A SIMPLE STACK ALGORITHM FOR A CODE DIVISION MULTIPLE ACCESS COMMUNICATION SYSTEM

Ioannis Stavrakakis and Dimitri Kazakos

Electrical Engineering Department
University of Virginia
Charlottesville, VA 22901

Abstract

A frequency hopping multiple access communication system with an infinite population of potential users is considered. By making use of the idea of the t-orthogonal frequency hopping pattern of order Δ, the channel is split into a number of mutually interference free subchannels; then the idea of frequency hopping pattern sensing (FHPS) is adopted to provide ternary feedback information to the users.

A packet is made to consist of a number of minipackets, which are capable of revealing the frequency hopping pattern that was used in the transmission of the whole packet. A simple algorithm, that belongs to the class of the STACK algorithms, is adopted for conflict resolutions.

The performance of the system in terms of throughput and average packet delay is investigated and both analytical and simulation results are provided.

The Communication System

We assume that an infinite population of bursty users share a common communication channel. The users can have access to the channel anytime they have a message to transmit.

The message consists of M minipackets. Each minipacket has length equal to T time units and consists of N bytes. Time is slotted and the length of the slot is equal to T. Packet transmission can start only at the beginning of a slot.

We assume that message transmission employs a frequency hopping (FH) scheme. The available frequency spectrum is divided into q frequency slots. Each of the N bytes of a minipacket is transmitted at a frequency chosen from the set Q of the frequency slots, according to a frequency hopping pattern represented by the vector

$$\overline{f} = (f_1, f_2, \ldots f_N) \ , \ f \epsilon Q = \{q_0, q_1, \ldots, q_{q-1}\}.$$

Let $C_{N,q}$ be the set of all FH patterns with N coordinates that take values from a set of q frequency slots and has the t-orthogonality property. By the t-orthogonality property we mean that no two elements of $C_{N,q}$ can have more than t equal coordinates and yet $C_{N,q}$ is a t-error correcting codebook. R-S codes can be used for this purpose.[1]

Even if synchronization of a receiver with the starting point of a given minipacket addressed to it is possible, all other packets in the channel will undergo a time offset in their arrival time at the particular receiver, depending on the propagation delay. When propagation delay is not negligible compared with the time duration of a byte, then it is necessary that the FH patterns are chosen from a set $\overline{C}_{N,q}$. The elements of $\overline{C}_{N,q}$ have the same properties as those in $C_{N,q}$ and furthermore these properties hold for cyclically shifted by up to Δ positions elements of $\overline{C}_{N,q}$; we say that $\overline{C}_{N,q}$ has the property of t-orthogonality of order Δ.[2,3]

Each user is assigned one of the FH patterns that belong to $C_{N,q}$ in such a way that all FH patterns have the same probability of being used. By that assignment we create a number of mutually interference free subchannels, each one of them having Poisson input traffic with intensity

$$\lambda_i = \frac{\Lambda}{|C_{N,q}|}$$

where Λ is the global input traffic to the channel and $|C_{N,q}|$ is the number of FH patterns in $C_{N,q}$, i.e. the number of subchannels.

The FH pattern assignment is supposed to be a receiver based one.[4] This means that a user receives messages only through the subchannel assigned to him and searches for a message addressed to him only according to the specific FH pattern that he has been assigned.

Frequency Hopping Pattern Sensing (FHPS)

The FHPS characteristic of our model is used to provide feedback information, concerning the status of the particular subchannel, to all active users who want to make use of that subchannel. An active user keeps sensing the subchannel of his interest, starting from the first time slot following his message arrival, until this message has been successfully transmitted (limited subchannel sensing). A ternary feedback information revealing whether the subchannel was idle, involved in a collision or successfully transmitting, is available before the end of the current time slot.

At this point, we should make clear that FHPS and subchannel sensing are equivalent statements. Unlike the simple Carrier Sensing procedure, FHPS cannot be carried out so fast and the whole slot is considered to be involved in the FHPS procedure.

Research supported by the Air Force Office of Scientific Research through Grant AFOSR 82-0030.

