Burst erasure correction to improve the efficiency of broadband CSMA/CD.

John J. Metzner
Pennsylvania State University
Department of Computer Science and Engineering
University Park, Pennsylvania 16802
Email: metzner@cse.psu.edu

Abstract. Assume a broadcast network with a central station. All senders send to the central station, and the central station rebroadcasts all receptions on a different band. In standard CSMA/CD, all senders stop when a collision is detected. In the suggested modified algorithm, exactly one continues to send. A colliding sender can know if it was the first to arrive at the central station. If so, it continues to send, else it stops. The one that continues to send incurs an observable-length erasure burst, and appends a redundant part to its frame to allow filling in the burst erasure. Also, a collision resolution-like algorithm is introduced which improves fairness and performance.
Key words: CSMA/CD, multiaccess, collision resolution

I. Introduction.

Carrier-sense multiple access with collision detection (CSMA/CD) has become the standard for Ethernet local area networks [1]. In some multi-access local area networks a station is able to see the return of its own sending as well as a collision. Examples are: 1) Broadband Ethernet, where carrier-frequency modulated signals are sent to a headend, which regenerates the received signals with a different frequency and rebroadcasts the signals to all the stations; 2) Wireless transmission to a central station, which rebroadcasts as with the headend example, except wirelessly; 3) Unidirectional folded buses, where each station has two separate connections to the bus – one on the first half of the bus for sending, and one on the second half of the bus for receiving.

For the first two cases, the usual procedure is for all senders to stop when a collision is detected. In case 3), there is a scheme called Expressnet [2]. This scheme uses a method of continuing one transmission and aborting the other in case of a collision. It has substantial throughput and stability advantages over Ethernet. However, it requires an

ordered topology, which is not the case in examples 1) and 2) above. In examples 1) and 2), the ability of a sender to observe whether its sending is the first to return also can allow a colliding sender to abort while another collider continues successfully. In this paper an algorithm and some variations will be described that improve efficiency by allowing one collider to continue while the others abort. Also, collision resolution-like algorithms are introduced.

There also is work on broadcast star networks [3,4], where the central station can select the first message received and reject others. However, these networks assume the central station can receive time overlapping signals without interference, such as by having a separate channel for each sender. This capability is not assumed here. Rather, we assume that the central station receives all signals on the same up-channel, and simply broadcasts everything, including the collisions, on a separate down-channel, with no delay or a small fixed delay.

II. The basic algorithm and variations.

A. The basic algorithm.

1. A sender uses Carrier Sense. If a message arrives when the channel is sensed idle, it is sent immediately. If a message arrives when the channel is sensed busy, the potential sender waits until the channel goes idle, and then waits a random time uniformly between 0 and a, where a is small, preferably less than or equal to the maximum round trip propagation time.

2. A sender stops if a signal is seen on the return channel before the appointed return of its own signal. If its own signal returns before any other signal, the sender continues even if it observes a collision.

3. If there is a collision, the length of the collision is directly observable. It will appear as an error burst. The sender appends a redundant tail to its packet, sufficient for burst erasure correction. Possibly a few extra bits could be added for allowance of uncertainty as to the exact burst length. Burst erasure correction is a simple operation for a cyclic code, and the minimum number of extra bits needed is exactly the burst length.

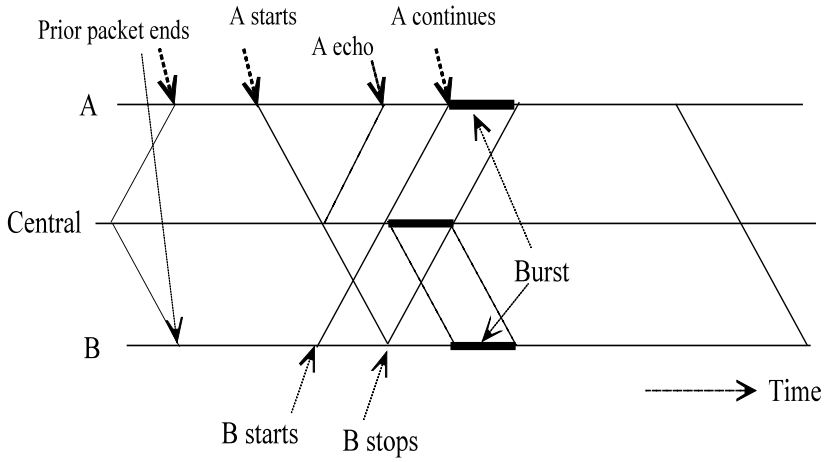

Figure 1. Timng diagram illustrating burst interference.

Figure 1 shows an example where A starts just before B. Then B stops and A continues. A recognizes the burst, which is the same size at all receivers, and appends a redundant tail slightly greater than the burst, for burst erasure correction. Burst erasure correction is a simple operation for a cyclic code, and the minimum number of extra bits needed is exactly the burst length.

B. Variation 1.

For small β, say $\beta < 0.1$, a simpler alternate rule can be used: Always include a redundant preamble of length = β. Then no burst erasure correction need be added, since only the first β seconds can be interfered with, and this part is not needed. This will create somewhat longer delays, but the asymptotic maximum λ will not change much, because at high backlog an appended length of very close to β will usually be needed.

