MIT/LCS/TM-287

NETWORK CONTROL BY BAYESIAN BROADCAST

Ronald L. Rivest

September 1985

Network Control by Bayesian Broadcast[*]

Ronald L. Rivest[**]

Version 7/22/85 – Revised 9/17/85

Abstract

We present a new transmission control strategy for local-area networks in which each station dynamically estimates the probability that r stations are ready to transmit a packet, for each nonnegative integer r. A station then transmits a packet with a probability based on these estimates. Each station updates its estimates using Bayes' Rule after observing whether the current time slot contained a collision, a successful transmission, or a hole (an empty slot). This strategy we call Bayesian Broadcast.

An elegant related strategy – which we call Pseudo-Bayesian Broadcast – is derived by approximating the probability estimates by a Poisson distribution with mean λ and further simplifying. Each station keeps a copy of λ, transmits a packet with probability $\frac{1}{\lambda}$, and then updates λ in two steps:

- If there was a collision increment λ by $\frac{1}{e-2} = 1.39221$, otherwise (for holes and successes) decrement λ by 1.
- Set λ to $\max(\lambda + \alpha, 1)$, where α is the average success rate on the channel.

Simulation results are presented demonstrating that the Pseudo-Bayesian Broadcast algorithm, although simple, performs very well in practice.

Keywords: Networks, broadcast procedures, packet networks, ALOHA, Bayes' Rule, binary exponential backoff, Ethernet, local-area networks.

[*] This research was supported by NSF grant MCS-80-06938.

[**] MIT Laboratory for Computer Science; Room NE43-324, 545 Technology Square Cambridge, Mass. 02139

I. INTRODUCTION

We propose three new strategies for the classic problem of controlling traffic on a local-area or satellite broadcast communications network. The first strategy, which we call Bayesian Broadcast, is presented to develop the underlying ideas. It is an "ideal" strategy that is unlikely to be cost-effective in practice. The second strategy, which we call Pseudo-Bayesian Broadcast, is an extremely simple and elegant approximation to Bayesian Broadcast. We give simulation results demonstrating that Pseudo-Bayesian Broadcast is exceptionally effective and stable in practice. The name "Bayesian Broadcast" reflects the fact that each station will dynamically estimate the number of active stations on the network using Bayes' Rule, and will calculate an transmission probability from this information that is optimum given the available information. The third strategy, Recursive Pseudo-Bayesian Broadcast, is a variation of the second strategy that performs better, at the expense of increased control complexity.

Let us consider a network with some number (possibly infinite) of stations. Each station is given packets to transmit by an associated processor. In practice a station may keep a queue of packets ready to send, if its processor momentarily generates packets more quickly than the stations can transmit them over the network. In this paper we concentrate on the model where each station has at most one packet to transmit at any time (often called the *infinite source* model). We call a station *active* if it has a packet to transmit, otherwise we call it *inactive.*

We assume that time is divided into *slots*, each long enough to tranmit one packet (the "slotted ALOHA" or "S-ALOHA" model). Our procedures generalize for other models (see section VI).

When a slot begins each active station must decide, either deterministically or stochastically, whether or not to transmit its packet. There are three possible outcomes:

- A *hole* if no stations transmit.
- A *success* if one station transmits.
- A *collision* if more than one station transmits.

When a hole or a collision occurs, no stations receive any feedback other than the fact that a hole or a collision has occurred.

We assume that the network objective is to minimize the average delay experienced by a packet between the time it is given to a station and the time it is successfully transmitted; by Little's result [Kl75] this is equivalent to minimizing the average backlog in the network.

There are a number of different models possible for the feedback each station obtains from the network. We use the most straightforward model, where each station can distinguish collisions from successes from holes.

Each station will have a (common) *control strategy* specifying how often it will transmit packets, including how often it will retransmit a packet which was involved in a collision.

Metcalfe and Boggs [MB76] recommend the *Binary Exponential Backoff* strategy, where a packet that has been involved in k collisions waits an amount of time randomly chosen between 1 and 2^k time slots before it is retransmitted. Thus, as the network becomes more congested, the probability that a station will transmit decreases. Gerla and Kleinrock[GK77] discuss a number of adaptive strategies for the S-ALOHA network.