Protocol Description

Through the manner in which the subchannels were created, it is obvious that all of them are mutually interference free. Thus we can treat them separately, and apply the protocol that is described in this section to each one of them independently.

Since access to the subchannels is random, collisions occur and the need of a collision resolution algorithm arises. For this purpose, we will adopt an algorithm that belongs to the class of the STACK Algorithms.[5,6]

We assign $|C_{N,q}|$ counters to each user, one for each subchannel. No counter of a user is enabled, until a message arrives. Then the counter that corresponds to the subchannel that the user has to use is enabled, and its content increases or decreases depending on the outcome of the FHPS procedure, the content of the counter itself and the algorithm steps.

At this point, we concentrate on a simple subchannel. We define the classes $B_0, B_1, B_2 \cdots$ to be the groups of users whose counter content is 0,1,2,.. respectively, and

B = {users who don't have a packet to transmit}.

Also, let

$$B_0 = \left\{ \text{new users, i.e., users who have a packet to transmit, but have not attempted any packet transmission so far} \right.$$

and

$$B_1 = \left\{ \right.$$

Users who belong to the set $\bigcup_{n=0}^{\infty} B_n$ are called active users while users who belong to the set $\bigcup_{n=1}^{\infty} B_n$ and have entered class B_2 at least once since the time when they became active for the last time, are called blocked users.

Algorithm Description

All active users keep sensing the subchannel by making use of the FHPS procedure. All users that are found to belong to the class B_1 at the beginning of a slot, attempt packet transmission at the beginning of that slot. At the end of each slot, a ternary feedback information, revealing the status of the subchannel during that slot, is available to all active users. Let F denote the feedback information; it takes values from the set {I,C,S} depending on whether the subchannel was found to be idle, involved in a packet collision or successfully transmitting, respectively. Then,

I. If F=S, then

(a) $B_0 \to B_0$ (b) $B_2 \to B$ if $\omega=0$, $B_2 \to B_1$ if $\omega>0$

(c) $B_k \to B_k$, $k \geq 2$

II. If F=C, then

(a) $B_0 \to B_1$ (b) $B_1 \to B_1$ with prob. p, $B_1 \to B_2$ with prob 1-p

(c) $B_k \to B_{k+1}$, $k \geq 2$

III. If F=I, then

(a) $B_0 \to B_1$

(b) If last non idle slot was involved in a collision, then

$B_2 \to B_1$ with prob. p, $B_2 \to B_2$ with prob. 1-p , $B_k \to B_k$ for $k \geq 3$

(c) If last non idle slot was involved in a successful transmission, then

(i) if this slot is the first idle slot after the successful one, then $B_k \to B_k$, $k \geq 2$

(ii) if this slot is not as in (i), then $B_k \to B_{k-1}$, $k \geq 2$

Comments on the Protocol

It should be made clear that a packet collision can be detected by all active users. When this event occurs, the sender aborts his transmission before the end of the current slot. As a result, only one slot is wasted in a collision. If T is large compared with the propagation delay, then only a small portion of the FHP is capable of revealing whether a collision has occurred or not. As a result, abortion of transmission can be completed within the current slot.

In addition to the main counter that determines the class of a user, there is also a downcounter ω assigned to each user. It starts downcounting, from M to 0, at the slot in which the first minipacket was successfully transmitted, and decreases by one unit every slot. Its existence serves two purposes: The first one is to determine the time when the user will enter class B. The second is to provide some protection against the loss of a packet, due to erroneous feedback information. In the latter case another user might attempt transmission while a successful transmission was in progress, resulting in a collision and destroying the original message. The user who was interrupted will still belong to the class B_1 and thus he will attempt transmission at the beginning of the next slot. Thus, message loss is avoided and some priority is given to the unlucky user as well. Since the analysis of the protocol will be based on the assumption of an errorfree channel, such events will not be considered in the analysis.

Successful transmissions are never interrupted by new or blocked users. We assume that the whole packet is successfully transmitted once the first minipacket has been successfully transmitted.

