughput.

The synchronization phase is successful if and only if

(i) the receiving node is not transmitting,

(ii) the arriving packet is *receivable*, i.e., its preamble does not overlap (at $\mathcal{S}$) with that of another packet that was transmitted on the same code, and

(iii) there is an available receiver on the appropriate code. (A receiver can only await packets on a single code at any instant.)

The phenomenon of overlapping preambles of packets with the same code will be referred to as *intracode interference*. Figure 1 shows an example of a packet-arrival stream. Note that an unsuccessful attempt to synchronize onto a nonreceivable packet does not prevent the receiver from synchronizing onto the next receivable packet, since the preamble of a receivable packet never overlaps with that of a nonreceivable one.

The finite capacity of the channel results in interference which depends primarily on the number of ongoing transmissions and is largely independent of code; this will be referred to as *intercode interference*. Intercode interference is assumed to manifest itself only in the form of erroneous bits in received packets, thus rendering those receptions unsuccessful; it cannot cause a receiver to abort an ongoing reception. From this, along with an assumption that a node never begins transmitting while engaged in data-reception, it follows that the data-reception phase begins upon successful completion of the synchronization phase, and is always completed.

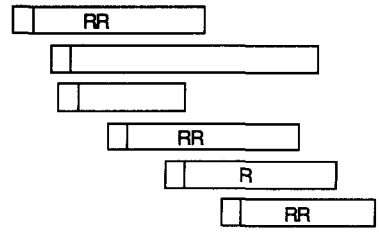


Fig. 1. Example of a packet arrival process. We assume that all packets are transmitted using a common code, and that there are two receivers. Packets marked "RR" are received; those marked "R" are receivable but cannot find an available receiver, and the unmarked ones are nonreceivable.

With this model, throughput analysis can be carried out in two stages. Initially, the raw throughput is computed. This stage accounts for the loss of packets due to preamble-overlap on the same code or to receiver unavailability. Both of these depend on the architecture, on the code assignment policy and on the level of $\mathcal{S}$'s inbound traffic, but not on channel parameters such as coding scheme, signal-to-noise ratio and capacity, or on the level of background traffic. The second stage accounts for the remaining cause for packet loss, namely erroneous bits due to intercode interference, which depends almost solely on the total traffic level and on channel parameters, and yields the error-free throughput. This approach decouples the architecture-dependent factors from the channel-dependent ones, thus permitting our raw-throughput results to be used in conjunction with intercode-interference results obtained (by others) for different coding schemes, levels of background traffic, etc.

The above model is an approximate one. We next provide some insight into the approximations and the consequences of using them.

Synchronization. In practice, the synchronization pattern is repeated several times in the pre-

amble. Therefore, partial overlap of preambles with the same code may still permit synchronization onto them, and the model used here is thus somewhat pessimistic. Given a specific preamble design, our model can be used in a more accurate way by replacing the true preamble length with an appropriately shorter one. This has the desired effect of reducing the probability of destructive preamble overlap for any given arrival rate. Another approximation involves the implicit assumption that inter-code interference does not affect the synchronization. The logic behind this approximation is that if the level of inter-code interference is such that a short, robust preamble is interfered with in a significant way, the probability of no errors in a received packet is very low, and such operating conditions are thus of very little interest.

Decoupling of error-freedom from reception. Let us consider the probability that a packet is error-free; i.e., the probability that if there were an infinite number of receivers, as well as some magic way of guaranteeing synchronization, the packet would be received successfully. Due to the Markovian nature of the system, the only dependence of this probability on the history of the system is through the number of ongoing transmissions at the time of arrival of the packet. Using k to denote this number, it follows that

$$P[\text{error-free} \mid k, \text{received}] = P[\text{error-free} \mid k]. \quad (1)$$

Also,

$$P[\text{error-free}] = \sum_{m=0}^{\infty} P[k = m] \cdot P[\text{error-free} \mid m]. \quad (2)$$

The probability that a packet is received and is error-free (with a finite number of receivers) can always be expressed as

$$P[\text{successful reception}] = P[\text{received}] \cdot P[\text{error-free} \mid \text{received}]. \quad (3)$$

The first term on the right-hand side of (3) is equal to the ratio of the raw throughput to the mean packet-arrival rate; from (1) it follows that the second one can be expressed as

$$P[\text{error-free} \mid \text{received}] = \sum_{m=0}^{\infty} P[k = m \mid \text{received}] \cdot P[\text{error-free} \mid m]. \quad (4)$$

Therefore, the decoupling approximation is close if and only if the knowledge that a packet was received has little effect on the distribution of the number of ongoing transmissions at the time of the packet's arrival. Furthermore, a very good first-order correction can be obtained by using the value of P[error-free] which corresponds to the correct value of the mean number of ongoing transmissions. This will be evaluated for a specific case and further elaborated upon in a later section.

3. Link Masking

In this section, we explore the funneling of all the inbound traffic destined for any given receiver of $\mathcal{S}$ through a subset of $\mathcal{S}$'s neighbors (*authorized neighbors* for that receiver) as a means of increasing its inbound throughput. $\mathcal{S}$'s remaining inbound links are thus masked. We assume a slotted system, with packet lengths of exactly one slot. (Link masking is particularly relevant to such slotted systems, since they do not benefit from time capture.) Recalling that the throughput of a conventional Slotted ALOHA [7] channel is $1/e$ for an infinite population and 0.5 for a population of 2, the funneling can potentially increase inbound throughput by up to 36%. To prevent obstruction of the main issue at hand, we consider a single receiver and infinite channel capacity. The accommodation of multiple receivers is straightforward, provided that N, the number of neighbors, satisfies $N \geqslant 2M$; otherwise, it is slightly more complicated due to an overlap of the funnels for different receivers. The accommodation of finite channel capacity is discussed in [4]; the relationship between raw and error-free throughput will be commented upon.

Viewing a network as a graph whose nodes correspond to network nodes, there is a link from node i to node j with tag k if and only if j can hear i's transmissions and has a receiver on code k. In networks employing CDMA with Receiver-Directed Codes (CDMA/RDC), whereby nodes are allocated disjoint sets of codes for reception, each outbound link of any given node has a distinct tag. Each transmission therefore activates only one link, and it is thus possible to *mask individual links* of the graph. This is different from narrowband networks, in which the decision as to

whether or not to mask links applies jointly to all of a node's outgoing links.

Let us define the *routing graph* to be the directed graph consisting of the union of the paths to be used for the routing of packets from each node to $\mathcal{S}$, but excluding the initial hop of those paths. The goal is to determine the maximum attainable throughput into $\mathcal{S}$ and the simplest routing graph that can achieve it. (Minimum number of hops.) Since throughput with slotted ALOHA increases as the size of the contending population decreases, each node in the simplest routing graph should transmit to exactly one other node. Combining this with the requirement that all paths of the routing graph end at $\mathcal{S}$, it follows that the simplest routing graph is a tree which has $\mathcal{S}$ as its root.

Our analysis of link masking is similar to that of routing packets to a central node in a narrowband network via a sequence of repeaters [8–10]. The main differences are:

(i) we take into account $\mathcal{S}$'s nonzero probability of transmission ($p_0 \geq 0$) and its consequent unavailability for reception, whereas in the referenced studies the central node was assumed to never transmit, and

(ii) we assume the use of receiver-directed codes (CDMA/RDC), allowing a node to receive in the presence of a transmission by its father in the routing tree.

Our analysis was carried out by assuming that the upper bound of $0.5(1-p_0)$ on $\mathcal{S}$'s inbound throughput is achievable, and proceeding to derive the maximal number of nodes in each level of the routing tree along with the optimal value of the probability of transmission at each level. ($\mathcal{S}$ constitutes level 0.) An infinite maximum number of nodes served as the indication for having reached the leaves of the tree. Details of the analysis appear in [4]. To achieve an inbound throughput of $0.5(1-p_0)$, the permitted indegree of level-1 nodes, n_1, must satisfy

$$\left(1-\frac{1}{n_1}\right)^{n_1-1} \geq 0.5(1-p_0). \tag{5}$$

As p_0 decreases below $(1-2/e)$, [1] n_1 drops from infinity to finite values. The two extremes of the minimal routing tree are shown in Fig. 2. Note that the size of the routing tree is independent of the number of network nodes.

