AN ACKNOWLEDGING CONTENTION ALGORITHM SUITABLE FOR LOCAL RADIO NETWORKS

by

Michael A. Malcolm
Lawrence D. Rogers
John E. Spracklen

Research Report CS-83-23

Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1 Canada

June 1982 Reference: ICR Report 1

AN ACKNOWLEDGING CONTENTION ALGORITHM SUITABLE FOR LOCAL RADIO NETWORKS

Michael A. Malcolm
Lawrence D. Rogers
John E. Spracklen

Report 1

June 1982

University of Waterloo
Waterloo, Ontario, Canada

An Acknowledging Contention Algorithm Suitable For Local Radio Networks

Michael A. Malcolm

Lawrence D. Rogers

John E. Spracklen

Abstract: We present a contention transmission algorithm for local networks which is related to the p-persistent family of algorithms previously analyzed by Kleinrock and Tobagi. Our algorithm incorporates an automatic acknowledgement signal which is sent by the receiving station immediately after each packet is correctly received. The algorithm is suitable for radio environments in which transmitting stations can "capture" nearby receivers. Assuming that stations are not placed too close to each other, we prove that an acknowledgement signal is received by a transmitting station only if the packet has been received correctly by the intended receiver. In a cable network where transmitters do not capture receivers, the acknowledgement signal is guaranteed (in the same sense) regardless of the distances between stations. This result depends on very weak assumptions about the type of data encoding used on the channel. We discuss the more interesting aspects of implementing this scheme, and we outline how higher-level protocols can make use of the acknowledgement signal and its properties.

1. Objectives

The contention algorithm described in this paper was developed at Burroughs Corporation as part of the Shared Terminal Interface local network project. The primary objectives of the network design were simplicity and reliability. We wanted a network that could be implemented with a small amount of hardware and software. We wanted reliability in the sense that if any station in the network fails, the rest of the network continues to function normally. We also wanted a highly-reliable network communication channel that could be implemented with a passive medium such as coaxial cable or radio.

To achieve these objectives, we designed a carrier-sense multiple access (CSMA) contention algorithm without collision detection. For reasons of simplicity (and for other reasons discussed later), fixed length packets are used. The objective of implementing a reliable datagram protocol level motivated the use of automatic hardware acknowledgement. Automatic hardware acknowledgement provides partial assurance of packet data transmission, thus simplifying the software protocol and reducing network traffic.

There are potential pitfalls in passive medium networks that are due to differing relative signal strengths that allow one station to capture another. We say that a receiving station is captured by a transmitting station when the transmitter's signal (at the receiving station) is not discernably affected by one or more other simultaneous transmissions. Some of the potential problems due to a transmitter capturing a nearby receiver are discussed in Section 3 along with some possible solutions and a proof that our algorithm is not susceptible to such problems.

The contention algorithm is implemented as a 24-pin DIP NMOS integrated circuit, with a transmission speed of one million bits per second, and a maximum network diameter of 5000 feet. The implementation has been tested using a coaxial cable medium. At the time of this writing, the implementation has not been tested using a radio communication channel medium.

2. Contention Algorithm

The network consists of a shared communication channel which is accessed using distributed control. In the case of a coaxial cable communication channel medium, the cable is tapped at arbitrary locations to accommodate station connections. In the case of a radio communication channel medium, each station has its own transmitter, receiver, and antenna, and each station uses the same frequency and transmission power.

Data are transmitted in fixed-sized packets, as shown in Figure 1.

SYNC	Destination Address	Data	CRC	Pad

Figure 1: Packet Layout

Each packet starts with a "1" bit SYNC signal followed by a 16-bit destination address. (Each station in the local network has a unique 16-bit address.) The Data field of each packet is 132 bytes in length. The contents of the Data field are ignored by the hardware, except for computing the CRC which is appended to the Data field. The CRC is a 16-bit cyclic redundancy error detection code which is generated by the transmitting station and checked by the receiving station. Pad is a single "0" bit which is used to allow receivers to clearly distinguish the end of a packet. The Destination Address, Data, CRC, and Pad are encoded using Manchester encoding. The receiver, in addition to checking the CRC, checks that the correct number of bits are received (recall that all packets are the same length), and that each bit is a legal Manchester encoding.

