A FAST DIVIDE AND CONQUER PROTOCOL FOR CONTENTION RESOLUTION ON BROADCAST CHANNELS

by

Karl Abrahamson

Technical Report 85-3

April 1985

A FAST DIVIDE AND CONQUER PROTOCOL FOR CONTENTION RESOLUTION ON BROADCAST CHANNELS

Karl Abrahamson
Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1W5
Canada

Abstract

We describe a contention resolution protocol for an ethernet-like broadcast channel. The protocol is based on tree algorithms, particularly that of Greenberg. We show how to obtain a simpler and more accurate estimate of the number of contending stations than Greenberg's method, and use the new estimation method to obtain an improved protocol.

1. Introduction

Greenberg and Ladner [3] describe an abstract model of a broadcast network. The model had previously been studied in [1,5,6,7,8], and subsequently by [4,9]. Greenberg and Ladner describe a protocol for efficient and fair sharing of the broadcast channel. This paper describes an elegant modification of their protocol which improves both its efficiency and its practicality.

In the next seciton, the model of a network is described. Following that, we discuss a divide and conquer paradigm for designing protocols, and describe some known divide and conquer protocols, including Greenberg and Ladner's. In section 5 the new protocol, also a divide and conquer protocol, is described. Finally, the protocols are analyzed and compared.

2. The Model

Our model is an abstraction of an ethernet-like broadcast network. Our notation is borrowed from Willard [9]. There are N stations connected to a single broadcast channel. Time is discrete. At each time slot, each station may choose either to broadcast or to listen. In each time slot, one of three possible events occurs: 0 (no station broadcasts), 1 (exactly one station broadcasts) or e (two or more stations broadcast). Each station knows the events which occurred in time slots 0,...,k-1 before it decides whether to broadcast or listen in slot k.

Messages are generated by the stations in some unknown random fashion. Each station must successfully transmit its messages. A 1 event corresponds to the successful transmission of a message. An e event signifies an interval of contention for the channel, during which no messages are successfully transmitted.

Our goal is to develop a protocol for channel access which a) gives each station fair access to the channel, and b) obtains a high expected proportion of time slots containing 1 events.

Our definition of fairness is as follows: A protocol is k-fair if it permits any given station at most k successful transmissions during any time interval in which a) some station x does no successful transmissions, and b) station x desires to send a message throughout the interval. All of the protocols discussed here are 2-fair.

The protocols considered in this paper are probabilistic. Each strives for a high expected throughput, averaged over the random values that it uses. Throughput is discussed in section 6.

3. Divide and Conquer Protocols

It is most convenient to present a protocol as if it were being executed by the channel, rather than by each of the stations. Imagine the channel maintaining sets of stations, and causing all of the members of a given set to broadcast in a given time slot.

Of course, in reality a protocol must be executed by the individual stations. It must therefore be designed in such a way that each station can,, by simulating the channel, determine what its own behaviour should be. Each station may make use of locally available information and the sequence of events which has occurred on the channel. Two operations, conceptually executed by the channel, but which can be simulated by each station, are

a) $(A_1,...,A_n) :=$ partition$(S,p_1,...,p_n)$. Here, $S,A_1,...,A_n$ are sets of stations, and $p_1,...,p_n$ are probabilities which sum to 1. The sets $A_1,...,A_n$ form a partition of S, and are computed by independently placing each member of S into a randomly chosen A_i.

A_i is chosen with probability p_i, $i = 1,...,n$.

b) Broadcast(S). Cause each member of S to broadcast during the next time slot, and advance the clock one unit. Future actions may depend on the event caused by Broadcast(s); that is, on whether $|S| = 0$, $|S| = 1$ or $|S| \geq 2$.

A divide and conquer protocol proceeds in a series of *sessions*. At the beginning of a session, each station with a message to send is placed in a set S. Then SATISFY(S) is executed. SATISFY(S) causes each member of S to send exactly one successful message. When SATISFY(S) is finished, the next session begins. By listening to the events which occur on the channel, and simulating the execution of SATISFY(S), every station (including those not in S) can determine exactly when SATISFY(S) is finished.