C. Variation 2.

The basic algorithm includes an extra β second pause, after each success, in transmission from the central station. This is illustrated in Figure 2. If station A knew when the prior message would end, it could start β seconds earlier. In variation 2, a packet includes a special marker signal,

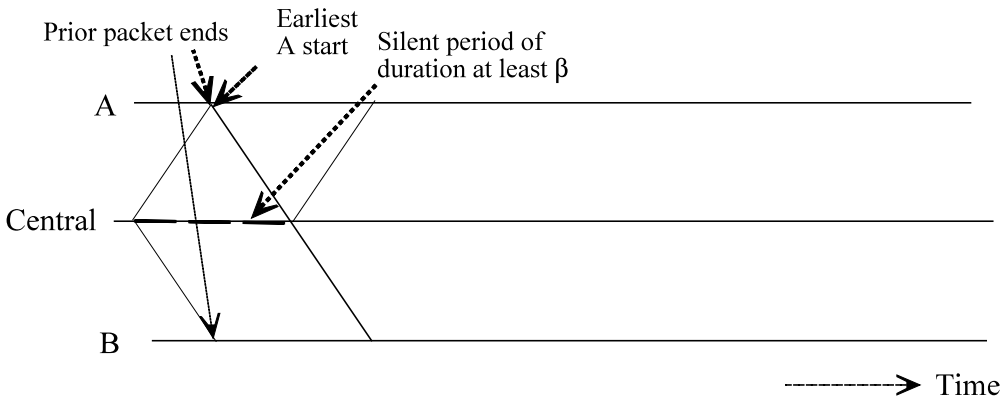

Figure 2. The silent period with the basic algorithm.

such as an unusually strong, clearly recognized pulse, exactly β seconds before the end of the packet. (It is assumed that the packet length is always > β.) Alternatively, all packets could be the same size. This would lead to a timing diagram as in Figure 3.

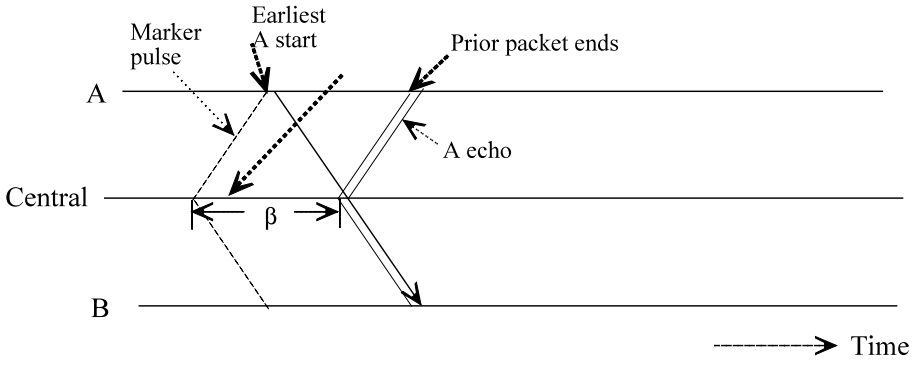

Figure 3. Starting earlier with aid of marker pulse - variation 2.

Assumptions for the analysis.

1. Assume all stations are equidistant from the central station. Let τ be the one-way propagation time to or from the central station. The assumption of equidistance is a common one [5]. If the distance is taken as the maximum end-to-end distance, it is a worst-case assumption; greater efficiencies would occur if some stations were closer to each other. The algorithm still works effectively if stations are different distances from the central station or headend. The return signals still establish unambiguously which sender's signal returns first. (which actually started first doesn't matter; only which returns first.)

2. Assume the sender always can tell if its sending returns first. Possibly, a strong pulse could define the starting point of any message. Each station can have a very good estimate of when its signal will return (i.e., it knows the round-trip time to the central station.) For a very close call it is possible that two will continue to send, both thinking they are first, or both will stop, each thinking the other is first. Either event will quickly become evident, but will result in some additional lost time. However, for simplicity, the analysis will ignore this possibility. Later we will discuss the effect of a violation.

3. The average packet duration is normalized to be $T = 1$, with a minimum duration of β. $\beta = 2\tau/T$ is the ratio, round-trip propagation time/average packet duration.

4. New packets needing transmission arrive according to a Poisson distribution at a normalized rate λ. None are turned away, assuming λ is less than the critical value.

5. If a packet arrives while the channel is sensed idle, it is sent immediately. If a packet arrives while the channel is sensed busy, it waits until the channel is idle, and then waits a random time y between 0 and a, and then sends. Choose $a \leq \beta$. For small β, $a = \beta$ might be chosen, but for large β, a should be smaller, say 0.1.

IV. Analysis.

We will look at the system state at the end of a transmission interval. The state is the number, n, of packets that have arrived and are waiting transmission.

Consider the state n = 0. There will be an average duration $1/\lambda$ before the first arrival. If no other arrivals occur an interval β after the first, the signals will end after an average time of $1/\lambda+1+\beta$. If there is another arrival in the next β seconds, there will be an interference burst that will add to the duration. If a burst length, consider the effect of a second arrival at normalized time x. Figure 4 shows that this causes a burst of length $\beta+y-x$, where $y < x$.