Each station transmits packets independently of other stations, possibly interfering or colliding with other transmissions. If there is no collision, and the transmitted packet is received correctly by the destination station, then the receiver responds with an acknowledgement (ACK) signal. If an ACK is received by the transmitting station, then it is assumed that the packet was correctly received by the intended receiver; if an ACK is not received, then it is assumed that the transmission was unsuccessful.

The transmission algorithm is a variant of Kleinrock and Tobagi's p-persistent CSMA algorithm [1], which incorporates Tokoro and Tamaru's idea of automatic acknowledgement [2]. Automatic acknowledgments have been applied previously in the Hyperchannel of Network Systems Corporation, as described by Donnelley and Yeh [3].

The channel cycles sequentially through three states: _idle_, _packet-being-transmitted_, and _acknowledgement-period_. Each station continually monitors the channel and keeps track of its state. Because there is a propagation delay through the channel, the exact times of transitions between these states vary from point to point along the network.

Thus, each station monitors the state of the channel at the station's point in the network. This channel monitoring is accomplished by a hardware state machine called the <u>Channel State Machine</u> (CSM), described in Figure 2. Whenever the channel is idle, the CSM is in SYNC WAIT state. When a SYNC signal is detected on the channel, the CSM goes into PACKET state, which indicates that the channel is in packet-being-transmitted state. The CSM remains in PACKET state for a fixed duration T, the time required to transmit a (fixed-size) packet. The CSM then goes into ACK IDLE state which is the first of two CSM states corresponding to the channel's acknowledgement-period. The ACK IDLE state is of fixed duration $A = 2 + D + 2t$ bit transmission times, where D is the maximum receiver delay between receiving the last bit of a correct packet and transmitting the first bit of the acknowledgement, and t is the longest possible end-to-end propagation delay for a maximum diameter network. (For the present implementation, a bit transmission time is one microsecond, D is 2 microseconds, and t is 8 microseconds.) The CSM then goes into ACK WAIT state which is also of fixed duration A. (An ACK signal is two bits in length, so the ACK WAIT state is long enough for the ACK transmission plus two signal propagations each of duration t.) The CSM then goes from ACK WAIT state to SYNC WAIT state, to complete the cycle. Whenever the station is powered on or reset, it waits in IDLE WAIT state until the channel has been idle for a continuous period of $T + 2A$ before going into SYNC WAIT state.

```
         reset
           |
         __V__
        /     \
       | IDLE  |
       | WAIT  |
        \_____/
           |
           |  idle for duration T
         __V__
        /     \
  ---->| SYNC  |
 /     | WAIT  |_______
|       \_____/        /_____  channel IDLE
|          |
|          |  SYNC signal detected
|        __V____
|       /       \
|      | PACKET  |_______
|      |         |       /____  packet-being-transmitted
|       \_______/
|          |
|          |  timeout T
|        __V__
|       /     \
|      | ACK   |
|      | IDLE  |_________
|       \_____/          \
|          |              |
|          |              |
|          |  timeout A    \__  acknowledgement-period
|        __V__             /
|       /     \           |
|      | ACK   |          |
|      | WAIT  |_________/
|       \_____/
|          |
|          |  timeout A
 \________/
```

Figure 2: Channel State Machine

When a packet is transmitted by a station, the following algorithm is used:

Step 1 The transmitter waits, if necessary, until the channel is idle. Then Step 2 is performed.

Step 2 An integer s is chosen randomly in the interval [0,S], with each of the S+1 possible choices being equally probable. The transmitter then delays for s bit transmission times. If the channel is still idle at the end of the delay, then the packet is transmitted, and Step 3 is performed; otherwise, the transmitter waits until the channel becomes idle and repeats Step 2.

Step 3 If any channel activity is detected during the ACK IDLE state, the transmitter ignores the ACK signal, if any, during the ACK WAIT state.