The basic design for SATISFY is as follows. For convenience later on, we add an output parameter n to SATISFY, which is set equal to |S|.

```
SATISFY(S,n):
        Broadcast(S);
            on 0: n:=0;
            on 1: n:=1;
            on e: RESOLVE(S,n)
end.
```

RESOLVE(S,n):

1. Choose a number m and probabilities $p_1,...,p_m$, in such a way that all stations agree on their values;
2. $(A_1,...,A_m)$:= partition$(S,p_1,...,p_m)$;
3. For i := 1 to m do SATISFY'(A_i,n_i);

4. $n := n_1 + ... + n_m$

end.

Protocol SATISFY′ may be SATISFY itself, or may be some other protocol. A divide and conquer protocol is determined by its choice of $m, p_1, ..., p_m$, and its choice of protocol SATISFY′.

In the next two sections, we describe several different versions of SATISFY. Some do not fit exactly into the framework given in this section, but require small modifications.

3. Previously Known Protocols

This section describes three divide and conquer protocols: Basic Binary Tree [1,8], Improved Binary Tree [6,7,8] and m-ary Tree [2,3,4].

3.1. Basic Binary Tree (BBT)

BBT is the simplest divide and conquer protocol. Simply choose $m = 2$, $p_1 = p_2 = 0.5$, and SATISFY′ = SATISFY. Figure 1 shows a possible execution of BBT with $|S| = 4$. The tree represents the recursion structure of an execution.

3.2. Improved Binary Tree (IBT)

Consider the procedure RESOLVE(S,n) with $m = 2$:

(A_1, A_2) := partition(S, p_1, p_2);

SATISFY (A_1, n_1);

SATISFY (A_2, n_2);

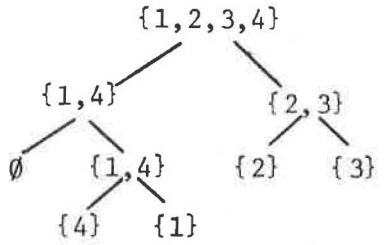

Figure 1. A possible execution of BBT with n = 4.
The sequence of events is ee0e11e11.

Suppose a given execution of partition chooses $A_1 = \emptyset$, $A_2 = S$, as will occasionally happen. Then $n_1 = 0$. Without any further action, $A_2 = S$ can be inferred. Furthermore, RESOLVE(S,n) is only called when $|S| \geq 2$. So $|A_2| \geq 2$ is inferred, and Broadcast(A_2) is guaranteed to produce an e event. That broadcast could be saved by replacing SATISFY(A_2,n_2) by RESOLVE(A_2,n_2), whenever $n_1 = 0$.

Notice that no such savings is possible when $A_1 = S$ and $A_2 = \emptyset$. So $A_1 = \emptyset$ is preferable to $A_2 = \emptyset$. Such a preference makes it desirable to bias the probabilities p_1 and p_2, with $p_2 > p_1$. Hofri [6] shows that the optimum value of p_1 is about .4175.

Protocol IBT is BBT with the above two modifications: skipped broadcasts and biased probabilities.

3.3. M-way Tree (MWT)

MWT uses an *estimation procedure* to obtain an estimate $\hat{n}$ of $n = |S|$, agreed on by all stations. The value of m is chosen to be $\max(1,\lfloor \alpha \hat{n} \rfloor)$ for some fixed constant α; $p_i = 1/m$ for i = 1,...,m. For SATISFY′, MWT uses IBT.

Suppose that $\hat{n}$ is a good estimate of n, and α is close to 1.0. Then the effect of MWT is to partition S into subsets $A_1,...,A_m$ of expected size close to one. With any luck, a good fraction of the subsets will be singleton.

The accuracy of $\hat{n}$ as an estimate of n is important to the efficiency of MWT. If $\hat{n}$ is too large, many of the sets $A_1,...,A_m$ are empty, and MWT generates many 0 events. If $\hat{n}$ is too small, MWT degenerates to IBT.

Greenberg and Ladner propose an estimation procedure. To estimate |S|, perform

$$(A,B) := \text{partition}(S,2^{-i},1-2^{-i});$$

$$\text{Broadcast}(A)$$

for $i = 1,2,...$, until a 0 or 1 event occurs. If the first 0 or 1 event occurs when $i = k$, let $\hat{n} = c2^k$, where c is a bias-counteracting constant (approx. 1.1).

Using the above estimation procedure, with $\alpha \approx 0.9$, does give an improvement over IBT (see section 7). But the estimation procedure produces an estimate $\hat{n}$ with standard deviation greater than .6n [4]. A better estimation procedure might yield a better algorithm.