Let us now take a closer look at 2-hop link masking (a height-1 binary routing tree); the 1st hop is from the source to an authorized neighbor, and the 2nd hop is from an authorized neighbor to $\mathcal{S}$. When $p_0 < 0.264$, the 1st hop cannot support the maximal throughput of the 2nd hop. When $0.264 \leq p_0 < 1$, the bottleneck is in the 2nd hop and $S_{\text{in}_{\max}} = 0.5(1-p_0)$. The maximum (over the probabilities of transmission of all nodes other than $\mathcal{S}$) is given by

$$S_{\text{in}_{\max}}(p_0) = \begin{cases} \dfrac{2}{e}\left(1-\dfrac{e^{-1}}{1-p_0}\right), & 0 \leq p_0 \leq 0.264, \\ 0.5(1-p_0), & 0.264 < p_0 < 1. \end{cases} \tag{6}$$

A plot of $S_{\text{in}_{\max}}/(1-p_0)$ versus p_0 is shown in Fig. 3; results for direct transmissions and for 3-hop link masking are presented for reference. We see that 2-hop link masking comes close to achieving $S_{\text{in}} = 0.5(1-p_0)$, thus rendering the height-2 binary tree (3-hop link masking) unnecessary. The results of this section depend on the

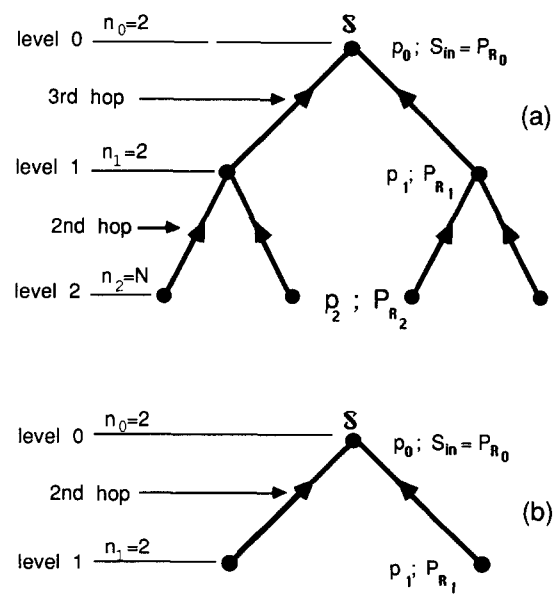


Fig. 2. Link masking: minimal routing trees. (a) $p_0 < (1-2/e)$; (b) $p_0 \geq (1-2/e)$.

[1] $1-2/e = 0.264$.

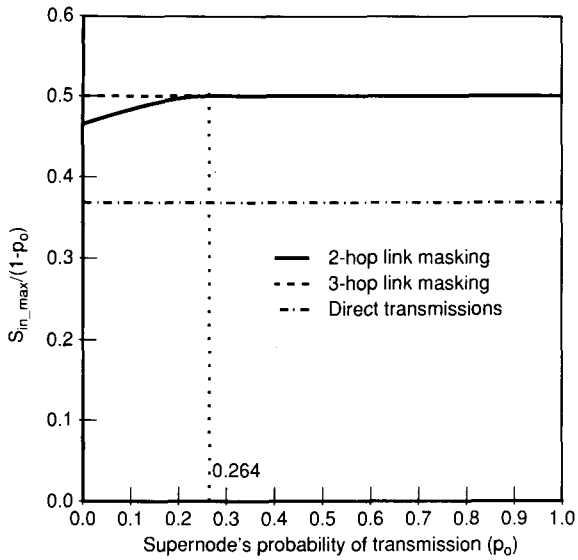


Fig. 3. Link masking: maximum normalized inbound throughput (Slotted). $M = 1$; $N \gg M$.

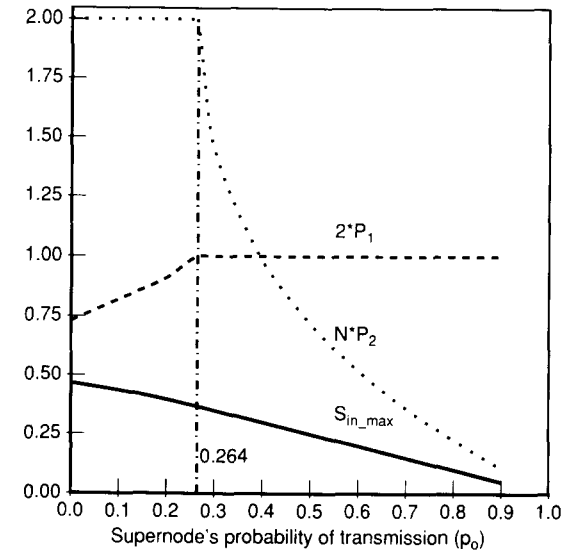


Fig. 4. Maximal inbound throughput and required transmission rates with 2-hop link masking. $M = 1$; $N \gg M$. Observe that the bottleneck moves from the 1st hop to the 2nd one when p_0 becomes greater than $(1 - 2/e)$.

total number of nodes, N, only when it becomes small. [2]

Having observed the throughput advantage of 2-hop link masking over direct transmissions, we next compare them in terms of efficiency. Figure 4 depicts $S_{\text{in}_{\max}}$ for 2-hop link masking, along with the corresponding aggregate transmission rate on each of the hops, as a function of p_0. Figure 5 depicts the efficiency of channel usage as function of inbound throughput for various values of p_0; curves are presented for direct transmissions and for 2-hop link masking. Both S_{in} and the efficiency are divided by $(1 - p_0)$ in order to remove the effect of $\mathscr{S}$'s unavailability for reception due to its own transmissions. We see that with direct transmissions, this "normalized" efficiency is independent of p_0. With 2-hop link masking, however, it increases with an increase in p_0. This is due to the fact that, while the normalized efficiency of the 2nd hop remains constant, the first hop becomes very efficient when it is not the bottleneck. Figure 6 depicts $S_{\text{in}_{\max}}(p_0)$ as a function of p_0 for direct transmissions and for 2-hop link masking. For small values of p_0, direct transmissions are more efficient as long as they are feasible, and the boundary is $S_{\text{in}_{\max}}(p_0$, direct transmissions). However, as p_0 increases, the boundary moves into the feasible domain of direct transmissions. This result contradicts the intuition whereby more transmissions per packet are required with 2-hop routes than with single-hop routes.

We have thus far considered raw throughput. However, as seen in Fig. 4, the aggregate transmis-

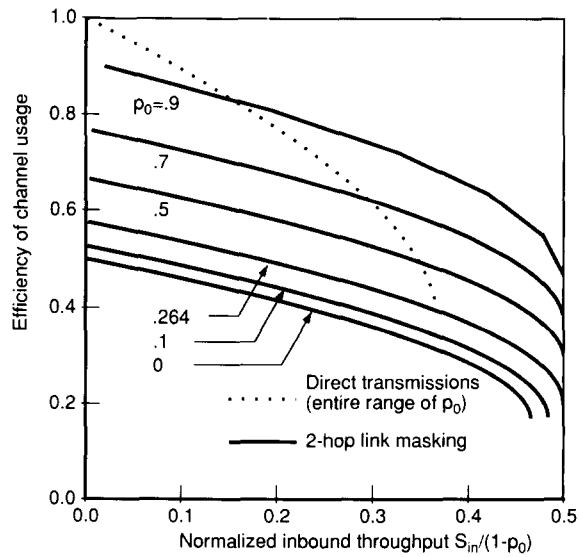


Fig. 5. Efficiency of channel usage with 2-hop link masking and with direct transmissions.

[2] The indicator for the closeness of the approximation in assuming "very large N" is the relative difference between $(1-1/N)^{N-1}$ and $1/e$. For example, the differences for $N = 5$, 10, 20 are 11%, 5.3% and 2.5%, respectively.