Step 4 After the ACK WAIT state has completed, the station is informed whether or not an ACK was received. (This could be done through an interrupt request and/or a status register.)

The transmission algorithm is implemented by the state machine illustrated in Figure 3.

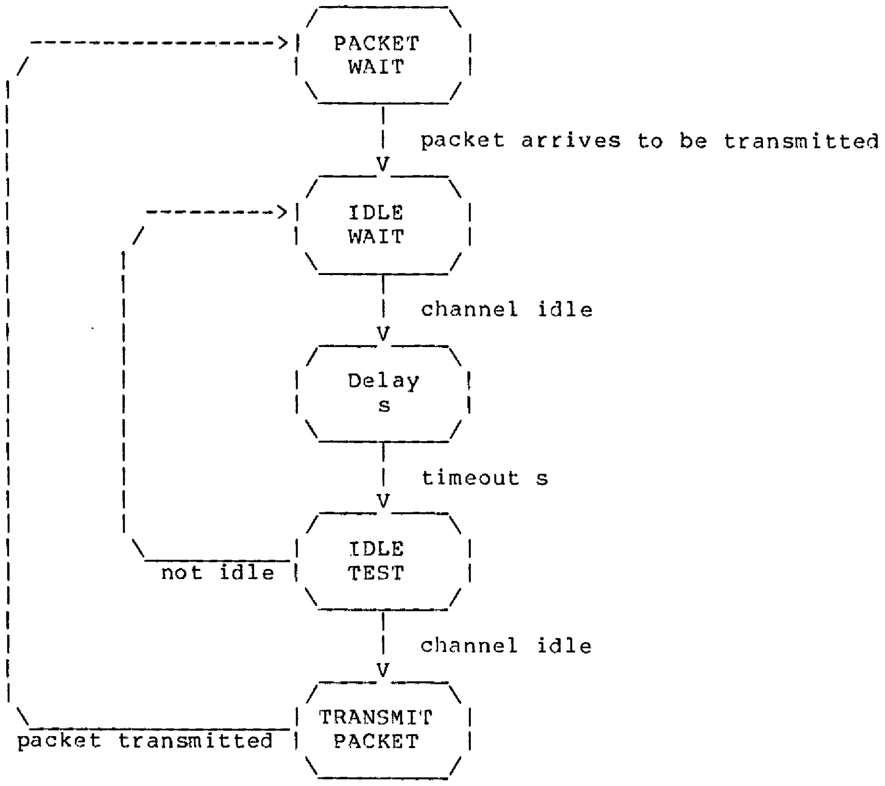


Figure 3: Transmitter Control State Machine

```
                              _______
                             /       \
      ---------------------->| PACKET |
     /                       |  WAIT  |
    |                        \_______/
    |                            |
    |                            |  SYNC detected
    |                            V
  __|__                       ___V___
 /     \                     /       \
| IDLE  |                    | ADDRESS |
| WAIT  |<-------------------| COMPARE |
 \_____/       not equal     \_______/
    x                            |
    |                            |  equal
    |                            V
    |                         ___V___
    |                        /       \
    |\_______________________| PACKET |
    |          packet error  |  LOAD  |
    |                        \_______/
    |                            |
    |                            |  receiver buffer full
    |                            V
    |                         ___V___
    |                        /       \
    |                        |   ACK  |
    |\_______________________|  IDLE  |
    |          packet error  |  WAIT  |
    |                        \_______/
    |                            |
    |                            |  CSM goes into ACK WAIT
    |                            V
  __|___                      ___V___
 /      \                    /       \
| PACKET |                   |  XMIT  |
| UNLOAD |<------------------|  ACK   |
 \______/                    \_______/
```