Tanenbaum [Ta76] surveys a number of possible approaches to this problem. Some related work appears in [HVL82, GGMM85]. Gallager [Ga85] provides an excellent overview of the field, and the special issue of the IEEE Trans. Infor. Theory [Ma85] contains many superb papers on this topic.

Our approach has the following general form. Just before slot t begins, each station k in the network computes a value for its *broadcast probability* $b_{k,t}$. Then station k will transmit a packet (if it has one) with probability $b_{k,t}$, independent of whether previous attempts had been made to transmit that packet.)

How should station k compute $b_{k,t}$? It can use *global* inform<ation that is known by every station, and *local* information known only by station k.

The available global information might consist of:

- The *network hole/success/collision history*, indicating whether each slot was a hole, a success, or a collision.
- The *sender history*, indicating the source of each successfully transmitted packet.
- The *sender parameter vector*, indicating the local variables of the source of each successfully transmitted packet. (e.g. its transmission probability).

The last two items would only be globally known if stations include it in their packets. Control information can be included or "piggybacked" onto the successfully transmitted data packets to make it global.

The kinds of *private* information known by station k include:

- The *private transmission history* indicating which time slots station k actually tried to transmit. Other stations know that station k tried to transmit only if its packet was successfully transmitted; if a collision occurs the stations do not know which stations were transmitting. When a collision occurs, those stations that tried to transmit know there is *at least one* other active station, whereas stations that didn't try to transmit know there are *at least two* active stations (not counting itself) in the system.
- The *private parameter vector*, containing station k's control parameters. As noted above, from time to time this information might be made global by station k, but in between such times, it is private information.

We consider here at first the situation where the stations only use *global* information to compute the transmission probabilities $b_{k,t}$. Then each station will compute the *same* value b_t for $b_{k,t}$, and our Bayesian updating procedure will be relatively straightforward.

In fact, the only kind of global information we will use will be the network hole/success/collision history. Other global information might be incorporated into a Bayesian Broadcast procedure; we do not consider this possibility here.

In section II we develop the "theory" of our Bayesian Broadcast, showing how each station can choose an optimum broadcast probability for each slot. However, the full Bayesian Broadcast is a bit demanding to implement, so in section III we provide a very simple practical implementation based on these ideas, which we call "Pseudo-Bayesian Broadcast". In section IV we present some very encouraging experimental results on the average backlog when using Pseudo-Bayesian Broadcast. In section V we discuss a recursive implementation of this idea which uses non-global information. In section VI we discuss the application of Pseudo-Bayesian Broadcast to other network models (such as the Ethernet).

II. BAYESIAN BROADCAST

Let R_t denote the number of active stations at time t (they are *ready* to transmit a packet). This value will decrease with successes, and increase when a processor gives an inactive station a packet.

To motivate our development, we begin by considering the "complete knowledge" case where each station knows the value of R_t before slot t begins. This is unrealistic, since R_t can not be determined from the available information, but it is of interest to determine how the stations should act in this case.

How likely is it to have a hole, success, or collision for a given broadcast probability b_t (and *waiting probability* $w_t = 1 - b_t$) and given value $R_t = r$? The probabilities are:

$$P(\text{hole} \mid R_t = r) = H_{b_t}(r) = w_t^r, \tag{1}$$

$$P(\text{success} \mid R_t = r) = S_{b_t}(r) = r \cdot b_t \cdot w_t^{r-1}, \tag{2}$$

$$P(\text{collision} \mid R_t = r) = C_{b_t}(r) = 1 - H_{b_t}(r) - S_{b_t}(r). \tag{3}$$

The optimal value for b_t is:

$$b_t = 1/R_t; \tag{4}$$

this maximizes $S_{b_t}(R_t)$. Note that b_t depends only on R_t.

If b_t is chosen optimally as $1/R_t$, the expected number of stations attempting to transmit will be one, and the probabilities of holes, successes, and collisions will be:

$$H_{\frac{1}{R_t}}(R_t) = (1 - \frac{1}{R_t})^{R_t} \approx \frac{1}{e}, \tag{5}$$

$$S_{\frac{1}{R_t}}(R_t) = (1 - \frac{1}{R_t})^{R_t - 1} \approx \frac{1}{e} \tag{6}$$

$$C_{\frac{1}{R_t}}(R_t) = 1 - H_{\frac{1}{R_t}}(R_t) - S_{\frac{1}{R_t}}(R_t) \approx 1 - \frac{2}{e}. \tag{7}$$

(The approximations hold for large R_t.)