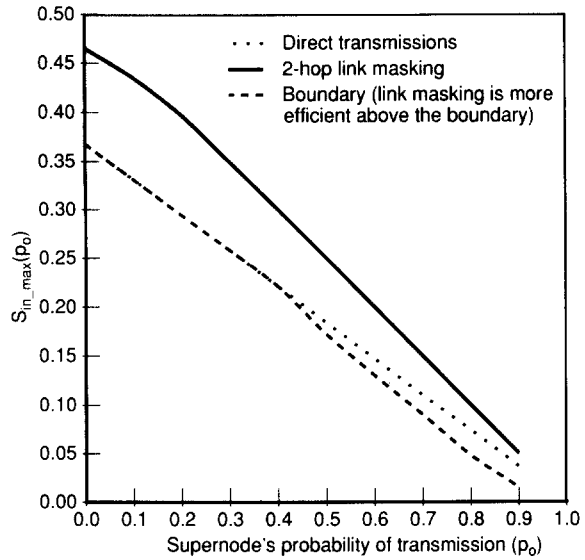


Fig. 6. Feasibility and superiority boundaries for direct transmissions and for 2-hop link masking. $M=1$; $N \gg M$. The feasible (p_0, S_{in}) combinations for the two schemes are represented by the regions under the respective curves. The dashed curve is the equal efficiency line. Below it, direct transmissions are more efficient (fewer transmissions per reception) than 2-hop link masking. Above the boundary, 2-hop link masking is more efficient.

sion rate associated with a single receiver that employs link masking is at most 3, which is typically much smaller than channel capacity. Therefore, if only few receivers mask their inbound links, the effect of intercode interference on the performance of link masking relative to that of direct transmissions is very small, and this is the issue at hand. Intercode interference does limit the number of receivers that may employ link masking efficiently; the limit depends on channel capacity and on the level of background traffic. (Another upper limit is $\frac{1}{3}$ of the number of nodes.)

The protocol required to support 2-hop link masking is very simple and robust; each network node keeps two addresses for $\mathcal{S}$, which are actually the addresses of the two authorized neighbors, and uses either one (at random). This has the additional benefit of balancing the load on the two authorized neighbors.

Link masking should not be used for outbound single-destination traffic, because the probability of reception of a supernode packet by its lightly-loaded destination node is higher than the probability of reception by the busy forwarding node. It may, however, be beneficial for multi-destination packets. $\mathcal{S}$ would transmit such a packet once to one of its neighbors, which would then retransmit it on the code of each of the intended recipients. A similar approach can be taken with acknowledgments for inbound traffic. Since acknowledgements are very short, $\mathcal{S}$ can collect several acknowledgements into a single packet and send them to one of its neighbors, which would then distribute them to their destinations. Unlike the inbound funnel, which consists of two specific neighbors, the outbound funnel can change dynamically. ($\mathcal{S}$ may select an ad-hoc "helper" for each such acknowledgment packet.)

4. Multiple Receivers and Multiple Transmitters

In the previous section, we gave special treatment to $\mathcal{S}$'s traffic in order to increase throughput, making use of lightly-utilized hardware at neighboring nodes. We now turn to the study of a supernode which is equipped with multiple transmitters and receivers. We start out by establishing several guidelines for the design and operation of such a node, and then go on to show how coordinating its receivers can further increase its throughput.

4.1. Design Guidelines

- Increasing the number of receivers increases throughput; however, the benefit of additional receivers tapers off as the channel capacity becomes the bottleneck.
- With a half-duplex node, which cannot receive if any of its transmitters is transmitting, it is best to operate all transmitters concurrently (using different codes). Otherwise, the probability that the node is not blocked for reception diminishes rapidly with an increase in the number of transmitters. We refer to this as *time synchronization*.
- The optimal number of transmitters. Initially, the outbound throughput grows with an increase in the number of supernode transmitters. However, an excessive number of concurrent supernode transmissions decreases throughput due to the increased probability that the total number of ongoing transmissions ($\mathcal{S}$'s as well as others') that can be heard by the intended receiver exceeds the channel's capacity, result-

ing in the destruction of all transmissions. Taken to an extreme, if the number of $\mathcal{S}$'s transmitters exceeds the channel's capacity and they are all operated concurrently, the outbound throughput is zero.

A quantification of these guidelines for a slotted system (no time capture) appears in [4].

4.2. Multiple Receivers with Time-Capture

Let us now consider an unslotted model, which exposes the effect of time capture. The preambles of packets are of fixed length, and the length of the data portion is assumed to follow an exponential distribution. Without loss of generality, the transmission time of a preamble is selected as the unit of time, and the mean transmission time of the data portion is denoted by $1/\mu$.

4.2.1. Network Model (Unslotted)

A single supernode $\mathcal{S}$ is considered; it is equipped with M receivers and is allocated $N_c \leqslant M$ codes for reception. The arrival process of packets with any given code, consisting of new as well as retransmitted packets, is Poisson, and is i.i.d. from code to code. The aggregate arrival rate at $\mathcal{S}$ is λ. The length of arriving packets is assumed to be i.i.d. according to the aforementioned packet-length distribution. To avoid unnecessary complexity of the mathematical derivations, it is assumed that $\mathcal{S}$ never transmits. The accommodation of $\mathcal{S}$'s transmissions is deferred to the end of this section. We now proceed to calculate raw throughput for various architectures and to compare them. (Unlike in the slotted case, the throughput analysis here is carried out in two stages: raw throughput and error-free throughput.)

4.2.2. Fixed-Code-Assignment Architecture (FCAA)

$\mathcal{S}$'s receivers are partitioned into groups of R; [3] all receivers of any given group are permanently assigned the same code. Each group has a controller, which designates one of the idle receivers (if any) to receive the next incoming packet. This architecture is illustrated in Fig. 7. The packet arrival rate to a group is $\lambda' \triangleq \lambda R/M$. We next proceed to derive the raw throughput for FCAA.

Based on the model for packet reception, each group of receivers may be studied separately, considering only its R receivers and only arriving packets with its code. Furthermore, only receivable packets need to be considered. Intuitively, it seems that inbound throughput (S_{in}) should be maximized (over R and λ') by assigning each receiver a unique code ($R = 1$), since such an assignment minimizes the likelihood of intracode interference. We prove that this is indeed the case, by contradiction: assume that S_{in} is maximized by setting $R = R_0$, $R_0 > 1$, and $\lambda = \lambda_0$. Consider now a second system, with $\lambda = R_0\lambda_0$ and $R = 1$. The two systems have the same value of $\lambda' \triangleq \lambda R/M$; consequently, the arrival process of receivable packets to a group is the same in both systems. Since the ongoing reception of a receivable packet cannot be interfered with, it follows that, for any process of receivable-packet arrivals to the group, the throughput of any given receiver in the group is maximized if all the packets are directed to it (as opposed to being shared with other receivers). As a result, the inbound throughput of each receiver in the second system is higher than in the first one; this, in turn, results in a higher aggregate throughput and contradicts the optimality assumption. Note that this result is independent of packet length and of the number of receivers.

For any given inbound throughput S_{in}, the optimal group size R_{opt} is defined to be the value of R that maximizes efficiency. We claim that, for $S_{in} < S_{max}$, there are cases in which $R_{opt} > 1$. To see why, let us interpret R_{opt} as the value of R that maximizes S_{in} for a given value of λ. For given values of λ and M, the probability of receivability is maximized if $R = 1$, because setting $R = 1$ minimizes the arrival rate of packets with any given code. However, such a choice minimizes resource-sharing and therefore maximizes the likelihood of a packet being discarded because all the receivers *in its group* are busy, even if there are idle receivers in other groups. There is hence a design tradeoff in selecting the value of R.