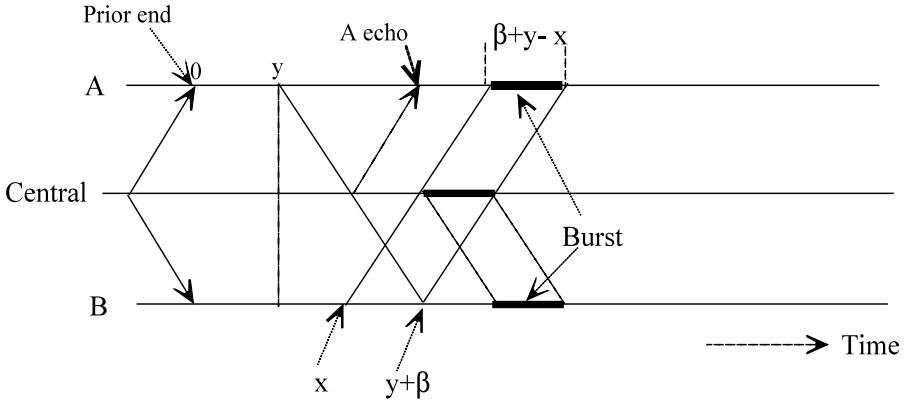

Figure 4. A burst after state n = 0.

Since arrivals are Poisson, with z = y-x,

$$\bar{b} = \int_0^{\beta} (\beta - z)\lambda \cdot e^{-\lambda z} dz = \frac{\beta\lambda - 1 + e^{-\lambda\beta}}{\lambda} < \frac{\beta^2 \lambda}{2} \qquad (1)$$

The above inequality is a close approximation, so we can assume the average cycle duration from n = 0 is

$C_0 = 1/\lambda + 1 + \beta + \beta^2\lambda/2$ (2)

during which 1 unit of successful delivery occurs.

Y A echo

Now consider state n = 1. The waiting sender picks a starting time y between 0 and a. There may be a first new arrival at some time z, If z > y+β there is no burst interference. If z < y then the new arrival will be send its packet subject to a burst of length β-y+z. If y<z<y+β, the waiting sender will send its packet subject to a burst of length β-z+y. Integrating over the joint arrival densities, one uniform and the other Poisson, the average cycle time starting from state n=1 is

$$C_1 = 1 + 2\beta + \frac{2}{\lambda} - \frac{a}{2} - \frac{(1-e^{-\lambda a})(3-e^{-\lambda\beta})}{a\lambda^2} \quad (3)$$

For small λβ and λa, $C_1 \approx 1+\beta+a/2$.

Now consider state n > 1. If a ≤ β, there will always be a burst interference. The average time to when the first of these n will try to send is readily shown to be a/(n+1). Once a packet has begun to send first, it can be subject to a burst of at most β. The average time for the second arrival after the first is also a/(n+1). This reduces the burst length from the maximum length β by exactly the same average as the time to start, so the total average waste is β. There is another effect for the case of new arrivals. See Figure 5. A new arrival at time x, prior to the start y of the first transmission would

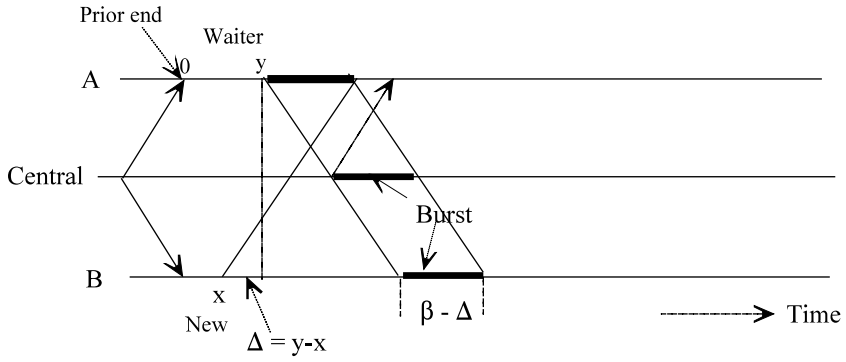

Figure 5. New packet sent prior to waiting packet.

create an arrival time + burst = x + β - Δ = β-y+2x. Let s be the average of x, given x<y, so the average arrival time + burst = β - (a/2 -2s). But s < a/4, half the average of y, because the exponential arrival density favors arrival in [0,y/2] above [y/2,y]

Another possibility is that the new arrival falls between the first and the second waiter's starting time, as in Figure 6. Here, arrival time + burst = y + β - Δ, where the average of Δ is s, the same as the average of x in the first case, since the conditional spacing between the first and second waiter is the same as from 0 to y. Thus the average arrival time + burst = β + (a/2 - s). This is greater than β by slightly more than it was less than β for the figure 5 case, but, unconditionally, it is more likely a first arrival will occur in the first of two equal size intervals due to the exponential arrival density. Also, the more likely case, especially if n is large, is that no new arrival will come before the

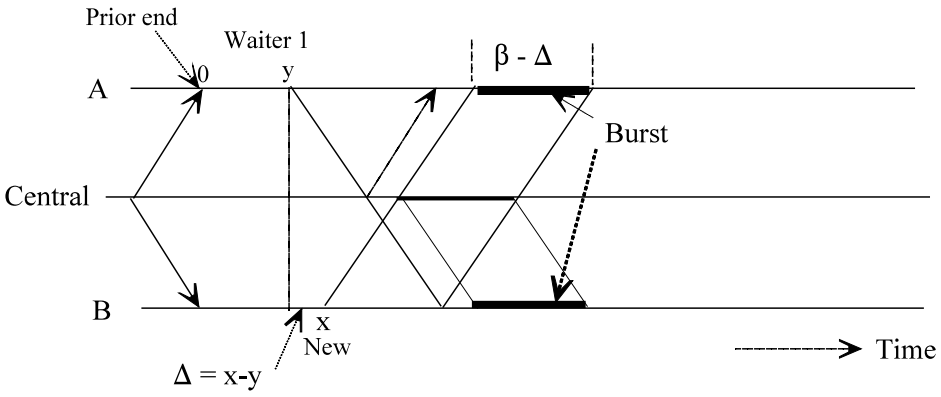

Figure 6. New packet starts between first and second waiting packet.

second waiting sender, in which case the average arrival time + burst is β. Thus, for simplicity, we will use the same 1+2β assumption for the transmission time for all n>1.