However, the stations will typically not know the correct value for R_t. For example, some inactive stations may have been given newly generated packets during slot $t - 1$.

In our first procedure, which we call the Bayesian Broadcast algorithm, each station will use the evidence available evidence up to time time t to estimate the likelihood $p_{r,t}$ that $R_t = r$ for each $r \geq 0$. That is,

$$p_{r,t} = P(R_t = r), \text{ for } r = 0, \ldots \tag{8}$$

given the available evidence. We call our procedure "Bayesian Broadcast", since it relies on Bayesian reasoning to estimate $\pi_t = (p_{0,t}, \ldots)$.

Initially, each station begins with the distribution $\pi_0 = (1, 0, 0, \ldots)$ – it assumes that all stations are inactive. Each station will compute the *same* vector π_t using the available *global* information. The vector $\pi_t = (p_{0,t}, \ldots)$ summarizes the global information available about R_t.

With the Bayesian Broadcast procedure, each station performs the following four steps during each time slot:

- Compute the optimum transmission probability b_t from the initial probability vector π_t.
- Transmit a packet (if one needs to be sent) with probability b_t.
- Perform a Bayesian update of π_t (the initial probability distribution for R_t) to obtain χ_t (the final probability distribution for R_t), using the evidence (hole, success, or collision) observed in time slot t.
- Convert the final probabilities χ_t for R_t into initial probabilities π_{t+1} for R_{t+1} by considering the generation of new packets and the fact that a packet may have been successfully transmitted during time slot t (i.e. modelling the flow of packets into and out of the system).

In subsections II.A – II.C below we consider the details involved in the preceding steps.

II.A. Computing the Broadcast Probability

One can choose b_t to maximize the expected chance of a success, even though there is uncertainty about R_t as summarized in π_t, since

$$E(P(\text{success at time t})) = \sum_r p_{r,t} \cdot S_{b_t}(r). \tag{9}$$

Given π_t, this is a polynomial in the unknown variable b_t. In practice there would be at most a finite number of nonzero coefficients at any time. We can compute the value $\hat{b}_t$ which maximizes (9) by differentiating and root finding (or by Fibonacci search [GKHK75] if (9) is known to be a unimodal function of b_t).

In practice the computation required to find the optimum b_t will probably be excessive. One might use shortcuts such as computing b_t only every so often, or approximating it by $(E(R_t))^{-1}$. However, we believe that in practice the "Pseudo-Bayesian Broadcast" algorithm to be described later will be the best choice.

This section has described how to compute the optimum broadcast probability b_t from π_t, the globally known initial probability distribution for R_t.

II.B. Bayesian Updating of the Probability Vector

We now describe how each station computes its *final* probability distribution for R_t, given that slot t was a hole, a success, or a collision.

This problem is ideally suited for an application of Bayes' Rule:

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}. \tag{10}$$

(The final probability $P(H \mid E)$ of a hypothesis H, after evidence E is received, is equal to the initial probability $P(H)$ of H times the probability $P(E \mid H)$ that E will occur given H, divided by the overall probability $P(E)$ of evidence E.)

We have a hypothesis "$R_t = r$", for each $r \geq 0$. The values $p_{r,t}$ are the initial probabilities of these hypotheses before the evidence from time slot t is considered.

Let $q_{r,t}$ denote the final probability $P(R_t = r \mid E_t)$ where E_t is the slot t evidence (hole, success, or collision), and let

$$\chi_t = (q_{0,t}, q_{1,t}, \ldots) \tag{11}$$

denote the corresponding final probability vector. The $q_{r,t}$'s are easily obtained using Bayes' Rule by multiplying each initial probability $p_{r,t}$ by the appropriate likelihood $H_{b_t}(r), S_{b_t}(r)$, or $C_{b_t}(r)$ according to whether a hole, success, or collision was observed, and then normalizing so that the $q_{r,t}$'s add up to one.