And indeed, it does. Greenberg et al [4] suggest using the above estimation procedure, but with $(1+\epsilon)^i$ in place of 2^i. With suitable adjustments, they achieve a better estimate, and hence an (asymptotic) improvement in the efficiency of the MWT protocol. Unfortunately, for ϵ and n both small, the time spent getting an estimate greatly exceeds the time it would take to run a simpler protocol, such as BBT. A simpler and more practical estimation procedure is given in the next section.

5. Improved Estimates

Suppose we partition S randomly into two sets A and B of equal expected size. Then we count |A|, *exactly,* and use that as an estimate of |B|. For |S| large, we have a very accurate estimate of |B|, which can be used for the m-way tree algorithm.

How can |A| be counted? Recall that our goal is to SATISFY(S), which can be accomplished by SATISFY(A) followed by SATISFY(B). But, as a side effect, SATISFY(A) computes |A|. The Sibling Estimator (SE) protocol makes use of that side effect. Let SATISFY-IBT and RESOLVE-IBT be the Improved Binary Tree protocol described in the previous section.

```
SATISFY-SE(S,n):
    Broadcast(S);
      on 0: n:=0;
      on 1: n:=1;
      on e: {(A,B):=partition(S,0.5,0.5);
              SATISFY-SE(A,n1);
              if n1 = 0 then RESOLVE-IBT(B,n2)
                        else {m:=max(1,⌊αn1⌋);
                              SOLVE-MWT(B,n2,m)};
              n:=n1+n2}
end.

SOLVE-MWT(S,n,m):
    (A1,...,Am):=parition(S,1/m,...,1/m);
    For i:=1 to m do SATISFY-IBT(Ai,ni);
    n:=n1+...+nm
end.
```

6. Efficiency Measures

The expected number of time slots in a session of SATISFY(S,n) depends only on $n = |S|$ (the *multiplicity* of the session) and on the version of SATISFY used (BBT, IBT, etc.). Let $t^A(n)$ be the expected number of time slots in a session of multiplicity n, for protocol A. Define the *local throughput* of A as

$$T^A(n) = n/t^A(n).$$

$T^A(n)$ is the expected proportion of time slots containing successful transmissions, for sessions of multiplicity n.

The efficiency of a protocol is measured by its *throughput,* which is just the expected proportion of time slots containing successful transmissions, under conditions of heavy load on the channel. The load on the channel may be distributed arbitrarily among the N stations, and a protocol's throughput depends on that distribution. As an extreme example, consider the case when all of the messages are generated by a single station. Then a naive protocol has a throughput of 1.0.

We analyze protocols under two different assumptions about the distribution of the load among the stations.

a) The load is distributed in a completely unknown fashion, and throughput is measured under the worst possible distribution. The *min throughput* of protocol A is defined as

$$T^A_{min} = \inf\{T^A(n): n \geq 1\}.$$

b) N is infinite, and the load is distributed in such a way that, as the number of untransmitted messages grows, so does the number of stations holding untransmitted messages. The *limit throughput* of protocol A is defined as

$$T_{lim}^{A} = \lim_{n\to\infty} \inf \{T^{A}(k): k \geq n\}.$$

The limit throughput is useful under the common assumption [1,2,6,8] that the messages are generated by an infinite Poisson process of fixed intensity λ. For any λ less than the limit throughput, the expected delay between generation and successful transmission of each message is bounded [6].

7. Limit Throughput

The limit throughputs for BBT, IBT, MWT(2) (MWT using Greenberg and Ladner's estimation procedure) and MWT(ϵ) (MWT using the modified $1+\epsilon$ estimation procedure, for $\epsilon = 10^{-4}$) are given below, rounded to four decimal places. Sources for the four results are, respectively, [1,8], [6], [2,3] and [4].

$$T_{lim}^{BBT} = .3465 \tag{1}$$

$$T_{lim}^{IBT} = .3813 \tag{2}$$

$$T_{lim}^{MWT(2)} = .4303 \tag{3}$$

$$T_{lim}^{MWT(\epsilon)} = .4686 \tag{4}$$

Our goal in this section is to analyze the limit throughput of the SE protocol. Recall that the SE protocol involves a parameter α. It is convenient in what follows to assume $1/2 < \alpha < 2$, so that $\max(1,\lfloor \alpha k \rfloor)$ is 1 for $k = 1$, and $\lfloor \alpha k \rfloor$ for $k > 1$. We will see that the optimal value of α is in fact about 0.785.