A. Basic algorithm.

The state at each end of transmission of a packet can be described by a Markov chain. Let $\ell(i)$ be the expected duration starting from state i. From the analysis above,

$\ell(i) = 1+2\beta$ for $i > 1$ (4)

$\ell(1) = 1+\beta + a/2$ (5)

$\ell(0) = 1 + \beta + \beta^2\lambda/2$ (6)

Note that from state 0, the time averaging $1/\lambda$ to first arrival doesn't count toward new arrival, since there wasn't an arrival.

The probability of i arrivals per transition from state k is

$$g_k(i) = \frac{[\lambda \cdot \ell(k)]^i}{i!} e - \lambda \cdot \ell(k) \qquad (7)$$

Figure 7 shows a cut between states n and n–1. There is only one transition from state n, n>0, to a lower state, and that is to state n–1, but there are many transitions from a state < n to a state ≥ n.

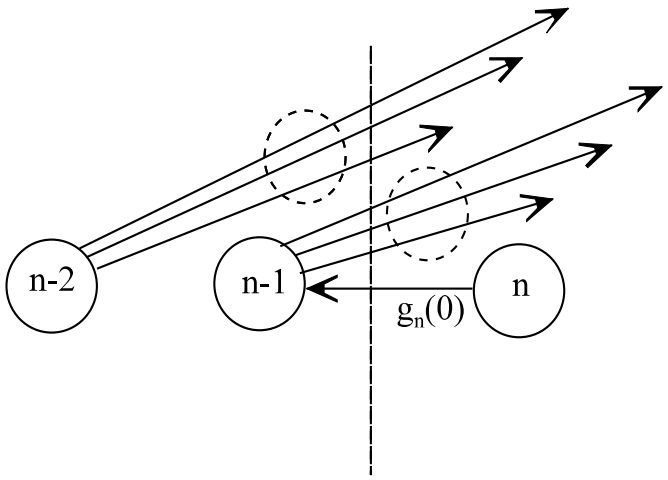

Figure 7. Transition balance line for the Markov state diagram.

A balance equation across the cut defines an iterative relation on the state probabilities:

$P_n g_n(0) = P_{n-1}\sigma_{n-1}(1) + P_{n-2}\sigma_{n-2}(2) + \ldots\ldots P_1\sigma_1(n-1) + P_0\sigma_0(n-1)$, (8)

$n > 1$, where

$$\sigma_k(j) = 1 - \sum_{i=0}^{j} g_k(i) \qquad (9)$$

and

$$P_1 = \frac{[1 - g_0(0)]}{g_1(0)} \qquad (10)$$

Note that in either state 1 or state 0 it takes n or more arrivals to cross the barrier.

B. Variation 1.

In this case g is independent of the prior state, since $\ell(i) = 1+2\beta$ for all i. so equation (7) simplifies to

$$g(i) = \frac{[\lambda(1+2\beta)]^i}{i!} e^{-\lambda(1+2\beta)} \qquad (11)$$

and the subscripts in equation (8) and (9) are not relevant. The average length of a cycle is $1+2\beta + P_0/\lambda$, and this must be $1/\lambda$, since there is one packet resolved per cycle. Thus

$P_0 = 1 - \lambda(1+2\beta)$ (12).

C. Variation 2.

Basically, all the $\ell(i)$ values in the basic algorithm or variation 1 are reduced by the amount β, since events occur in the same way, except β seconds earlier.

V. Some performance comparisons.

Figures 8 and 9 show average backlog (average state n at start of a new transmission) versus throughput of the scheme and variations for β = 0.2 and 0.3. For both cases a = 0.1. The limiting throughput of the basic algorithm and variation 1 is about 1/(1+2β), and for variation 2 it is about 1/(1+β).