This completes our description of how each station incorporates the slot t evidence into its probability distribution for R_t. The resulting distribution should make the best possible use of the globally available information; it is hard to imagine improving over this application of Bayes' Rule.

II.C. Converting the final probabilites χ_t into the initial probabilities π_{t+1}

Finally, we convert χ_t, the final probability distribution for R_t, into an initial distribution for R_{t+1}. Why $p_{i,t+1}$ might be different than $q_{i,t}$? First, if slot t was a success, the expected number of active stations will decrease by one. Second, we expect some inactive stations to receive new packets from their processors during slot t, so the expected number of active stations will increase for this reason.

II.C.1 Modelling Successful Packet Transmission

We model the effect of successes as follows. We let $u_{r,t}$ denote a station's estimate of the probability that the number of active stations is r, taking into account the evidence from the channel, including the effect of a success on the number of active stations.

If time slot t contained a success, then we set

$$u_{r,t} = q_{r+1,t}, \text{ for } r = 1, \ldots, \tag{12}$$

otherwise we set

$$u_{r,t} = q_{r,t}. \text{ for } r = 0, \ldots. \tag{13}$$

The vector $\rho_t = (u_{0,t}, \ldots,)$ is used as input into the next step, where the generation of new packets is taken into account.

II.C.2 Modelling the Generation of New Packets

The are many ways to model the generation of new packets. We are actually concerned with the rate at which stations "become active", i.e. convert from having no packets to send at time t to having a packet to send at time $t + 1$. We consider two approaches.

II.C.2.a No Modelling of Packet Generation

Here we will rely on the Bayesian updating to keep the probability vector reasonably accurate. This is the simplest case. We simply set $\pi_{t+1} = \rho_t$.

II.C.2.b Poisson Model of Packet Generation

Here we assume that new packets arrive according to a Poisson distribution with parameter α, and that α is estimated reasonably accurately. We assume that new packets are given to inactive processors. We can compute the initial probabilities for R_{t+1}:

$$p_{r,t+1} = \sum_{j=0}^{i} u_{j,t} \cdot P_\alpha(r-j). \tag{14}$$

Here $P_\alpha(r-j)$ denotes the value of the Poisson density function at point $r-j$; i.e. the probability that $r-j$ new packets will arrive during a time slot.

This completes our description of the Bayesian Broadcast procedure, since we now have our initial estimates for the distribution of R_{t+1} for the next time slot.

III. THE PSEUDO-BAYESIAN BROADCAST ALGORITHM

We now present a practical implementation of the above ideas, which we call the Pseudo-Bayesian Broadcast algorithm.

We derive this algorithm by assuming that π_t can be reasonably approximated by a Poisson distribution with mean λ; the station's value of λ at time t represents the station's estimate of R_t. (We use the notation λ rather than the subscripted form λ_t for convenience in this section: λ now denotes a changeable control parameter for the stations.) Let

$$P_\lambda(r) = \frac{e^{-\lambda} \cdot \lambda^r}{r!} \tag{15}$$

denote the Poisson density at r for Poisson parameter λ. Each station will keep only λ, rather than the vector π_t, and will approximate the initial probability $p_{r,t}$ by $P_\lambda(r)$.

To develop the "Pseudo-Bayesian" broadcast and probability updating procedure, we first consider the equations that would be used for a true Bayesian update if b_t is the actual broadcast probability (and $w_t = 1 - b_t$). (These equations represent the unnormalized final probability values.)

$$P_\lambda(r) \cdot H_{b_t}(r) = e^{-\lambda b_t} \cdot P_{\lambda w_t}(r) \tag{16}$$

$$P_\lambda(r) \cdot S_{b_t}(r) = \lambda b_t \cdot e^{-\lambda b_t} \cdot P_{\lambda w_t}(r-1) \tag{17}$$

$$P_\lambda(r) \cdot C_{b_t}(r) = P_\lambda(r) \cdot (1 - H_{b_t}(r) - S_{b_t}(r)) \tag{18}$$

From (9) and (17) it is easy to compute the optimal broadcast probability:

$$b_t = \min(\frac{1}{\lambda}, 1); \tag{19}$$

no complicated root-finding is needed. Thus we have derived our first practical benefit from the Poisson approximation: the optimal value for b_t (given the approximation) is trivial to compute.