Let w(n,m) be the expected number of time slots used by SOLVE-MWT(S,n,m). Let $t^{SE}(n)$ and $t^{IBT}(n)$ be the expected number of time slots used by SATISFY-SE(S,n)

and SATISFY-IBT(S,n), respectively. The following equations describe the behaviours of the SE and IBT protocols. Let $p = .4175$, and $q = 1-p$.

$$t^{IBT}(0) = t^{IBT}(1) = 1 \tag{5}$$

$$t^{IBT}(n) = 1 - q^m + \sum_{k=1}^{n} \binom{n}{k} p^k q^{n-k} (t^{IBT}(k) + t^{IBT}(n-k)) \quad (n>1) \tag{6}$$

$$w(n,1) = t^{IBT}(n) \tag{7}$$

$$w(n,m) = m \sum_{k=0}^{n} \binom{n}{k} (\frac{1}{m})^k (1-\frac{1}{m})^{n-k} t^{IBT}(k) \qquad (m>1) \tag{8}$$

$$t^{SE}(0) = t^{SE}(1) = 1 \tag{9}$$

$$t^{SE}(n) = 1+2^{-n} t^{IBT}(n) + n2^{-n} (1+t^{IBT}(n-1)) \tag{10}$$
$$+ \sum_{k=2}^{n} \binom{n}{k} 2^{-n} (t^{SE}(k) + w(n-k,\lfloor \alpha k \rfloor)) \quad (n>1)$$

The quantity $w(n,m)$ can be bounded as follows. Let

$$K_\lambda = \frac{e^{-\lambda}}{\lambda} \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} t^{IBT}(k)$$

Lemma 1. Let $\mu = \lambda n/(n-\lambda)$. Then for $\lambda<n$,

$$w(n,n/\lambda) \leq \frac{\mu}{\lambda} e^{\mu-\lambda} K_\mu n.$$

Proof.

$$w(n,n/\lambda) = \frac{n}{\lambda} \sum_{k=0}^{n} \binom{n}{k} (\frac{\lambda}{n})^k (1-\frac{\lambda}{n})^{n-k} t^{IBT}(k) \qquad \text{by (8)}$$

$$\leq \frac{n}{\lambda} (1-\frac{\lambda}{n})^n \sum_{k=0}^{n} \frac{\lambda^k}{k!} (\frac{n-\lambda}{\lambda})^{-k} t^{IBT}(k)$$

$$\leq \frac{n}{\lambda} e^{-\lambda} \sum_{k=0}^{\infty} \frac{\mu^k}{k!} t^{IBT}(k) \qquad \text{by elementary calculus}$$

$$= \frac{\mu}{\lambda} e^{\mu-\lambda} K_\mu n.$$

□

Corollary 2: As n approaches infinity, for fixed λ, $w(n,n/\lambda)$ is asymptotic to $K_\lambda n$.

Proof. Examination of the proof of lemma 1 shows that asymptotic equality can be proved.

□

Choose an arbitrarily large l, and imagine executing SATISFY-SE(S,n) to l levels of recursion (see figure 2). Let A_i be the left sibling of B_i, and define random variables $a_i = |A_i|$, $b_i = |B_i|$, $i = 1,...,l$. Due to the independence of the partitions at the various levels, a_l is binomially distributed with mean $n2^{-l}$. In the next section we show that $t^{SE}(n) = O(n)$. Hence, the expected time use by SATISFY-SE($A_b a_l$) is

$$t^{SE}(a_l) \leq E(c \cdot a_l)$$

$$= cn2^{-l} \qquad (12)$$

for some constant c.

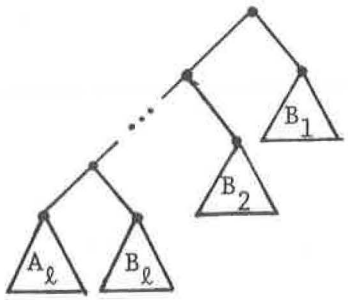

Figure 2. Execution of SATISFY-SE to l levels.