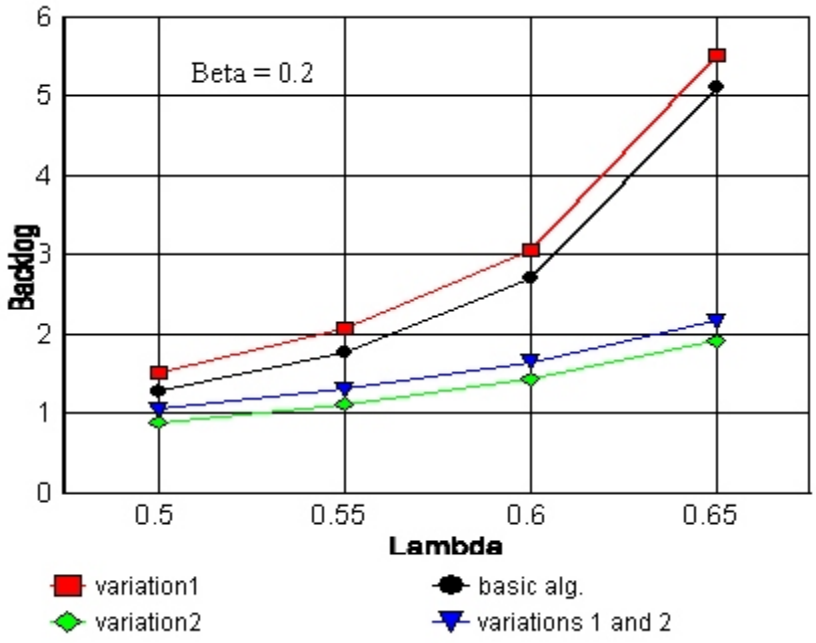

Figure 8. Average backlog vs. throughput for β = 0.2

Variation 1 shows only a small deterioration for β = .2 over the basic algorithm, and somewhat more for β = 0.3, but saves the need for burst erasure correction. Variation 2 shows a large improvement over the basic algorithm. Again, including the variation 1 of always including a β preamble has a moderate extra cost, but is simpler. However, these results assume all distances are equal. If distances varied over a wide range, variation1 would be penalized, since it would have to include a preamble for the largest β, even though much smaller bursts would often be encountered with shorter range senders.

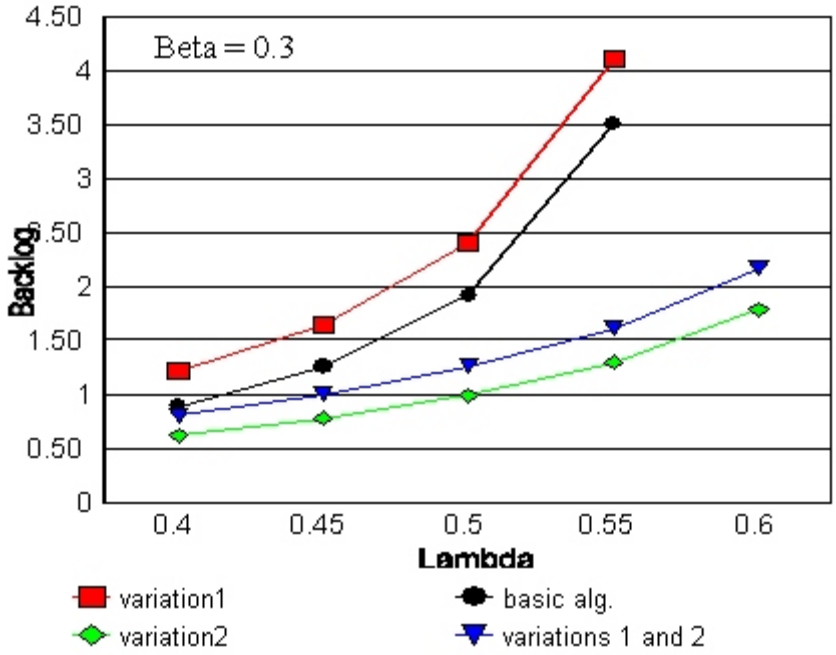

Figure 9. Average backlog vs. throughput for β = 0.3

Consider the alternative of unslotted CSMA/CD in Ethernet. The limiting efficiency is about 1/(1+6.2β) according to [6], compared to 1/(1+2β) or 1/(1+β) for the new algorithms. Furthermore, the 1/(1+6.2β) limit often is achieved under rather unrealistic offered loads, and rates near the limit are highly unstable. . The exact limit expression in [6] is:

$$\frac{e^{-\beta G}}{\beta + 1/G + 2\beta(1-e^{-\beta G}) + e^{-\beta G}}$$

For example, at β=0.01 a maximum throughput of .94 occurs at G = 43, which would correspond to about 43 attempts per success! At G = 1 and β=0.01 , the throughput is only .495, with an average of 2 attempts per success.

The proposed methods do not exhibit an instability problem, because for any $n>0$ there is at worst one success per $1+2\beta$, so if on average less than one arrival occurs per time $1+2\beta$, n will always drift down, assuming one can always tell which is first to arrive.

VI. Effect of misinterpreting which is first.

At very large n the effect of near-simultaneous arrivals may be significant. This may lead to too many cases where two think they are first or none think they are first. Such cases are detectable by the burst lasting too long or the message being too short, but some time is wasted. Large n is very rare, however, unless λ is very close to the limiting value. If arrival times are extremely close it might not be possible to tell which of two came first. If uncertain, a station might decide with probability ½ whether to continue. In this case, there are three possibilities:

1. Both keep going - event is discovered after interference continues for $>\beta$. After say 1.5β of collision both stop. A wasted time of 2.5β seconds occurs before the idle state returns (at receivers).
2. Both stop. A new idle period now starts at time β seconds (at receivers) after the transmission start.
3. One stops. Success with a β second interference burst. There is no extra waste.

   The average extra waste in this event is $2.5\beta/4 + \beta/4 = .875\beta$. Inability to resolve the starting times could be rare, but the probability increases when the backlog of waiting senders increases. If it happened in 10% of the cases, it would add less than 0.1β to the average duration of a success. The case of large n can be recognized. A possible strategy is to increase a in this case. Another possibility is collision resolution.

VII. Collision resolution-like algorithms.

Collision resolution algorithms often are considered in multiaccess networks. Here, a modified form of collision resolution is suggested. It is similar to ordinary collision resolution in that all not involved in a collision will defer until the collision is resolved. It is different because, in the collision, one sender becomes successful, while the others stop after causing a burst. If $a \leq \beta$ and $n>1$, there will always be a burst, and if there is a burst, n will

still be at least 1. Assume that channel observers can tell if there is an interference burst. Thus we can have the following rule: If there is a burst, only those which contributed to the burst will attempt in the next cycle. Stations experiencing new arrivals are assumed to have been sensing the channel prior to arrival, so they also can refrain from sending until a non-burst transmission succeeds. For example, if n = 3 initially, first one will succeed, with a burst experience. Then the second will succeed, also with a burst experience. Finally, the third will send, with no burst experience. This lack of burst interference triggers to the other arrivals that they are free to send. Thus we have the following rule: any new arrival can attempt to send only if the prior observed transmission has been a burst-free transmission; otherwise it will join the queue, refraining until a there is a non-burst occurrence.