We next consider the problem of updating λ in as Bayesian a manner as possible, while preserving our Poisson approximation. We shall see that for holes and successes we can use Bayes Rule exactly, while for collisions we must introduce an approximation error in order to preserve the Poisson approximation.

From (16) we see that for holes the Bayesian updating takes a simple form, since the resulting distribution will also be Poisson with mean $\lambda w_t = \min(\lambda - 1, 0)$. In other words, when a hole occurs the stations reduce their estimate of the expected number of active stations by one, unless λ is already less than 1, in which case they set λ to zero.

From (17) we see that for successes the Bayesian updating and success modelling also takes a simple form. Here (17) will yield a Poisson distribution with mean $\lambda - 1$ shifted one place to the right. However, the effect of modelling a successful transmission shifts the distribution one place to the left. The net result is that the Poisson assumption remains valid, and each station should decrement its state variable λ by 1.

If there is a collision, Bayes Rule will not yield a Poisson distribution for the final probabilities. However, we approximate the result by a Poisson distribution, by setting λ to be the mean of the resulting distribution, which is (using x to denote $\lambda \cdot b_t$):

$$\lambda + \frac{x^2}{e^x - x - 1} \tag{20}$$

which simplifies in the case $\lambda \geq 1, b_t = \frac{1}{\lambda}$ to:

$$\lambda + \frac{1}{e - 2}. \tag{21}$$

(It is somewhat surprising that we get a *constant* increment to λ in this case.) For $\lambda \leq 1$ (20) is reasonably well approximated by

$$2.39221 \tag{22}$$

which is the value (20) yields for $\lambda = 1$. Using (22) is equivalent requiring that $\lambda \geq 1$ at all times. The following algorithm makes this simplification.

We now summarize the above analysis, assumptions, and approximations in the following presentation of the "Pseudo-Bayesian Broadcast" algorithm.

The Pseudo-Bayesian Broadcast Procedure:

Each station maintains a copy of λ, and during each slot:

- Broadcasts with probability $\frac{1}{\lambda}$, if it has a packet.
- Decrements λ by 1 if the current slot is a hole or a success, and increments λ by $\frac{1}{e-2} = 1.392211\ldots$ if the current slot is a collision.
- Sets λ to $\max(\lambda + \alpha, 1)$, where α is the observed average success rate (which is also the average arrival rate for new packets).

We note that since each station now only maintains a single parameter λ, it would be simple to broadcast λ with every packet. In this way stations which have just powered-up can "synchronize" easily.

IV. EXPERIMENTAL RESULTS

The Pseudo-Bayesian Broadcast Procedure was simulated for 10^6 trials for a number of different Poisson arrival rates α. The following results were obtained. Here R_α^{ave} denotes the average value of R, and R_α^{max} denotes the observed maximum value of R:

α	R_α^{ave}	R_α^{max}
0.10	0.044	7
0.15	0.13	10
0.20	0.32	18
0.25	0.74	19
0.30	2.13	41
0.32	3.42	54
0.34	7.17	101
0.35	13.04	113
0.36	30.65	217
0.37	1014.7	1763

It is clear from (5) that we should not expect to be able to handle $\alpha >> e^{-1} = 0.3678\ldots.$. We see that the algorithm becomes unstable for $\alpha > e^{-1}$, as expected.

These results are superior to those of previously published statistics for adaptive control algorithms that use only global information. For the algorithm of Hajek and van Loon [HVL82] it is reported that the average backlog for $\alpha = 0.32$ is approximately 5.0. (To compare our results with theirs, you should *subtract* λ from our values of R_α^{ave}, since they do not count newly arrived packets in the backlog.) To be fair, we note that their main objective was to *prove* that their algorithm was stable for $\alpha < e^{-1}$; we do not yet have such a proof for Pseudo-Bayesian Broadcast.

For another comparison, in [GGMM85] it is reported that for Binary Exponential Backoff

$$R_{0.20}^{ave} = 0.44, R_{0.25}^{ave} = 1.483, R_{0.30}^{ave} = 13.526, \text{ and } R_{0.35}^{ave} = 12519.4.$$

Some practical data relating to the performace of Binary Exponential Backoff can be found in [SH80].