The cost of satisfying the B_i's may be counted as follows. For any $r>1$, let $P_{ub}(n,l,r)$ be the probability that at least one of the l partitions shown in figure 2 is unbalanced by more than a factor of r; that is, $P_{ub}(n,l,r) = \text{Prob}(\text{for some i, either } ra_i < b_i \text{ or } a_i > rb_i)$. Then

$$E(\sum_i \text{cost of } B_i) = P_{ub}(n,l,r)\, E\left(\sum_i w(b_i^{(1)}, \lfloor \alpha a_i^{(1)} \rfloor)\right)$$

$$+ (1-P_{ub}(n,l,r))\, E\left(\sum_i w(b_i^{(2)}, \lfloor \alpha a_i^{(2)} \rfloor)\right),$$

where $a_i^{(1)}$, $b_i^{(1)}$, $a_i^{(2)}$ and $b_i^{(2)}$ are appropriately distributed random variables. We show in the next section that $w(n,m) = O(n+m)$. Choose λ and $\mu = \lambda n/(n-\lambda)$ such that

$$\frac{\mu}{\lambda} e^{\mu-\lambda} K_\mu = \max_{\frac{1}{\alpha r} \le \lambda \le \frac{r}{\alpha}} \frac{\mu}{\lambda} e^{\mu-\lambda} K_\mu$$

Then, by lemma 1,

$$E(\sum_i \text{cost of } B_i) \le P_{ub}(n,l,r) \sum_i c(b_i + \alpha a_i) + \sum_i \frac{\mu}{\lambda} e^{\mu-\lambda} K_\mu b_i$$

$$\le 2(c+\alpha l) P_{ub}(n,l,r) n + \frac{\mu}{\lambda} e^{\mu-\lambda} K_\mu n \qquad (13)$$

for some constant c. For any fixed l and $r>1$, $\lim_{n\to\infty} P_{ub}(n,l,r) = 0$. Hence, there must be

functions $\ell(n)$ and $r(n)$ such that $\lim_{n\to\infty} \ell(n) = \infty$, $\lim_{n\to\infty} r(n) = 1$ and $\lim_{n\to\infty} \ell(n) P_{ub}(n,\ell(n),r(n)) = 0$. From those functions, and inequalities (12) and (13) we obtain

Lemma 3. As n approaches infinity, $t^{SE}(n)$ is asymptotic to $K_{1/\alpha} \cdot n$.

Corollary 4. $T_{lim}^{SE} = 1/K_{1/\alpha}$.

Figure 3 shows a graph of $K_{1/\alpha}$ vs α. The minimum occurs at approximately $\alpha = .785$, where $K_{1/\alpha} \approx 2.134$ and $T_{lim}^{SE} \approx .4686$. It is no coincidence that $T_{lim}^{SE} \approx T_{lim}^{MWT(\epsilon)}$; SE uses MWT with asymptotically exact estimates, while the estimate used by MWT(ϵ) approaches asymptotic exactness as ϵ approaches zero.

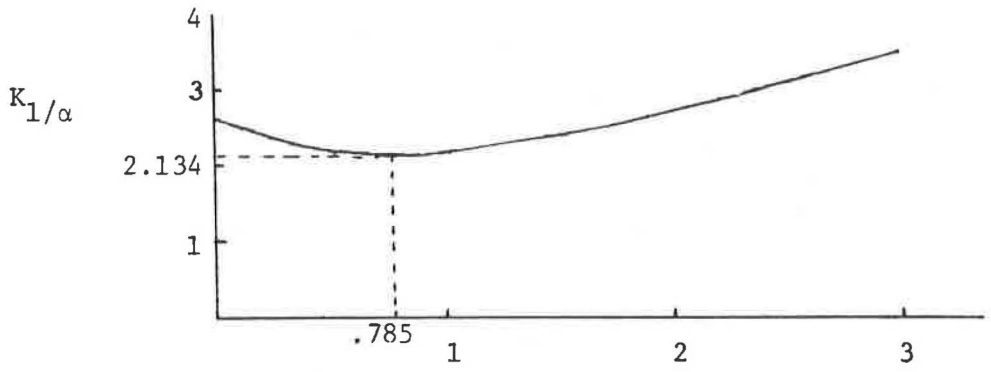

Figure 3: $K_{1/\alpha}$ vs. α.

8. Min Throughput

Equations (5) through (10) provide a basis for computing $t^{SE}(n)$ for small values of n. Figure 4 shows the results of such a computation, as well as similar results for IBT

and MWT(2). The computation indicates the following (rounded to four decimal places):

$$T_{min}^{IBT} = T_{lim}^{IBT} = .3813.$$

$$T_{min}^{MWT(2)} = T^{MWT(2)}(2) = .3166.$$

$$T_{min}^{SE} = T^{SE}(4) = .4009$$

It remains only to demonstrate that the observed minima are the true global minima. For IBT, the result is easy. We are willing to give MWT(2) the benefit of the doubt; certainly, the true min throughput is no better than given above. The remainder of this section is devoted to showing that $T^{SE}(n) > .416$ for $n \geq 200$. The reader who is convinced by Figure 4 may skip to the next section.