Users in class B_0 (i.e. new users) are allowed to attempt packet transmission after a slot involved in a collision. This makes sense since the detected collision will be ended before the beginning of the next slot and the probability of appearance of a new message becomes small as M increases. A collision implies that there are some users in the system that may claim transmission in the next slot but this event depends on the splitting probability p and thus collision in the next slot is not certain. On the other hand we maintain the characteristics of the continuous entry to the system and that of the priority of the new users over the blocked ones.

Algorithm step III(c)ii gives a chance to the new messages that have arrived during a successful transmission, to be transmitted before further resolution of the previous conflicts is performed. In other words, new users are given some priority over the blocked ones to either transmit successfully or join the blocked users. This step together with the comment that we made in the previous paragraph, emphasizes the continuous entry to the system and furthermore the priority that is given to the new users over the blocked ones.

Analysis of the Algorithm

We used the concept of the session and developed recursive equations to describe the operation of the system. A session is defined as the time interval between two renewal slots; the latter is defined as the second of two consecutive idle slots in which there was no blocked user in the system. The number of users who attempt packet transmission at the beginning of a session determines the multiplicity of that session and all users transmit successfully during that session. Since the last two slots of a session are idle, the multiplicity of the sessions are independent Poisson distributed random variables with intensity λT.

Let L be the mean session length and let C be the mean cumulative in system delay of all packets that arrive in a single session. The in system delay of a packet is defined as the time that elapses between the instant when a packet enters class B_1 for the first time and the instant when the whole packet has been successfully received by the receiver.

By following procedures similar to those that appear in [5], [6], [7] and [8] we found tight lower and upper bounds on the max stable throughput S_{max} and on L and C. The results appear in tables I and II for various values of the input traffic, λ, and using the number of minipackets per packet, M, as a parameter. In the sequence, by using the strong law of large numbers[9], we proved that the mean packet delay is given by

$$D = D_A + \frac{C}{\lambda \cdot L} \quad \text{with probability 1} \quad (\lambda < \lambda^1) \qquad (1)$$

where D_A is the mean access delay, i.e. the average time that elapses between a packet arrival instant and the instant when the packet enters class B_1 for the first time. By using again the strong law of large numbers we found

$$D_A = \frac{6(1-\lambda(M+1)) + \lambda(M+1)(2M+3)}{4}$$

$$\text{with probability 1} \quad (\lambda < \lambda^1)$$

Results and Comments

In table I approximate values (up to the fourth decimal digit) for the maximum stable throughput are shown. It can be easily seen that S_{max} increases drastically as M (the number of minipackets per packet) increases.

By substituting the values for the upper and lower bounds on the mean session length and the mean cumulative in system delay into expression (1), upper and lower bounds on the average packet delay can be obtained. Since $L^u \approx L^1$ and $C^u \approx C^1$ we obtain the approximate expression

$$D \approx D_A + \frac{C^1}{\lambda L^1}$$

for the mean packet delay; the accuracy of the previous expression is restricted by the accuracy of the fourth - or beyond that - decimal digit in C^1 and L^1. The values of the mean packet delay for some values of $\lambda < \lambda^1$ and for M=1,2,5,10,100 minipackets per packet are shown in table II.

A plot of the mean packet delay, D, versus the input traffic rate for M=1,2,5,10,100 appears in Fig. 1; some simulation results are also shown in Fig. 1. The simulation results seem to be in accordance with the analytical ones for M=1,2. For the cases of M=5,10 the simulation results appear to be smaller and this is mostly due to the need for longer operation of the simulator and, to some degree, to the finiteness of the population.

We should note that the analytical results were obtained for an infinite population user model which justifies the poisson arrival model that was adopted. If the user population is finite, simulation results showed that the delay performance of the algorithm is better than that of the infinite user population model. From Fig. 2 can be seen that, when $\lambda < \lambda^1$, D increases monotonically as the user population increases and tends to the value found for the infinite user population case. On the other hand D increases rapidly as the user population increases when $\lambda > \lambda^u$; we believe that if the simulator was free of computer time limitations the increase would be more rapid and D should eventually approach infinity as the user population increases.