For a fixed preamble length (one), the arrival process of receivable packets is nothing but the departure (reception) process of a Pure ALOHA [11] system with Poisson arrivals (rate λ'), zero

[3] Throughout the discussion, R is assumed to divide M. In practice, the number of receivers (M) is an upper bound on R; also, if the desired value of R does not divide M, groups of different sizes will have to be constructed. Lastly, note that $R = 1$ corresponds to M independent, collocated conventional nodes, since no use is made of the fact that the receivers are collocated.

capture and packets of unit length. These interdeparture times are i.i.d., and the mean rate of departures is $\lambda' e^{-2\lambda'}$. The Laplace transform of the probability density function of the interdeparture-time random variable X was derived by Takagi [12] as

$$X^*(s) = \frac{\lambda' e^{-(s+\lambda')}\left[s + \lambda' e^{-(s+\lambda')}\right]}{s^2 + s\lambda'\left[1 + e^{-(s+\lambda')}\right] + \lambda'^2 e^{-2(s+\lambda')}}. \tag{7}$$

Let $\{Y(t),\ t \geqslant 0\}$ be the random process representing the number of busy receivers at time t. Next, let $\{Y(t_n)\}_{n=1}^{\infty}$ be the embedded process at $\{t_n\}_{n=1}^{\infty}$, where t_n represents the arrival time of the nth receivable packet. We now make a key observation, namely that interarrival times of receivable packets always exceed the preamble length. Since there is no queue, this implies that an arriving receivable packet always finds all the busy receivers in the data-reception phase, whose duration is exponentially distributed, having completed the fixed-duration synchronization phase. Consequently, $Y(t_n)$ constitutes a complete specification of the state of the system at $t = t_n$. Recalling that interarrival times of receivable packets are i.i.d, we conclude that $\{Y(t_n)\}_{n=1}^{\infty}$ is a Markov chain; we conveniently denote it by $\{Y_n\}_{n=1}^{\infty}$. Defining $\Pi = (\pi_0, \pi_1, \pi_2, \ldots, \pi_R)$ to be the steady-state probability vector, the probability of a receivable packet finding an available receiver is simply $(1 - \pi_R)$. It is independent of that packet's length.

To construct the transition probability matrix $P \triangleq [p_{ij}] \triangleq P\{Y_{n+1} = j \mid Y_n = i\}$, let us initially condition on $X \triangleq t_{n+1} - t_n = x$. Assuming that $Y_n = i$, $i < R$, i.e., assuming that the nth receivable packet found a receiver, $(i + 1 - j)$ receivers must complete service in time x in order to have $Y_{n+1} = j$. At the beginning of x, i of the $i + 1$ busy receivers are in the exponentially distributed data-reception phase and the remaining one, which is receiving the nth receivable packet, is at the beginning of the fixed-duration synchronization phase. A slightly different situation occurs when $Y_n = R$: the nth receivable packet is lost, and the number of busy receivers stays R, all of which are in the exponentially distributed data-reception phase of ongoing receptions. The general expression for $p_{ij\mid X}(x)$ is given by

$$p_{ij\mid X}(x) = \begin{cases} [1 - e^{-\mu(x-1)}]\binom{i}{j}[1 - e^{-\mu x}]^{i-j} e^{-\mu x j} \\ \quad + e^{-\mu(x-1)}\binom{i}{j-1}[1 - e^{-\mu x}]^{(i+1)-j} \\ \quad \cdot e^{-\mu x(j-1)}, & 0 < j \leqslant i < R, \\ \binom{i}{j}[1 - e^{-\mu x}]^{i-j} e^{-\mu x j}, & 0 \leqslant j \leqslant i = R, \\ [1 - e^{-\mu(x-1)}][1 - e^{-\mu x}]^i, & j = 0,\ i < R, \\ e^{-\mu(x-1)} e^{-\mu x i}, & j = i + 1,\ i < R, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Relaxing the condition on X by letting

$$p_{ij} = \int_0^{\infty} f_X(x)\, p_{ij\mid X}(x)\, dx,$$

and recalling that $f_X(x) = 0$ for $x < 1$ yields

$$p_{i,j} = \begin{cases} \binom{i}{j}\sum_{m=0}^{i-j}\left[(-1)^m\binom{i-j}{m}\right. \\ \quad \left. \cdot\left[X^*(\mu_{m+j}) - e^{\mu} X^*(\mu_{m+j+1})\right]\right] \\ \quad + e^{\mu}\binom{i}{j-1}\left[\sum_{m=0}^{i+1-j}\left[(-1)^m\binom{i+1-j}{m}\right.\right. \\ \quad \left.\left. \cdot X^*(\mu_{m+j})\right]\right], \\ \qquad 0 < j \leqslant i < R, \\ \binom{i}{j}\sum_{m=0}^{i-j}\left[(-1)^m\binom{i-j}{m} X^*(\mu_{m+j})\right], \\ \qquad j \leqslant i = R, \\ \sum_{m=0}^{i}\left[(-1)^m\binom{i}{m}\left[X^*(\mu_m)\right.\right. \\ \quad \left.\left. - e^{\mu} X^*(\mu_{m+1})\right]\right], \\ \qquad j = 0,\ i < R, \\ e^{\mu} X^*(\mu_j), \quad 0 < j = i + 1 \leqslant R, \\ 0, \quad \text{otherwise,} \end{cases} \tag{9}$$

where $\mu_k \triangleq k\mu$. Since $p_{i,j} = 0$ for $j > i + 1$, the state-probability vector Π is given by the recursive expression

$$\pi_{i-1} = \frac{1}{p_{i-1,i}}\left[\pi_i - \sum_{m=i}^{R} \pi_m p_{m,i}\right], \quad i = 1, 2, \ldots, R \tag{10}$$

along with the constraint that $\sum_{i=0}^{R} \pi_i = 1$.

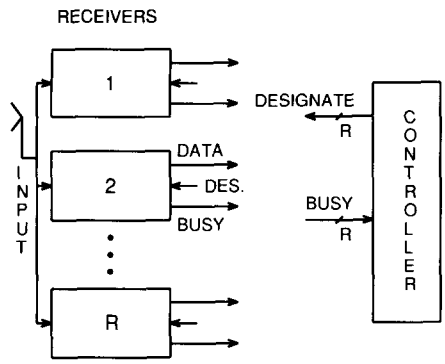


Fig. 7. Fixed code-assignment architecture (FCAA). A single group of R receivers is shown.

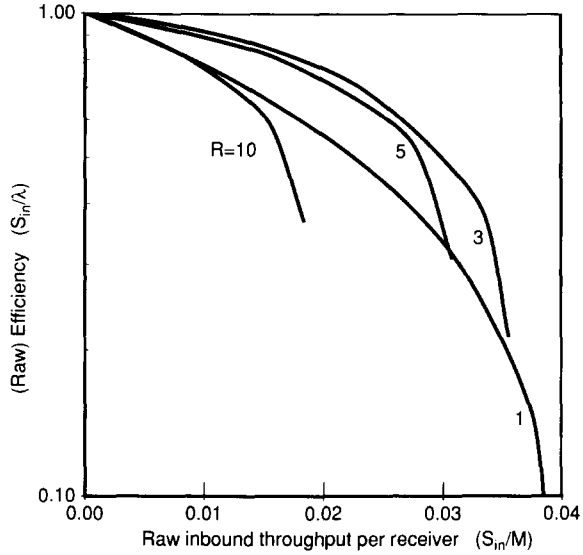
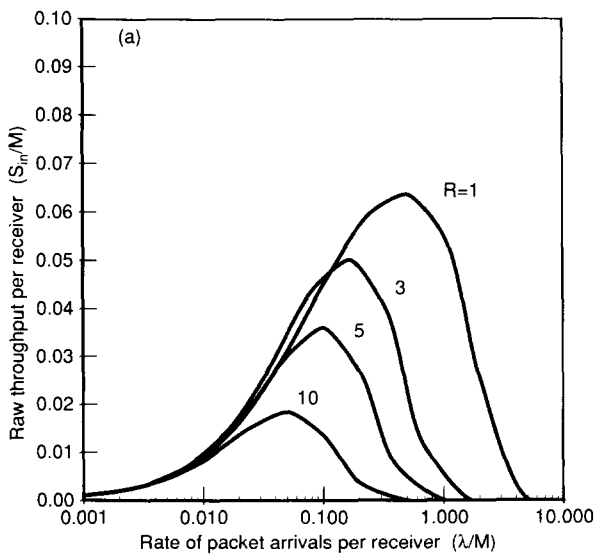


Fig. 9. Efficiency of channel usage with FCAA for various group sizes. $1/\mu = 20$.