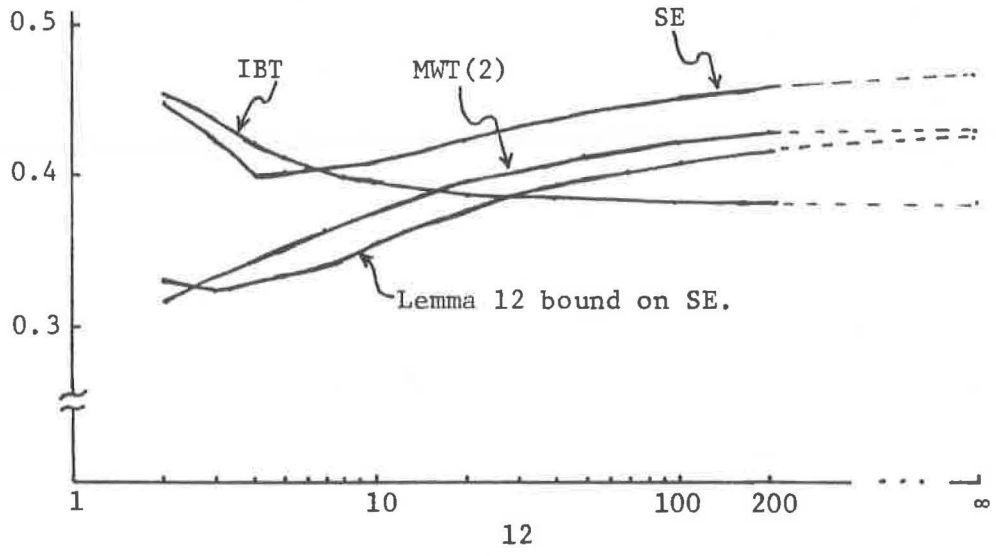

Figure 4. Local throughputs of three protocols, and the bound of Lemma 12.

We begin with some lemmas whose proofs are only sketched.

Lemma 5. For $n>0$, $t^{IBT}(n) \leq 8/3\, n - 5/6$.

Proof. See [7].

Lemma 6. For $n,m>0$, $w(n,m) \leq 8/3\, n+m- 11/6$.

Proof. The worst case in m-way split occurs when all of the partitions but one are empty.

Lemma 7. For $n>0$ and $1/2 \leq \alpha<2$, $t^{SE}(n) \leq (\frac{8}{3}+\alpha)n$.

Proof. Straightforward induction on n, using equation (10) and lemmas 5 and 6.

Lemma 8. For $0<\epsilon<1$, $(1+\epsilon)^{1+\epsilon}(1-\epsilon)^{1-\epsilon}>1+\epsilon^2$.

Proof. Elementary calculus.

We intend to break the binomial distribution $\binom{n}{k}2^{-n}$ into a middle region ($b \leq k \leq n-b$) and a tail region ($k<b$ or $k>n-b$), where $b = (\frac{1-\epsilon}{2})n$, for some suitably chosen ϵ. Throughout the following, let $b = (\frac{1-\epsilon}{2})n$. We begin by bounding the tail.

Lemma 9. $\sum_{k=0}^{b} \binom{n}{k} \leq \binom{n}{b}(\frac{1-\epsilon}{2\epsilon})$.

Proof. Let $x^{\underline{y}}$ denote the decending power $x!/(x-y)!$.

$$\sum_{k=0}^{b} \binom{n}{k} = \sum_{k=0}^{b} \binom{n}{b} \frac{b^{\underline{b-k}}}{(n-k)^{\underline{b-k}}}$$

$$\leq \binom{n}{b} \sum_{k=0}^{b} \left(\frac{b}{n-b}\right)^{b-k}$$

$$\leq \binom{n}{b} \sum_{j=0}^{b} \left(\frac{1-\epsilon}{1+\epsilon}\right)^{j} \qquad \text{since } b = \left(\frac{1-\epsilon}{2}\right)n$$

$$< \binom{n}{b}\left(\frac{1+\epsilon}{2\epsilon}\right).$$

□

Define

$$A_{\epsilon,n} = \frac{.42}{\epsilon}\left(\frac{1+\epsilon}{n(1-\epsilon)}\right)^{1/2}\left(\frac{1}{1+\epsilon^2}\right)^{n/2}.$$

Lemma 10. $\sum_{k=0}^{b} \binom{n}{k} 2^{-n} \leq A_{\epsilon,n}$ for $n \geq 2$.