Conclusions

A simple STACK algorithm for a Code Division Multiple Access Communication System was suggested. The channel was split into several mutually interference free subchannels by using a class of t-orthogonal frequency hopping patterns of order Δ. In the sequence, a collision resolution algorithm belonging to the class of the STACK algorithms was applied to each of the subchannels.

The emphasized continuous entry characteristic and the limited sensing property, which are basic characteristics of the class of the STACK algorithms, offer increased robustness and applicability to the system. Furthermore, the FHPS procedure provides feedback information which is more insensitive to channel errors compared to the simple CS procedure. This is due to the spread spectrum transmission scheme and the long sensing of the subchannel.

Analysis of the protocol was performed and very good approximations of the values of the maximum stable throughput and the average packet delay were obtained. The performance of the system increases drastically as the number of minipackets per packet, M, increases; large values for M is usually the case in a spread spectrum system and thus this protocol applies efficiently in such a communication system. Simulation results were also obtained and they were found to be in accordance with the analytical ones.

Table I

Upper and lower bounds on the maximum stable throughput for M minipackets per packet.

M	$\lambda^l = \lambda^u$
1	.273
2	.453
5	.676
10	.812
100	.977

Table II

Upper and lower bounds on L and C (in time slots) and values for D (in packet lengths) for $\lambda < \lambda^l$ and M minipackets per packet.

M = 1

λ	$L^l = L^u$	$C^l = C^u$	D
.010	1.020	0.010	2.535
.050	1.117	0.069	2.735
.100	1.290	0.222	3.200
.150	1.584	0.643	4.132
.200	2.211	2.300	6.602
.250	4.760	19.239	17.542
.260	6.580	41.559	25.662
.270	11.157	135.282	46.273
.273	14.280	230.128	60.394

M = 2

λ	$L^l = L^u$	$C^l = C^u$	D
.010	1.015	0.010	1.762
.050	1.082	0.057	1.823
.100	1.184	0.139	1.938
.150	1.317	0.265	2.114
.200	1.498	0.484	2.396
.250	1.764	0.912	2.849
.300	2.199	1.896	3.662
.350	3.049	4.868	5.355
.400	5.474	21.032	10.405
.440	20.075	359.548	41.510
.443	25.541	592.653	53.184

M = 5

λ	$L^l = L^u$	$C^l = C^u$	D
.010	1.012	0.010	1.305
.100	1.137	0.118	1.337
.200	1.362	0.301	1.477
.300	1.607	0.641	1.735
.400	2.077	1.428	2.158
.500	3.045	3.981	3.090
.600	6.231	21.617	6.292
.650	14.321	130.382	14.533
.670	31.075	647.895	31.652
.678	59.217	2404.23	60.506

M = 10

λ	$L^l = L^u$	$C^l = C^u$	D
.010	1.011	0.010	1.193
.100	1.124	0.114	1.211
.200	1.285	0.274	1.302
.300	1.506	0.524	1.438
.400	1.831	0.974	1.651
.500	2.356	1.944	2.012
.600	3.360	4.688	2.730
.700	6.062	18.028	4.695
.770	14.647	118.671	10.999
.812	116.156	8037.01	85.706

M = 100

λ	$L^l = L^u$	$C^l = C^u$	D
.010	1.010	0.010	1.019
.100	1.112	0.119	1.070
.200	1.253	0.257	1.141
.300	1.435	0.461	1.234
.400	1.680	0.771	1.360
.500	2.027	1.294	1.538
.600	2.557	2.304	1.812
.700	3.467	4.672	2.284
.800	5.395	12.425	3.288
.900	12.214	70.183	6.842
.950	33.395	552.386	17.893
.970	109.410	6065.79	57.647
.977	539.488	150042.	285.163