Figure 8 shows plots of the inbound throughput per receiver S_{in}/M as a function of the rate of packet arrivals per receiver λ/M for various group sizes R; throughput is expressed in packets per unit time, which has been taken to be the preamble length. Observe that, for any given packet length and arrival rate per receiver, there is an optimal group size; for low arrival rates, R_{opt} is large, since the throughput bottleneck is in finding an available receiver, and the increased resource sharing that is brought about by larger groups is important. As the arrival rate increases, preamble collisions become the limiting factor, and consequently R_{opt} decreases until it eventually becomes one. The dependence of R_{opt} on λ and M is only through λ/M. By comparing the two parts of Fig. 8, it is also evident that the advantage of using large groups, namely the increased sharing of resources, is more pronounced for long packets than for short ones. In fact, for $1/\mu < 10$, it is most practical to use $R = 1$ regardless of the arrival rate. Finally, note that for very low arrival rates, throughput is insensitive to group size.

Figure 9 depicts efficiency as a function of inbound throughput per receiver, for various group sizes. These results are obtained directly from the throughput results. Observe, for example, that with $1/\mu = 20$ and $S_{in}/M = 0.033$, the efficiency can be



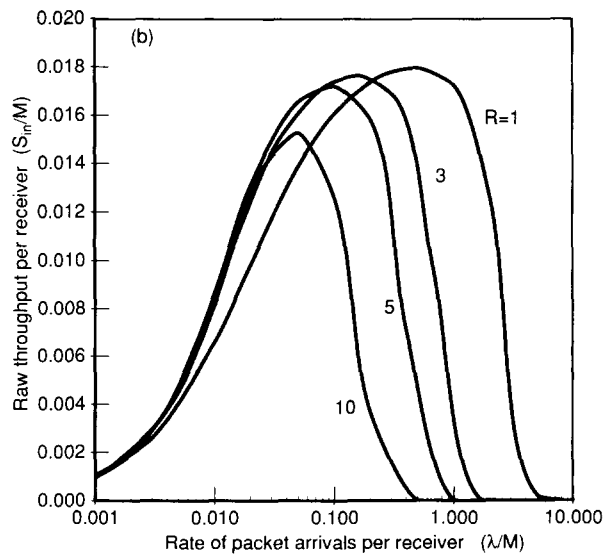


Fig. 8. Inbound throughput per receiver with FCAA for various group sizes. (a) $1/\mu = 10$; (b) $1/\mu = 50$.

increased by 50% by using $R = 3$ rather than $R = 1$.

4.2.3. *Dynamic-Code-Assignment Architectures (DCAA)*

In the dynamic-code-assignment architectures (DCAA), there is a controller that assigns codes to idle receivers and designates, for each code (if possible), an idle receiver to await packets on that code. Figure 10 shows a "generic" DCAA. The difference between various architectures of this class is in the knowledge that is available to the controller and in its consequent code-assignment policy. For any choice of (N_c, M, λ), the receivability with DCAA is the same as that with FCAA ($R = M/N_c$). The expected throughput enhancement stems from the increased sharing of receivers that is made possible by the dynamic reassignment of codes. For example, if $M = 12$ and $N_c = 4$, at least nine receivers must be busy before a receivable packet may be dropped by a DCAA node, whereas with FCAA, if three receivers with the same code are busy and a receivable packet arrives on that code, it is lost. Two extreme variants of DCAA are now explored:

Random assignment (*DCAA-RA*). The controller has no knowledge of the code of the next packet. Whenever the number of idle receivers, i, is smaller than N_c, the subset of codes that is covered is chosen at random. Therefore, a receivable packet that finds i idle receivers (receivers not busy in any phase of the reception of a receivable packet) is received with probability $\min\{i/N_c, 1\}$.

Optimal assignment (*DCAA-OA*). The controller knows the code of the next arriving packet, and designates an idle receiver to attempt to receive it. Consequently, a receivable packet will not

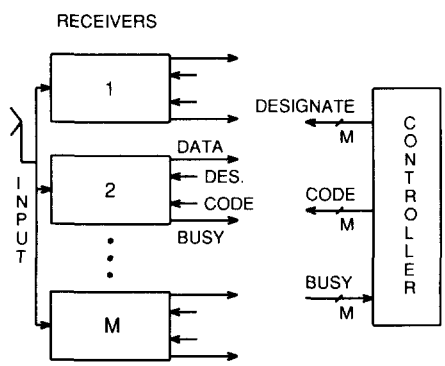


Fig. 10. Dynamic-code-assignment architectures (DCAA).

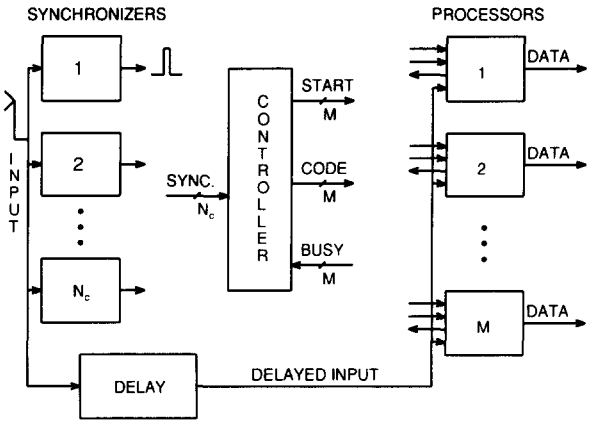


Fig. 11. Causal implementations of the seemingly noncausal DCAA-OA. The figure shows the pipelined version; the non-pipelined version is obtained by removing the "start" signals and replacing "processors" with "receivers".

be lost so long as there is an idle receiver; a possible implementation of this seemingly non-causal system is shown in Fig. 11. The multi-receiver node consists of a controller and of N_c "synchronizers", followed by a pool of M "processors". At all times, the incoming signal appears at the input of each synchronizer and, after going through a delay line, at the input of each processor. Each *synchronizer* operates independently with a distinct code. Whenever it synchronizes onto a packet, it so notifies the controller and immediately resets itself to wait for a new packet. The controller then instructs one of the idle *processors*, if any, to process this packet using the appropriate code. This architecture is pipelined, so the processors only perform the data-reception phase. However, bit synchronization must be maintained between the synchronizer and the processor that cooperate in the reception of any given packet. Alternatively, the processors could be replaced by complete receivers, which would not rely on the synchronization performed by the synchronizer. (The only task of the synchronizer would be to supply the advance knowledge of the arriving packet's code.) This obviates the need for bit-synchronization between the synchronizer and the processor, but requires a longer delay and gives up the advantage of pipelining. We use the latter version in the analysis of DCAA-OA in order to avoid distortion of the comparison due to the throughput advantage of pipelining, which can also by used with the other schemes.

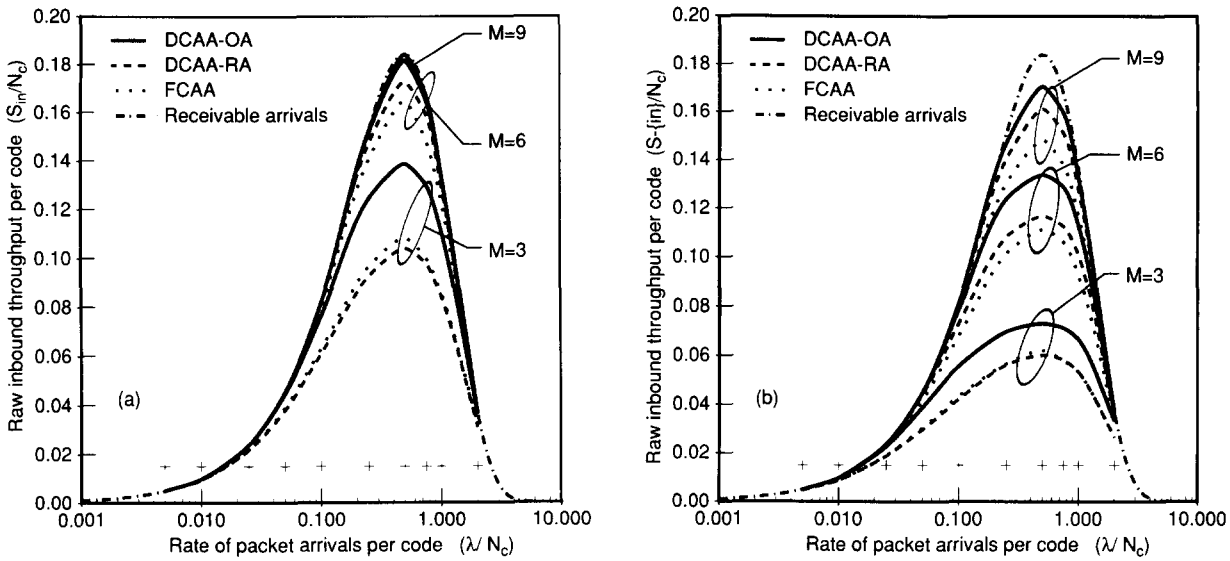


Fig. 12. Raw inbound throughput per code for FCAA, DCAA-RA and DCAA-OA. $N_c = 3$. (a) $1/\mu = 3.33$; (b) $1/\mu = 10$.