For our simulation results the stations model the input arrival rate, and increase λ by their estimated α in each slot. We were curious what would happen if this was omitted (as suggested in II.C.2.a). The following simulation results were obtained for this case:

α	R_α^{ave}	R_α^{max}
0.30	3.34	56
0.32	8.22	96
0.34	85.64	471
0.35	3809.32	7530

This version of the algorithm performs noticeably worse; it doesn't even look stable at $\alpha = 0.35$.

V. RECURSIVE PSEUDO-BAYESIAN BROADCAST

The simple Pseudo-Bayesian Broadcast procedure will not be stable for $\alpha > e^{-1} = 0.3678\ldots$. Yet transmission control procedures are known for the slotted ALOHA model which are stable for significantly higher values of α. The best result to date is an algorithm which is provably stable for $\alpha < 0.4878$. The history of this algorithm is involved, but the ideas and analysis are due to Capetanakis, Gallager, Tsybakov, Mikhailov, Likhanov, Huang, Berger, Mosely, and Humblet, among others (see [Ga85] for a presentation and references). Pippenger has even shown that if the stations can determine exactly how many stations were transmitting during a collision, then any rate $\alpha < 1$ can be handled [Pi81].

The ideas involved in these algorithms can be easily adapted for use here. We present below the sketch of a "recursive" Pseudo-Bayesian Broadcast algorithm.

The idea is that whenever a collision occurs, the stations involved in that collision invoke a "recursive" execution of the (recursive) Pseudo-Bayesian Broadcast algorithm, beginning with a new λ initialized to 2.392211 (an estimate for the expected number of stations involved in the collision). A recursive execution may of course invoke another recursive execution of its own, if it encounters a collision. A recursive execution terminates when its λ drops below 1. When a recursive execution terminates, the "parent" execution resumes, after adjusting its lambda to account for the number of successes observed on the channel since the recursive execution began. Note that the recursive execution may terminate *before* all of the stations involved in the collision have successfully transmitted their packets, in which case those stations that joined the recursive execution but which didn't get a packet sent successfully rejoin the parent execution. All stations can determine when a recursive execution terminates, since the recursive λ is updated only on the basis of globally available information. The lowest-level or initial execution of the procedure is slightly different, in that stations joining the system only join at the lowest level, and also in that when a recursive execution terminates and control returns to the lowest-level, the lowest level λ needs to be incremented by the expected number of new stations that have joined the system since the recursive execution began.

We omit our preliminary simulation results here, although they look very promising. (We hope to present these results in a later version of this paper, after more data is collected.) We expect that this recursive procedure should be stable for significantly larger values of α than the simple Pseudo-Bayesian Broadcast. It would of course be interesting to prove such a result about the Recursive Pseudo-Bayesian Broadcast algorithm.

VI. QUEUES AND VARIABLE-LENGTH PACKETS

In practice, a station might have more than one packet ready to transmit, and would keep its untransmitted packets in a queue. In this case we suggest that it could indicate in each packet it sends whether it wishes to reserve the next time slot for a subsequent packet.

With this approach there are then two kinds of slots: *contention* slots (as usual) and *reserved* slots. A contention slot followed by a sequence of reserved slots we term an *interval.* During an interval the transmitting station will completely empty its queue, making maximum use of the bandwidth available.

We can modify the Pseudo-Bayesian Broadcast algorithm to work in this case by doing the Bayesian updating by intervals rather than by slots: each station will increase λ by 1.39221 when there is a collision during a contention slot, and decrease λ by 1.0 if there is a hole or a success in a contention slot. During a reserved slot no Bayesian updating of λ is performed.

Here λ is modelling the number of stations with nonempty queues, *not* the number of packets backlogged in the system.

The "arrival rate" is then the average rate at which which stations with empty queues receive a packet to transmit per time slot. (The arrival rate is measured per slot and not per interval.) This can be estimated by computing the average number of intervals per time slot. During each time slot each station can increase its λ by this amount.

We observe that the above strategy is now applicable to networks where there are both queues and variable-length packets (such as the Ethernet), since we can define a "slot" to be the time period required for channel acquisition or collision detection, and consider the first slot of a long packet to "reserve" the time necessary to transmit the rest of the packet. (This may or may not work out well in practice; maintaining accurate slotting with such a fine resolution may be impractical.)