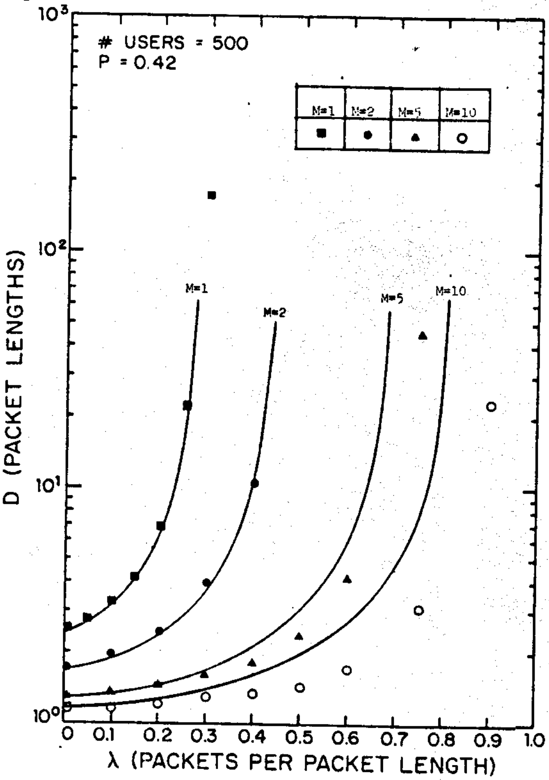

Figure 1

Analytical and simulation results for the average packet delay D versus the input traffic rate λ, with the number of minipackets per packet as a parameter.

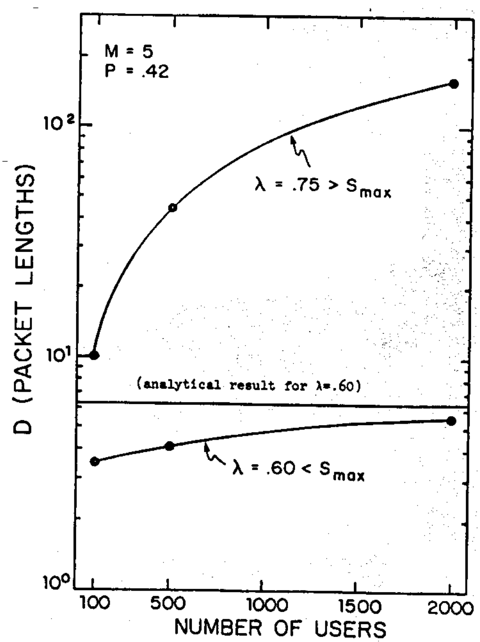

Figure 2

Simlation results for the average packet delay D vs. the number of users in the system.

References

1. I.S. Reed, "k^{th}Order Near Orthogonal Codes", IEEE Transactions on Information Theory, January 1971.

2. R.M. Mersereau, T.S. Seay, "Multiple Access Frequency Hopping Patterns with Low Ambiguity", IEEE Transactions on Aerospace and Electronics Systems, July 1981.

3. T.S. Seay, "Hopping Patterns for Bounded Mutal Interference in Frequency Hopping Multiple Acces", IEEE MILCOM'82, Vol. 1.

4. E.A. Sousa, J.A. Silvester, "Aspreading Code Protocol for a Distributed Spread Spectrum Packet Radio Network", IEEE GLOBECOM'84, November 26-29, 1984, Atlanta, Georgia.

5. B.S. Tsybakov, N.D. Vvedenskaya, "Random Multiple Access Stack Algorithm", translated from Problemy Peredachi Informatsii, Vol. 16, No. 3, pp. 80-94, July-September, 1980.

6. N.D. Vvedenskaya, B.S. Tsybakov, "Random Multiple Access of Packets to a Channel with Errors", translated from Problemy Peredachi Informatsii, Vol. 19, No. 2, pp. 52-68, April-June, 1983.

7. L. Georgiadis, P. Papantoni-Kazakos, "Limited Sensing Algorithms for the Broadcast Channel", EECS Dept. Technical Report TR-84-8, June 1984.

8. L.V. Kantorovich, V.I. Krylov, "Approximate methods of higher analysis", pp. 21, Interscience Publishers, 1958.

9. G.L. Chung, "A course in probability theory", Academic Press, Inc. 1974.