Analysis of DCAA-OA and of DCAA-RA differs from that of FCAA because the interarrival times of receivable packets (all codes combined) are not i.i.d. and the codes are not independent from arrival to arrival. We therefore resorted to simulation. [4]

Figure 12 depicts S_{in}/N_c versus λ/N_c for all three architectures; an additional curve shows the arrival rate of receivable packets per code, which is an upper bound on throughput. Note that the values are normalized per code, not per receiver. Curves are given for several values of M; the number of codes, N_c, is held fixed at 3. Graphs are presented for two packet lengths: (a) $1/\mu = 3.3$ and (b) $1/\mu = 10$. Figure 13 depicts the efficiency of the three architectures with $1/\mu = 10$.

4.2.4. Performance Comparison

Schemes will be compared on the basis of throughput for equal arrival rates. Consequently, the same results also apply to efficiency. Comparing DCAA-RA with FCAA, FCAA appears to slightly outperform DCAA-RA when $M = N_c$, contrary to the expectation that they would be identical. This minor anomaly stems from our modeling of DCAA-RA, and would disappear in a real implementation. [5] As the number of receivers increases, the advantage of DCAA-RA over FCAA becomes apparent.

Dynamic code assignment increases throughput as well as efficiency; the extent of the improvement depends heavily on the knowledge available to the controller. The advantage of DCAA over FCAA is most pronounced for intermediate values of λ: for very small values, the probability of reception is very high with any architecture; for very high values of λ, there is no improvement since receivability constitutes the bottleneck. Compared with FCAA, the maximum throughput is roughly 25% higher with DCAA-OA, and roughly 5% higher with DCAA-RA; this is for the range in which throughput is limited by receiver availability and not by receivability.

The dependence of the throughput advantage of DCAA on packet length is more complicated: consider, for example, the points of maximum throughput, and assume for the moment that the

[4] In the simulation, we used a method known as "common random numbers" (the exact same interarrival times and packet lengths for the schemes under comparison). This decreases the variance of the relative results.

[5] Our model for DCAA implies that there is a non-zero probability that, while one receiver is still performing the synchronization phase, another one is already designated to cover the same code. The latter is wasted during the remainder of the preamble, since no receivable packet can commence to arrive on that code at that time. In FCAA this can never happen when $M = N_c$. In practice, however, a designated receiver declares itself locked only at the end of the preamble, so no other receiver would be assigned to the same code until that time.

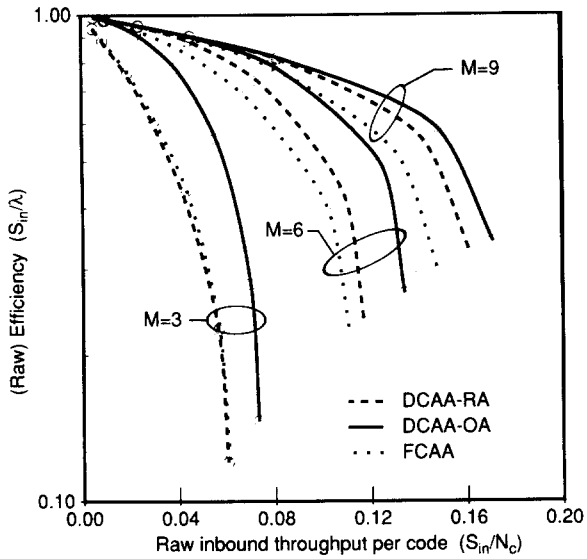


Fig. 13. Efficiency of channel usage with FCAA ($R = M/N_c$), DCAA-RA and DCAA-OA. $N_c = 3$; $1/\mu = 10$.

limiting factor is finding an available receiver. In this case, a receiver is nearly always designated to cover some code as soon as it becomes idle. The busy period of a receiver is therefore the packet length and its idle period is the time interval from the instant it becomes idle until a receivable packet arrives with the appropriate code. If the packet is long compared to the interarrival time of packets with any given code (which, in turn, is on the order of preamble length), the receiver's utilization is very high and cannot be improved much by decreasing the idle time through a knowledgeable code assignment as in DCAA-OA. If, on the other hand, the packet length is comparable to the preamble (short packets), there is more room for improvement. Consequently, one would expect a more significant improvement for short packets. There is, however, another trend: as the number of receivers is increased, the arrival rate of receivable packets becomes the limiting factor ("starved" receivers), in which case most codes are covered at any instant, thus causing the importance of knowledgeable code assignment to decrease. Obviously, this happens first for short packets, since the busy periods of the receivers are shorter. One should therefore expect a more significant improvement for long packets in this case. Indeed, referring to Fig. 12 and comparing DCAA-OA with FCAA: for ($N_c = 3$, $M = 3$), the improvement is 32% for short packets ($1/\mu = 3.33$) and only 23% for longer ones ($1/\mu = 10$), whereas for ($N_c = 3$, $M = 9$) it is down to a mere 1% for the short ones, whereas for the long ones it is 7%.

4.2.5. Error-Free Throughput

Unlike the probability of reception of a specific packet, which is independent of that packet's length, the probability of the packet being error-free may depend on its length. Consequently, the length-distribution of successfully received packets is not the same as that of transmitted or received packets, and stating traffic level or throughput in packets per unit time is ambiguous. We therefore replace the mean packet-arrival rate with the mean number of ongoing transmissions as the measure of traffic level; raw throughput is redefined to be the mean number of ongoing receptions, and error-free throughput is the mean number of ongoing receptions of packets that will be found error-free. Traffic level and raw throughput, expressed in packets per unit time, can be converted to the new units by multiplying them by the mean packet length ($1 + 1/\mu$).

Given the channel parameters, intercode interference can be characterized by the probability that a packet is error-free (i.e., if it were acquired successfully and received, there would be no erroneous bits) as a function of packet length and of the total traffic level λ_T. For a known length-distribution of received packets, it can be characterized simply by the probability that an arriving bit belongs to an error-free packet, as a function of λ_T. Then, given the level of background traffic λ_{BG}, the error-free throughput for each value of $\mathcal{S}$'s inbound traffic level, λ_S, is the product of the raw throughput at λ_S and this probability at the corresponding value of λ_T. Strictly speaking, the distributions of packet lengths used in obtaining the channel characteristics and the raw throughput must be the same; however, the results are clearly insensitive to small differences.

The value of λ_T which corresponds to given values of λ_S and λ_{BG} is only approximately ($\lambda_S + \lambda_{BG}$), since knowing that a packet was received biases the distribution of $\mathcal{S}$'s inbound-traffic level at the time of the packet's arrival; furthermore, this bias depends on the architecture. Nevertheless, we claim that the bias is very small, except for the case of very short packets and very low traffic levels. (In this case, however, intercode

interference is negligible altogether.) Our claim stems from the fact that the only knowledge gained from the fact that a packet with a preamble of length 1 is received at time t is that there was an available receiver on its code at time t and that no other packets with the same code commenced to arrive in $[t-1, t+1]$. No information is gained pertaining to background traffic or to arrivals after $t+1$. Very little information is gained pertaining to traffic on other codes (none in the case of FCAA), and not much regarding arrivals prior to $t-1$. Supported by simulation results, [6] we ignore the bias. (It should be emphasized that the only effect of this bias is to shift the curves horizontally, since it only affects λ_T. As will be seen in the curves that follow shortly, a horizontal shift of one curve with respect to the other by tens of percents has little effect on the relative performance of $\mathcal{S}$ with different code assignments. In other words, not only is the bias small, but the results are very insensitive to it.)