An advantage of this collision-like resolution is that it is fairer in its effect on delay. If a packet has arrived in state n following a collision resolution interval, it will not have to suffer more than an $n(1+2\beta)$ delay. Without collision resolution, an arriver in state n might have to wait an indefinite time, as it may continue by chance to be beaten out by later arrivals that happen to always start earlier. This is of particular importance if distances are unequal, as we shall see, since a near station might always send earlier than a far station.

Collision-like resolution state analysis is on a different time scale from the prior analysis. Previously, the state is observed after each successful transmission. Now the state is observed only after each collision resolution cycle. This problem is amenable to a drift analysis from a given state. Starting in state n_1, the time to resolve is $T(n_1) = (n_1-1)(1+2\beta) + (1+\beta)$; in variation 2 it is $T_2(n_1) = (n_1-1)(1+\beta) + 1$. The new state will be the number of arrivals during the duration of the prior contention period. We will see that if n_1 is large, n_2 is likely to be much smaller.

$$P(n_2 / n_1) = \frac{[\lambda T(n_1)]^{n_2}}{n_2!} \cdot e^{-\lambda T(n_1)} \qquad (13)$$

Also, it follows that

$$\bar{n}_2 = \lambda \cdot T(n_1) = \lambda \cdot [n_1 + (2n_1 - 1)\beta] < n_1 \lambda (1 + 2\beta) \qquad (14)$$

and

$$P[n_2 > n_1] < \frac{[\lambda T(n_1)]^{n_1+1}}{(n_1+1)!} \cdot e^{-\lambda T(n_1)} \cdot \frac{1}{1-\frac{\lambda T(n_1)}{n_1+2}} \quad (15)$$

For example, if $\lambda = .5$ and $\beta=.1$ for the basic algorithm or variation 1, the average number at the next cycle start is less than 60% of the prior cycle number. Also, if $n_1 > 21$ there is less than a 1% chance that the next cycle will see a greater or equal backlog. Thus the backlog has a strong tendency to drift downward as long as λ is not close to $1/(1+2\beta)$, or $1/(1+\beta)$ with variation 2.

Collision resolution also can be used in the case where two or more senders fail to stop due to confusion on which is first. This is recognized after about 2.5β of wasted collision, when they will stop. Others, who have terminated their attempts earlier, will see from the prolonged burst that two or more have failed to stop. When they do stop, only those who failed to stop will select a starting time from 0 to a from the aborted ending time. This will greatly diminish the risk of the confusion recurring. Since most likely only two or a small number $2 \le k < n$ will have picked about the same earliest time. The first k–1 of k will succeed with a burst event, then the last will succeed without a burst. The other n-k and any additional arrivals can then contend after the burst-free success.

Also, if a large input burst of n_1 arrivals occurs, these can be resolved in time $T(n_1)$, and the next number arriving in $T(n_1)$ will likely be much less than n_1, given another burst does not occur immediately after. Thus there is a quick recovery after an input burst.

VIII. Effect of Unequal distances to central station.

A. With the basic algorithm or variation 1.

Unequal distances do not significantly complicate operation; they only complicate analysis. The rule is that the first station whose signal comes back from the central station is the station that continues to send. This is not always the station that started first in absolute time. Stations close to the center may win out even if they send later, but the winner is unambiguous. This is illustrated in Figure 10. Station B is first to arrive at the central station, even though station A sent first in absolute time. However, there is no ambiguity, as A and B can observe that Station B is first in the broadcast.

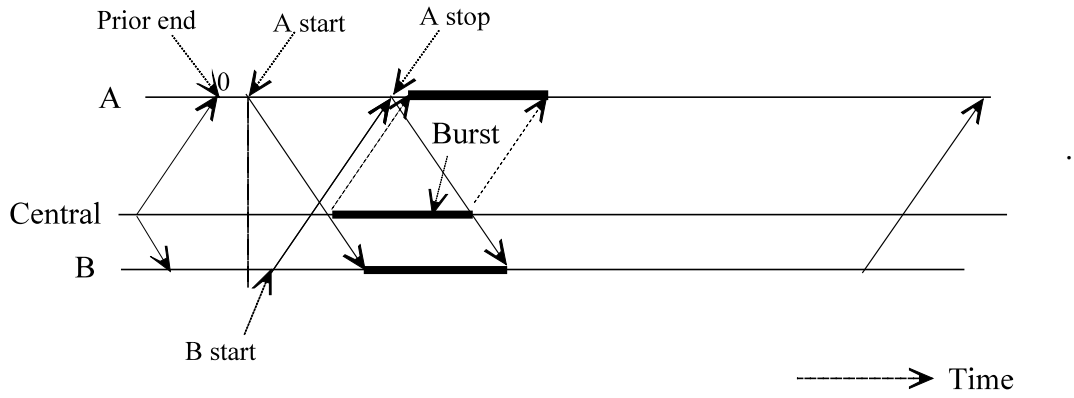

Figure 10. Unequal distances to central station.