Proof.

$$\sum_{k=0}^{b} \binom{n}{k} 2^{-n} \leq \binom{n}{b}\left(\frac{1+\epsilon}{2\epsilon}\right)2^{-n} \qquad \text{(by lemma 9)}$$

$$\leq \left(\frac{1+\epsilon}{2\epsilon}\right)2^{-n}\left(\frac{e^{1/24}}{\sqrt{2\pi}}\right)\left(\frac{n^{n+1/2}}{b^{b+1/2}(n-b)^{n-b+1/2}}\right) \qquad \text{(by Stirling's formula)}$$

$$\leq \frac{.42}{\epsilon}\left(\frac{1+\epsilon}{1-\epsilon}\right)^{1/2} n^{-1/2}\left(\frac{1}{(1+\epsilon)^{1+\epsilon}(1-\epsilon)^{1-\epsilon}}\right)^{n/2} \qquad \text{since } b = \left(\frac{1-\epsilon}{2}\right)n.$$

An application of lemma 8 gives the desired result.

□

Define

$$\text{tail}_{\epsilon,n} = \left(\frac{8}{3} + \alpha\right)A_{\epsilon,n-1}$$

Lemma 11.

a) $\sum_{k=2}^{b} \binom{n}{k} 2^{-n}(t^{SE}(k)+w(n-k,\lfloor \alpha k \rfloor)) \leq n \cdot tail_{\epsilon,n}/2$

b) $\sum_{k=b+1}^{n} \binom{n}{k} 2^{-n}(t^{SE}(k)+w(n-k,\lfloor \alpha k \rfloor)) \leq n \cdot tail_{\epsilon,n}/2$

Proof. The two sums have similar proofs. We prove (a) only.

$$\sum_{k=2}^{b} \binom{n}{k} 2^{-n}(t^{SE}(k)+w(n-k,\lfloor \alpha k \rfloor))$$

$$\leq \sum_{k=2}^{b} \binom{n}{k} 2^{-n}((\frac{8}{3}+\alpha)k + \frac{8}{3}(n-k)+\alpha k) \qquad \text{by lemmas 5,6}$$

$$= \frac{8}{3} n \sum_{k=2}^{b} \binom{n}{k} 2^{-n} + 2\alpha \sum_{k=2}^{b} n \binom{n-1}{k-1} 2^{-n}$$

$$\leq \frac{8}{3} n A_{\epsilon,n} + \alpha n A_{\epsilon,n-1} \qquad \text{by lemma 10}$$

$$\leq (\frac{8}{3}+\alpha) A_{\epsilon,n-1} \cdot n.$$

□

We are now ready to bound $t^{SE}(n)$.

Lemma 12. $t^{SE}(n) \leq 2.35n + 1.4\log n$ for $n>0$ and $\alpha = .785$.

Proof. Direct computation verifies the lemma for $n<200$ (see Figure 4). Choose $\epsilon = 0.2$. Then, for $n \geq 200$, $tail_{\epsilon,n} \leq 0.026$. Break the summation of equation (10) into tail and mid sections, and apply lemmas 5 and 11 to obtain

$$t^{SE}(n) \leq 1 + \frac{8}{3} n(n+1)2^{-n} + n \cdot tail_{\epsilon,n} + \sum_{k=b}^{n-b} \binom{n}{k} 2^{-n} (t^{SE}(k) + w(n-k,\lfloor \alpha k \rfloor)). \qquad (14)$$