We now turn to a specific example. Figure 14, which is based on Fig. 4.6 in [2], shows the characteristics of a specific channel. [7] Plots are shown for 16, 64, 256 and 1024 chips per bit. (The FEC contributes a factor of 2 and the remainder is the PN spread-factor.)

Figure 15 shows the error-free throughput for FCAA with $1/\mu = 50$ and $M = 3$, with no other traffic; this is a combination of Figs. 8(b) and 14. We see that, although maximum raw throughput is obtained with $R = 1$, error-free throughput is higher with $R = 3$; i.e., it is best to operate all three receivers on the same code. Including background traffic and using longer packets in the intercode interference model (a typical preamble length is 40 bits; with $1/\mu = 50$, the mean packet length should be 2000 bits instead of the 1000 used in [2]) would further increase the advantage of $R = 3$. The use of a more realistic preamble-collision model would have a similar effect.

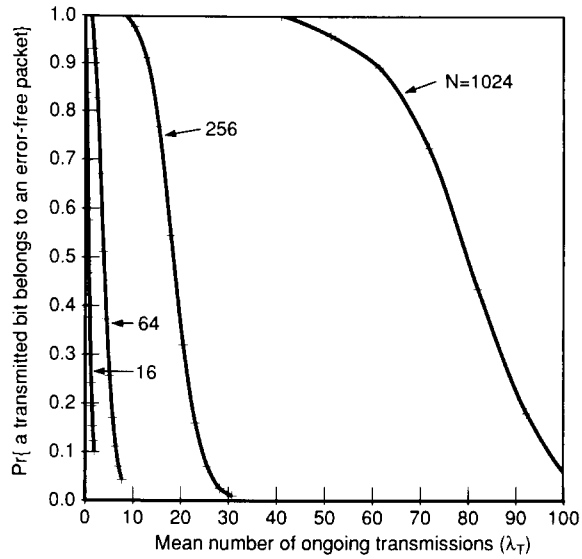


Fig. 14. Probability that a transmitted bit belongs to an error-free packet versus the mean number of ongoing transmissions. Packet-length is exponentially distributed with mean 1000 bits. Channel: DS-BPSK with convolutional FEC. The FEC contributes a factor of two to the number of chips per bit [2].

Error-free throughput can be obtained for all architectures in the same manner. Intercode interference has no qualitative effect on the comparison between FCAA and DCAA, since the curves representing raw throughput never cross over. As for the optimal number of different codes to be used with a given number of receivers, that number is never smaller than the corresponding number for FCAA; for DCAA-OA, it is always best to use as many codes as possible, constrained only by code availability and by the budget for synchronizers.

The dependence of error-free throughput on the arrival rate can be summarized as follows: as the arrival rate increases, raw throughput increases until it begins to decrease due to preamble overlaps which render arriving packets nonreceivable. The probability that a received packet is error-free decreases as the arrival rate increases, but is initially very insensitive to arrival rate. At some point, however, this probability begins to fall off sharply. Error-free throughput is maximized (approximately) at the lower of two arrival rates:

[6] We ran simulations for the case of FCAA with $M = 3$ and no background traffic; we used $R = 1$, 3 and $1/\mu = 10$, 50. For traffic levels of 10 ongoing transmissions or more, the bias of the traffic level is smaller than 10%. For levels of more than 30, the bias is smaller than 5% and for 75 it is down to 2%. The difference between the bias with $R = 1$ and with $R = 3$ is below 1%. These are the biases at the time of arrival; the bias decreases during the arrival of the remainder of the packet, so the average bias is even smaller.

[7] DS-BPSK channel with FEC. PN codes are assumed to be sequences of jointly independent Bernoulli (1/2) random variables. FEC: convolutional coding with hard decision Viterbi decoding. The specific code used is the rate 1/2 constraint length 7 code with generator polynomial (in Octal) 171, 133. Packet lengths are exponentially distributed with mean 1000 bits, and signal-to-noise ratio is $E_b/N_0 = 8.0$ [2].

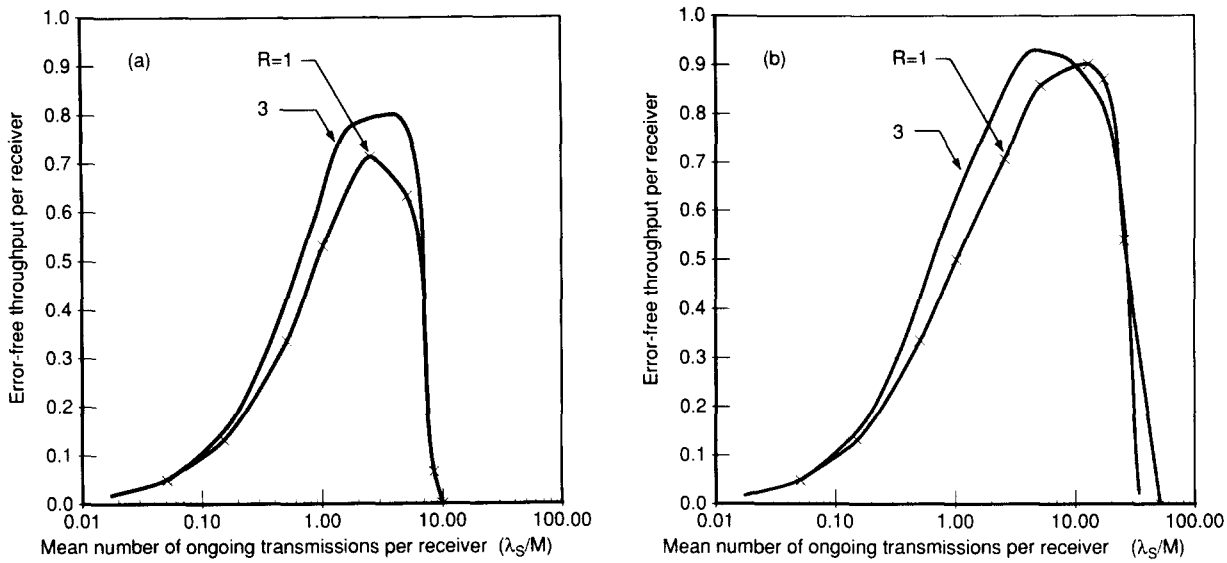


Fig. 15. Error-free throughput with FCAA. (a) 256 chips per bit; (b) 1024 chips per bit. $M = 3$, $1/\mu = 50$.

(i) the one which maximizes the raw throughput, and

(ii) the one at which the probability of a packet being error-free begins to fall off sharply.

4.6. *Permitting $\mathscr{S}$ to Transmit*

At the outset, it should be noted that a supernode is less likely than a conventional node to be subject to the half-duplex constraint. For example, a single antenna can serve all receivers (and no transmitters), thus constituting little overhead. Furthermore, in situations such as the terrestrial hub of a two-hop satellite network, the hub is always full duplex since the up-link and down-link use nonoverlapping spectral ranges. It should also be noted that the incorporation of the fact that a half-duplex supernode may be transmitting has no effect on the relative inbound throughput of the different architectures and code-assignment policies. Nevertheless, this issue is addressed briefly for the sake of completeness.

Let us consider the following policy for the operation of $\mathscr{S}$: transmission may not commence while any of the receivers are busy; whenever all receivers become idle, $\mathscr{S}$ commences transmission immediately if it has packets for transmission; if there are no packets for transmission, it must wait until at least one packet is received. The rule for terminating a transmission epoch is not specified; however, once $\mathscr{S}$ stops transmitting, it must wait for a reception epoch before it may transmit again. Some portions of this set of rules are realistic (e.g. no transmissions may commence when engaged in reception), while others would be modified slightly in a realistic situation. Note, however, that the transmission policy is consistent with the desire to operate all transmitters together, which was explained in Section 4.1.

From the above set of rules, it follows that $\mathscr{S}$ alternates between reception and transmission epochs. A reception epoch begins upon termination of transmissions. Initially, all receivers are idle; then, some packets are received. (This idle-busy cycle may be repeated any number of times, and is considered a single reception epoch.) The beginning of a transmission epoch always coincides with the last busy receiver becoming idle, and (obviously) terminates with all receivers idle. Note that idle times (no transmissions or receptions) are considered to be part of a reception epoch.