If the distances to the central station are very unequal, there can be unfairness. If a close and a far station are both in a backlog, the closer station will almost surely reach the central station first and be the one to complete its packet transmission. The near station, if it had a long string of packets to send, would almost always continue to win out over the far station. One could make a near station wait longer to improve the fairness, but this would negatively affect efficiency. We have seen that collision-like resolution can solve the

unfairness problem without impairing efficiency. The near station would still send its packet first, but the resulting burst would allow only packets that had been previously been backlogged to contend, until all n packets in the backlog would be sent successfully. Thus all n packets that had been waiting will get serviced before a new packet can be attempted by the first close station. Also, unequal distances actually reduce the chance of the very close arrival times that might cause incorrect behavior.

Burst effect with unequal distances. Consider the collision of a near and a far station. Figure 11 shows the case where the near station arrives first.

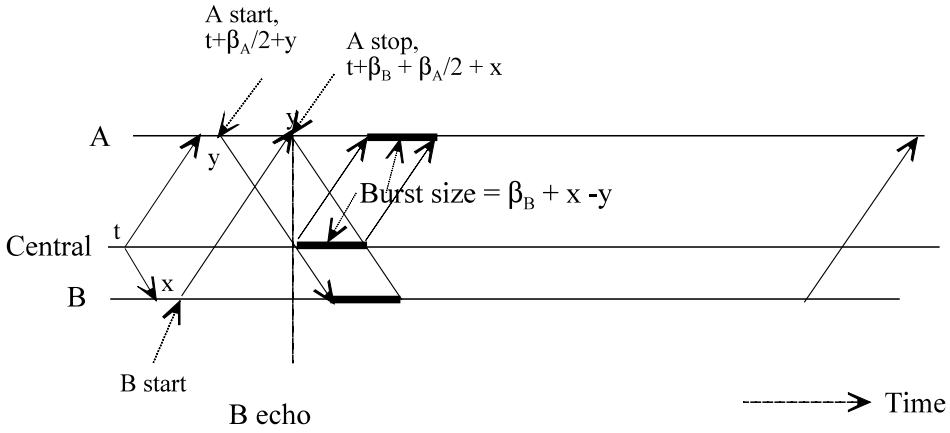

Figure 11. Burst when close station signal arrives first.

A similar argument for when the far station arrives first reveals that, in both cases of two attempts, the burst length is the β of the winning station minus the absolute difference between the winning and losing starting times. If both stations pick a time between 0 and a, and $\beta_A - \beta_B > a$, the farther station will never win. Despite this, with collision-like resolution, both winner and loser will send one packet each in the resolution interval. Also, the close station winner will experience a burst size less than its own small β, and if only 2 attempt, the loser will then succeed with no burst. As to variation 1, a single interferer can cause a burst no longer than the β of the winning sender, so a close station could afford to include a preamble of its own β. However, this is not sufficient if there are more than 2 interferers.

Suppose there are > 2 transmission attempts. If there are several interferers, their total burst effect can exceed the winning sender's β. Figure 12 shows how a near and a far interferer can cause two non-overlapping bursts.

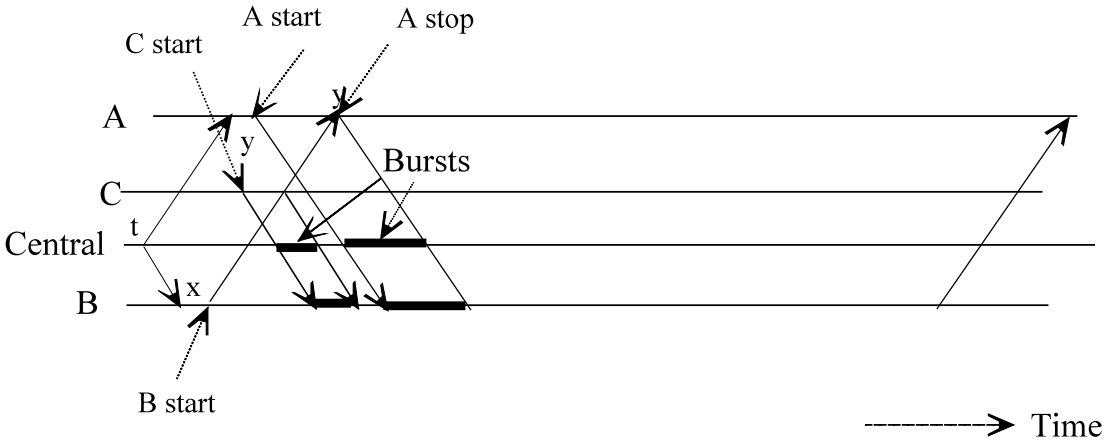

Figure 12. Effect of two interferers at different distances.

In any case, no matter how many interferers, the total duration of the burst cannot exceed the β for the farthest station among the interferers, since no interfering signal can arrive at the central station before the winner, and no interfering signal can arrive at the central station after the winner's first arrival by more than that interferer's β. But to always provide β_{MAX} redundancy for variation 1 might be too costly.

B. Unequal distances with variation 2.

In variation 2, let the marker be set at β_{MAX} , the maximum round trip propagation time before the end of transmission. Then each station, knowing its own β, can compute when the current transmission will end at the central station, and when it should send for its transmission to arrive just after that. Thus the near and far stations will select a time which can be in the same 0 to a range at the central point. Figure 13 illustrates the burst size in a conflict between a near and a far sender.