By lemma 1 we have

$$w(n-k,\lfloor \alpha k \rfloor) \leq \frac{\mu}{\lambda} e^{\mu-\lambda} K_{\mu}(n-k)$$

for $\lambda = (n-k)/\lfloor \alpha k \rfloor$, and $\mu = \lambda(n-k)/(n-k-\lambda)$. Maximizing $\frac{\mu}{\lambda} e^{\mu-\lambda}$ and K_{μ} independently subject to the restriction that $.4n \leq k \leq .6n$ gives (see Figure 1)

$$w(n-k,\lfloor \alpha k \rfloor) \leq 2.27 \cdot (n-k). \tag{15}$$

Substitute inequality (15) and lemma 12 inductively into inequality (14), yielding

$$t^{SE}(n) \leq 1 + 10^{-55} + 0.26n + \sum_{k=b}^{n-b} \binom{n}{k} 2^{-n}(2.35k + 1.4\log k + 2.27(n-k)).$$

$$\leq 1 + 10^{-55} + 1.4\log(.6n) + 2.296n + (.08)(.6n)$$

$$< 2.35n + 1.4\log n.$$

□

Corollary 13. For $n \geq 200$, $T^{SE}(n) \geq 0.416$.

9. Conclusion

Both protocols SE and MWT(ϵ) achieve a limit throughput of about .4686. But MWT(ϵ), using a special estimation procedure, must pay an overhead cost for obtaining estimates. For ϵ and n small, that overhead is prohibitively expensive. With SE, we have shown how to avoid the overhead entirely, while retaining highly accurate estimates.

As is apparent from figure 4, SE is slightly worse than IBT for small multiplicity sessions. The reader can probably see how to modify SE, without changing α from .785, so that its throughput is identical to that of IBT for sessions of multiplicity 2. However, it appears to be inherent to estimation protocols that they are inferior to similar protocols which do not make estimates, for some small value of n. The parameter α can be adjusted so that $\lfloor \alpha n \rfloor \leq 1$, for small n, and the estimate has no real effect. But there must be some smallest n_0 such that $\lfloor \alpha n_0 \rfloor > 1$. The estimate will have an effect on some sessions of multiplicity n_0, but only when the initial partition is biased completely to one side, and the estimate is very poor.

It therefore does not appear that IBT can be improved on for all n by any divide and conquer protocol.

Furthermore, the identical throughputs of SE and MWT(ϵ) in the limit make it very tempting to conjecture that no divide and conquer protocol can do better in the limit.

Which efficiency measure (min throughput or limit throughput) is more indicative of a protocol's true efficiency? That depends partly on the importance of fairness. Suppose station i is holding k unsent messages. We might cause station i to behave as if it were k different stations, each with one message to send. Then, as the total number of pending message grows, so does the multiplicity of each session; the worst local throughput is avoided, and the limit throughput gives a good indication of channel capacity.

But the above trick violates fairness, and could permit one station to monopolize the channel during a very long session. If 2-fairness is insisted upon, then it is realistic to presume that the channel protocol may be driven at its worst multiplicity for

moderately long periods. Then the min throughput is a more realistic indicator of channel capacity.

REFERENCES

1. Capetanakis, J., Tree Algorithms for Packet Broadcast Channels, IEEE Transactions on Information Theory IT-25, 5, Sept. 1979, 505-515.

2. Greenberg, A.G., Efficient Algorithms for Multiple Access Channels, Ph.D. Thesis, Univ. of Washington, Seattle, Washington (1983).

3. Greenberg, A.G. and Ladner, R.E., Estimating the Multiplicities of Conflicts in Multiple Access Channels, 24th IEEE Symp. on Foundations of Computer Science, Nov. 1983, 383-392.

4. Greenberg, A.G., Flajolet, P. and Ladner, R.E., Estimating the Multiplicities of Conflicts in Multiple Access Channels, T.R. 333, INRIA, Rocquencourt, France, (1984).

5. Hayes, J.F., An Adaptive Technique for Local Distribution, IEEE Transactions on Communications COM-26, Aug. 1978, 1178-1186.

6. Hofri, M., Stack Algorithms for Collision-Detecting Channels and Their Analysis: A Limited Survey, Tech. Rep. 266, Israel Institute of Technology, Dept. of Computer Science (1983).

7. Massey, J.L., Collision-Resolution Algorithms and Random-Access Communications, Tech. Rep. UCLA-ENG-8016, School of Engineering and Applied Science, Univ. of California, Los Angeles (1980).

8. Tsybakov, B.S., and Mikhailov, V.A., Free Synchronous Packet Access in a Broadcast Channel with Feedback, Problems of Information Transmission 14, 4, April, 1978, 259-280.

9. Willard, D.E., Log-logarithmic Protocols for Resolving Ethernet and Semaphore Conflicts (Preliminary Report), 16th ACM Symposium on Theory of Computing, May, 1984, 512-521.