The calculation of inbound throughput for a nontransmitting supernode were based on cycles that began and ended with the arrival of a receivable packet to an empty system. Comparing this with the reception epoch that has just been defined, one observes that the latter consists of several such cycles, but is missing the time interval from the instant that the last receiver becomes idle in the last cycle of the epoch until the arrival time of the next receivable packet. On the other hand,

the reception epoch contains an extra time interval at the beginning, namely the time from end of transmissions until the arrival of the first receivable packet. Due to the memoryless nature of the arrival process, and the meaningless difference in side-information provided by the two states, these two intervals are statistically equal. Therefore, the analysis that was presented for a nontransmitting supernode remains valid, and the new throughput can be obtained by multiplying those results by the fraction of time in which $\mathcal{S}$ is not transmitting. This fraction can be estimated in each specific case, but is beyond the scope of this discussion.

5. Full-Duplex Nodes

There are cases in which a node can transmit and receive concurrently. One example is a local-area network which uses low-attenuation broadcast media, so that the levels of received and transmitted signals are similar and the interference with a node's reception due to its own transmission is similar to the interference caused by a transmission of some other node. Another example would be a packet radio network in which the transmitter and the receiver of a node are not collocated, or at least use different antennas. In this section, the results of the previous sections are adapted to the case of full-duplex nodes.

5.1. Link Masking (Slotted System)

If only $\mathcal{S}$ is full duplex, the results obtained for $p_0 = 0$ are valid for all values of p_0. If all nodes are full duplex, one could construct a unary tree of all nodes, i.e., a chain, such that each node can receive traffic for $\mathcal{S}$ from only one of its neighbors and can transmit such traffic only to one of its neighbors. If there were no traffic other than $\mathcal{S}$'s inbound traffic, this could result in an inbound throughput of 1.0. However, this is highly impractical for several reasons: assuming that a packet originates from any of $\mathcal{S}$'s neighbors with equal probability, a packet has to travel $\frac{1}{2}N$ hops on average, which may cause delay to be prohibitive; this also amounts to a very low efficiency of channel usage. The mean number of ongoing transmissions is $\frac{1}{2}N$, which may even exceed the capacity of the channel, causing an error-free throughput of 1.0 to be unattainable.

Realizing that the chain idea is impractical, the next step is for $\mathcal{S}$ to authorize two of its neighbors to transmit to it. Since those neighbors are also full duplex, their inbound throughputs can be $1/e$ (even with infinite population), which is more than half of $\mathcal{S}$'s maximum inbound throughput (0.5). Consequently, a height 1 binary routing tree is sufficient (2-hop link masking). The improvement over the half-duplex case is in that an inbound throughput of 0.5 can be achieved for all values of p_0.

5.2. Multiple Receivers and Transmitters

With multiple transmitters, time-synchronization is no longer necessary in order to increase the availability of $\mathcal{S}$ for reception. However, we argue that it is best to operate all of $\mathcal{S}$'s transmitters concurrently and continuously in order to maximize its outbound throughput, and justify this as follows. Given the environment, characterized by mean number of ongoing transmissions (excluding $\mathcal{S}$'s) heard by $\mathcal{S}$'s neighbors, there is a number of ongoing transmissions by $\mathcal{S}$ which maximizes the expected value of its outbound throughput. Let this number be denoted by T. Ignoring the fact that T may not be an integer, it follows that equipping $\mathcal{S}$ with T transmitters and operating them continuously will maximize $\mathcal{S}$'s outbound throughput. This, in a sense, is time-synchronization. When the desired outbound throughput is smaller than the maximum, it can be achieved either by reducing the number of transmitters or by having them transmit intermittently. The difference between the two approaches is in the efficiency and, consequently, in the throughput of the other nodes. Whenever $P_S(l)$, the probability that a packet is received successfully in the presence of $(l-1)$ other transmissions, is concave, the better approach is to reduce T and transmit continuously.

6. Summary

Equipping a node with several receivers and transmitters increases its inbound and outbound throughput, respectively. The increase is eventually limited by channel capacity. The optimal number of receivers is infinite, although there is little to be gained beyond a certain number. The

optimal number of transmitters is finite. Whenever a node cannot receive while transmitting (half-duplex), it is important to enforce time-synchronization between the node's transmitters. This applies to slotted as well as unslotted systems.

In an unslotted system, time-capture permits various ways of assigning codes to receivers, and permits the M-receiver supernode to have a higher inbound throughput than M independent, collocated single-receiver nodes. With fixed assignment, the optimal number of receivers that should share a common code is higher for long packets than for short ones, increases with a decrease in channel capacity or an increase in the level of background traffic, and decreases with an increase in the level of inbound traffic. We note that the case in which each of the M receivers has its own code is the same as M separate nodes. Although maximum throughput can be quite insensitive to the number of receivers that share a common code, a significant improvement in efficiency can be achieved by the proper selection. Dynamic reassignment of codes to receivers improves throughput as well as efficiency.

In situations wherein only long-term uniformity of code usage can be assumed, the advantage of DCAA over FCAA is much more pronounced than suggested by our results, since DCAA would adapt to the transient skew in code usage, whereas FCAA would not. In the extreme case that all packets are arriving with a single code, the throughput advantage of DCAA-RA over FCAA would be on the order of $(M - N_c) : M/N_c$, and that of DCAA-OA over FCAA would be on the order of $N_c : 1$.

With a slotted system and single-slot packets, masking all but two of a node's incoming links can increase its inbound throughput by up to 36%; furthermore, at high throughput levels, particularly when $\mathscr{S}$ itself transmits frequently, link masking is more efficient than direct transmissions. Link masking requires no additional hardware, since it makes use of otherwise lightly utilized hardware in neighboring nodes. The protocol required to support link masking is very simple and robust. In unslotted systems, link masking has limited application due to time capture.

The number of supernodes that can coexist (efficiently) in the same region of a network is limited primarily by channel capacity; the availability of codes could also be a limiting factor, but more often than not this is not the case [13].

Delay was not addressed directly in this discussion. Nevertheless, whenever the throughput of one architecture exceeds that of another for all arrival rates, that architecture is also superior in terms of delay.

References

[1] Special Issue on Spread-Spectrum Communications, *IEEE Transactions on Communications 30* (5) (1982) 817–1072.

[2] J. Storey and F. Tobagi, Throughput Performance of a Direct Sequence CDMA Packet Radio Network, Stanford University Computer Systems Laboratory Report, SEL TR 85-277, 1985.

[3] E. Parker, Micro Earth Stations as Personal Computer Accessories, *IEEE Proceedings 72* (11) (1984) 1526–1531.

[4] Y. Birk and F.A. Tobagi, Supernodes in Packet Radio Networks Employing Code Division Multiple Access, *Proc. IEEE Military Communications Conference, MILCOM '85*, Boston, MA (1985).

[5] R. Kahn, S. Gronemeyer, J. Burchfiel and R. Kunzelman, Advances in Packet Radio Technology, *IEEE Proceedings 66* (11) (1978) 1468–1496.

[6] G. Clark, Jr. and J. Cain, *Error-Correction Coding for Digital Communications* (Plenum Press, New York, 1981).

[7] L.G. Roberts, ALOHA Packets with and without Slots and Capture, *Computer Communications Review 5* (1975) 28–42.

[8] I. Gitman, On the Capacity of Slotted ALOHA Networks and Some Design Problems, *IEEE Transactions on Communications 23* (3) (1975) 305–317.

[9] F.A. Tobagi, Performance Analysis of Packet Radio Communication Systems, *Proc. IEEE National Telecommunications Conference* (1977) 12:6-1–12:6-7.

[10] G. Akavia and L. Kleinrock, Performance Tradeoffs and Hierarchical Designs of Distributed Packet-switching Communication Networks, UCLA Technical Report No. UCLA-ENG-7952, 1979.

[11] N. Abramson, The ALOHA System—Another Alternative for Computer Communications, *AFIPS Conference Proceedings, 1970 Fall Joint Computer Conference 37* (1970) 281–285.

[12] H. Takagi and L. Kleinrock, Output Processes in Contention Packet Broadcasting Systems, *IEEE Transactions on Communications 33* (11) (1985) 1191–1199.

[13] Personal Correspondence with John H. Cafarella, MICRILOR, P.O.Box 624, Swampscott, MA 01907.