Figure 4: Receiver Control State Machine

The receiver control state machine is illustrated in Figure 4. When the channel is idle, the receiver waits in PACKET WAIT state until a packet is detected on the channel. The receiver then compares the Destination Address field of the incoming packet with the receiver's own address. If the two addresses are equal, the PACKET LOAD state is entered and the Data field of the packet is loaded into the receiver's buffer. When the Pad is received, the ACK IDLE WAIT state is entered; the receiver expects the channel to remain idle throughout the CSM's ACK IDLE state. The detection of any channel activity during the ACK IDLE WAIT state is considered an error in the received packet. Other possible packet errors that can occur during the PACKET LOAD state are illegal data encodings, too many bits, and bad CRC. If no packet errors have occurred by the time the CSM goes into ACK WAIT state, the ACK signal is transmitted and PACKET UNLOAD state is entered. The station must provide an empty buffer for the receiver during the PACKET UNLOAD state. After the PACKET UNLOAD state, receiver control waits for the channel to become idle and then returns to the PACKET WAIT state. Similarly, if during the ADDRESS COMPARE state, the Destination Address compares unequal to the receiver's address, or an error is detected during PACKET LOAD or ACK IDLE WAIT, then receiver control waits for the channel to become idle and returns to PACKET WAIT state.

We expect that the performance and stability properties of this scheme will be similar to those of the p-persistent scheme of Kleinrock and Tobagi [1]. (Note that the case S=0 corresponds to the 1-persistent scheme of Kleinrock and Tobagi.) The choice of the parameter S seems somewhat arbitrary; we have used S=100 but we have no performance analyses or measurements to support this choice.

The parameter A used to time the acknowledgement-period states of the CSM has been chosen to make it possible to ensure the correctness properties discussed in the following section.

3. Correctness Properties

If the automatic acknowledgement signal is to be useful, it must not be possible for a transmitting station to receive an ACK unless the intended receiver has received an error-free packet. Higher-level protocols will deadlock if the automatic acknowledgement does not have this property.

Figure 5 illustrates the type of problem that can result from an acknowledging contention algorithm in which capture can occur. Suppose the stations T1 and T2 begin transmitting simultaneously and their packets collide. Because R1 is captured, it receives good data and sends an ACK which is received by both T1 and T2. Therefore, T2 believes it has successfully transmitted a packet to R2, which is false because R2 was not captured. This may deadlock processes in

T2 and R2.

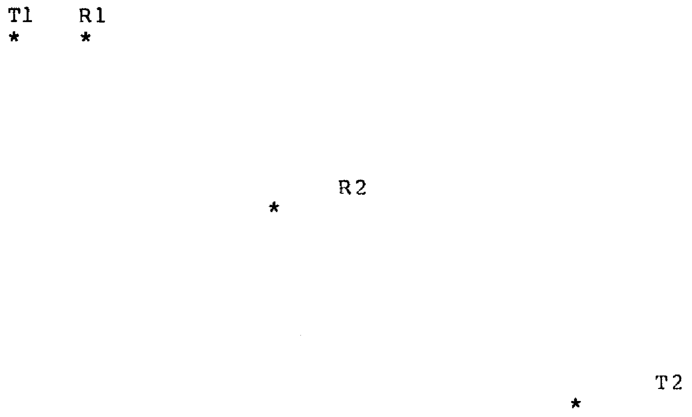


Figure 5: Radio network with a captured receiver. (The stations T1 and T2 simultaneously transmit packets to stations R1 and R2, respectively. R1 is captured by T1; R2 is not captured.)

In some cases we can solve this problem by an astute choice of transmission medium and receiver/transmitter circuitry. However, this is not always possible, as in the case of radio transmission. An alternate technique for solving the problem is to include information such as the transmitting station's address in the acknowledgement. This decreases the probability of a false acknowledgement to an acceptable level, though it is still theoretically possible. Implementing this scheme requires more costly and complex hardware because the acknowledgement information and its redundancy check (which is necessary) is no longer a fixed pattern.

The algorithm in this paper guarantees no false acknowledgements if we are careful about the placement of stations on the medium. This is a weaker constraint than being capture free, and admits the possibility of radio transmission.

To prove that the algorithm presented in Section 2 cannot falsely acknowledge a transmitting station, we consider a network with a fixed number of stations. The problems of adding new stations and removing existing stations will be discussed later.