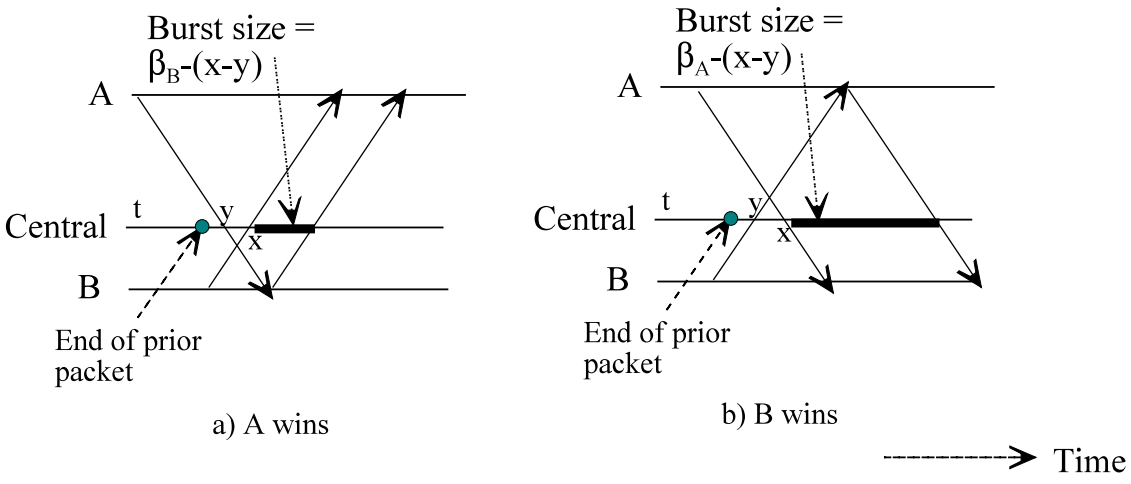

Figure 13. Variation 2 burst events.

If both pick for x or y a random number between 0 and a, each is equally likely to win, but there is a larger burst if the near station wins. This is the opposite of the effect with the basic algorithm, where there is a smaller burst if the near station wins. Again, if there are multiple interferers, the total burst length cannot exceed the largest β among the interferers.

Variation 2 does not gain the advantage of a wide spread of arrival times that unequal distances would have using the basic algorithm or variation 1. The spreading could be achieved by other means. For example, say a large number of close stations are at β, and a smaller number at β_{MAX}. The near station set could select their starting times in the range β + a to β + 2a, while the far set would select from 0 to a. Then the near stations would sense the far stations being first before their appointed time, and would defer. If no far station sent, they would send with interference only from the other near stations. To prevent far stations from hogging the channel, the near stations could switch to the 0 to a starting times after some amount of deferral.

.

IX. Discussion and Conclusions.

The policy of one station continuing to send on collision is effective for CSMA networks that employ a central broadcasting station. Much larger throughput can be achieved than with standard CSMA/CD.

Stability and low delay are also features of interest. The average time to successfully deliver one packet of unit length is less than a fixed $1+2\beta$ over a wide range of backlog size. As long as the average arrival rate is less than 1 packet in the $1+2\beta$ time units it takes to deliver one packet, backlog will drift lower. The only exception is that at high backlog the time spacing between attempts may be too close to ensure that the stations know which is first. Such high backlogs are rare unless the throughput is very close to capacity. Collision-like resolution can react to the case where two or more fail to stop because they believe they are first. It does it by making these the collision resolution set while others defer. In other cases, the one that started first is successful with a burst, and the remainder that caused bursts are the collision resolution set. Then, collision-like resolution creates fairness because, If n senders collide, each of the n senders will get to send one packet successfully, using only n successive intervals of at most $1+2\beta$ seconds each. This is even better than ordinary collision resolution algorithms, where probabilistically there will be intervals wasted because no one sends. Delays are small up to a large fraction of limiting rate.

It is possible that the offered load exceeds capacity. This would threaten to increase the backlog state numbers n without limit. In the collision-like resolution mode, the prior state n is readily discernable by counting the n successes in the resolution cycle. Input throttling strategies can be employed if n exceeds some threshold.

Burst erasure correction is easily accomplished by appending redundant data based on the structure of a cyclic code. At some extra cost, extra redundancy of length β could always be inserted as a preamble. But with varying distances it would be inefficient to always provide the maximum extra redundancy.

The analysis mostly assumes all waiting stations select a starting time uniformly from 0 to a. It is possible to give different values of a, or ranges, based either on priority or on the β for a particular station. Different ranges can reduce the chance of misinterpreting which is first.

X. References.

[1]. IEEE Standard 802.3, CSMA/CD Access Method and Physical Layer Specification, *IEEE Project 802, Local Area Network Standards*, IEEE, New York, July 1983.

[2]. F. A. Tobagi and M. Fine, "Performance of Unidirectional Broadcast Local Area Networks: Expressnet and Fasnet," *IEEE J. Select. Areas Commun.*, vol. SAC-1, pp. 913-926, Nov. 1983.

[3] S.H. Nam and C.K. Un, "Performance Analysis of Broadcast Star Network with Priorities," *IEEE Trans. on Commun.*, vol. 42, pp. 1785-1794, Feb./Mar./Apr. 1994.

[4] A. Albanese, "Star network with Collision-Avoidance Circuits," *Bell System Tech. Journal*, v. 62, pp. 631-638, Mar. 1983.

[5]. R. Rom and M. Sidi, *Multiple Access Protocols, Performance and Analysis*, Springer-Verlag, 1990.

[6]. D. Bertsekas and R. Gallager, *Data Networks*, *Second Edition*, Prentice Hall, 1987.