VII. OPEN PROBLEMS

Determine for what arrival rates α will the Bayesian Broadcast, Pseudo-Bayesian Broadcast, and Recursive Pseudo-Bayesian Broadcast algorithms remain stable (i.e. have finite expected backlog).

For the Bayesian Broadcast algorithm, determine the conditions under which it is true that if a station joins the system late, the initial probability vector π_t it uses shouldn't matter much, since it will converge to the values held by the other stations after a short while.

For the Bayesian Broadcast algorithm, demonstrate that, given the best possible estimates for R_t available from the past history, the described procedure for choosing the broadcast rate is optimal. The described procedure is *locally* optimal in that it maximizes the success rate at each slot. However, it might not be *asymptotically* optimal in that one could obtain an advantage by choosing b_t locally non-optimally in order to gain information for later slots. I conjecture that for reasonable models of packet generation the proposed procedure for finding for the broadcast probability is optimum in that one can't expect to do better over the long term with any other procedure that uses the same information.

For the Bayesian Broadcast algorithm, identify the most general conditions under which P(success at time t) will be unimodal in b_t, so that the optimal value of b_t can be determined by Fibonacci search.

Extend the Bayesian approach here to handle the case that a station only obtains information from the channel when it itself tries to transmit a packet. (This of course would mean that stations would be using non-global information.)

Identify the *best* control procedure when stations use all available information (including private information). It is not clear how to do this, even for as few as three stations, since the stations may no longer have equal values for the transmission probability b_t, and computing b_t for one station requires information about what values for b_t the other stations are likely to use, which requires knowing what information they is using about what b_t the first station will use, etc.

VIII. CONCLUSIONS

We believe the proposed Pseudo-Bayesian Broadcast procedure will be found to be exceptionally effective in practice, since it makes nearly the "best possible use" of the information available on the network in determining the broadcast probabilities to use.

Acknowledgments

I would like to thank Robert Gallager, Pierre Humblet, Tom Leighton, Charles Leiserson, and Bard Bloom for helpful discussions on this topic, and Brian Rogoff for helping to obtain the simulation results.

References

[Ab73] Abramson, N., "Packet Switching with Satellites," Proc. AFIPS National Computer Conference **42** (1973), 695-702.

[Ga85] Gallager, R. G., "A Perspective on Multiaccess Channels," IEEE Tran. Info. Theory **IT-31** (March 1985), 124-142.

[GKHK75] Gellert, W., H. Kustner, M. Hellwich, and H. Kastner (Editors). The VNR Concise Encyclopedia of Mathematics (Van Nostrand, 1975), 640-642.

[GK77] Gerla, Mario and Leonard Kleinrock, "Closed Loop Stability Controls for S-ALOHA Satellite Communications," Proc. Fifth Data Communications Symposium (Snowbird, Utah, Sept. 1977), 2.10 – 2.19.

[GGMM85] Goodman, J., A. Greenberg, N. Madras, and P. March, "On the Stability of the Ethernet," Proc. 17th ACM Symposium on Theory of Computing (Providence, 1985), 379-387.

[HVL82] Hajek, Bruce, and Timothy van Loon, "Decentralized Dynamic Control of a Multiaccess Broadcast Channel," IEEE Trans. Automatic Control **AC-27**,(June 1982), 559-569.

[Kl75] Kleinrock, Leonard, Queueing Systems. Volume I: Theory. (Wiley, New York, 1975).

[Ma85] Massey, James L. (ed.) *Special Issue on Random-Access Communications,* IEEE Trans. Infor. Theory **IT-31**, (March 1985).

[MB76] Metcalfe, Robert M. and David R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," CACM **19** (July 1976), 395-404.

[Pi81] Pippenger, N. "Bounds on the performance of protocols for a multiple-access broadcast channel," IEEE Trans. Infor. Theory **IT-31**, (Mar. 1981), 145-151.

[SH80] Shoch, J., and Jon A. Hupp, "Measured Performance of an Ethernet Local Network," CACM **23** (Dec. 1980), 711-721.

[Ta76] Tanenbaum, Andrew S. "Network Protocols," Computing Surveys **13** (Dec. 1981), 453-489.