We consider a 2-dimensional radio network with maximum diameter L. (The interpretation of results for one dimension, and the generalization to three dimensions, are obvious.) We assume that transmitted signals propagate uniformly through space. Each transmitting station captures its own receiver, and possibly other nearby receivers. A captured receiver generally receives correct data from the nearest transmitting station. All transmitted data are Manchester encoded. Receivers digitize this encoding by sampling 8 times per bit time and converting each sample, which we call a <u>mit</u>, into 0 or 1. If the receiver is perfectly synchronized with the encoded signal, a "1" bit is received as the 8-mit sequence 11110000, and a "0" bit is received as the 8-mit sequence 00001111. An idle channel appears to a receiver as a sequence of "0" mits. We will make the following Assumptions:

1. There is no noise on the channel.
2. If there is an error in a received packet, then the receiver detects the error, and does not send an ACK.
3. A collision of two or more packets causes an error in the data received by stations which are not captured.
4. In a collision, two or more superimposed "1" mit signals result in a signal which is a "1" mit.

Let the function $t(x)$ be the amount of time required for a transmission signal to propogate a distance x through the network channel medium. The end-to-end propagation delay for the network is given by

$$t = t(L)$$

Lemma 1: If a station's CSM is in SYNC WAIT state when a packet arrives, the receiver will recognize the SYNC signal, even if two or more packets have collided.

Proof: The SYNC signal consists of a "1" bit which begins with four "1" mits. By Assumption 4, a collision of two or more packets begins with four "1" mits. In SYNC WAIT state, the receiver recognizes a single "1" mit as the beginning of a SYNC signal.

Lemma 2: The CSM of each station in a network is synchronized (i.e., each state transition occurs) within time t of every other station's CSM.

Proof (by induction): Assume that each station's CSM goes through the sequence of states PACKET, ACK IDLE, ACK WAIT, SYNC WAIT, each time a packet (or set of colliding packets) is transmitted. Further assume that at the beginning of the i-th such sequence, station S is the first station to transmit a packet. By Lemma 1, each station which is distance x away from S changes from SYNC WAIT to PACKET state at most t(x) later than S. Hence, every CSM is synchronized at the beginning of the PACKET state of the i-th sequence within t = t(L) of each other. Because the duration of PACKET state is constant for each station, each CSM goes into ACK IDLE state within time t of the other CSMs. Similarly, because the durations of ACK IDLE and ACK WAIT states are constant for each station, each CSM goes into ACK WAIT and SYNC WAIT states within time t of the other CSMs.

Each station to transmit during sequence i+1 will do so when its CSM is in SYNC WAIT state. Each non-transmitting station's CSM is in SYNC WAIT state by the time a transmitted packet reaches it. The channel is initially idle, and all CSMs are initially in the SYNC WAIT state. Lemma 2 follows by induction on i.

Lemma 3: At any point in the network, two or more colliding packets must begin and end within time 2t of each other.

Proof: By Lemma 2, colliding packets must begin transmission at their respective transmitters within time t of each other. Each packet requires at most time t to propagate throughout the network. Hence, the leading edges must pass any given point in the network within time 2t of each other. Because each packet is of fixed length, the trailing edges must also pass any given point in the network within at most time 2t of each other.

Lemma 4: A station receiving a correct packet has a clear channel on which to send an ACK, except possibly for other colliding ACKs.

Proof: An ACK is sent at the beginning of the ACK WAIT state of the station's CSM. By Lemma 3, all colliding packets must end before the beginning of the station's ACK WAIT state. As an ACK propagates through the network, each station it passes must be in ACK WAIT state. By Lemma 2, no station can go into SYNC WAIT state, therefore no station can begin transmitting, before the ACK has propagated throughout the network.

Lemma 5: Assume two or more packets have collided. If at a given point in the network, the packets are offset by at least the time required to transmit a single mit (1/8 bit), then a station at that point in the network detects the collision.

Proof: The Pad of each packet ends with four "1" mits. By Assumption 4, any number of overlapped packets ends in four "1" mits. Therefore, the station detects at least one "1" mit at the beginning of its ACK IDLE state.

Let d denote the distance that a transmission signal propagates during the time required to transmit a single mit. Let the function C(x) be defined as the smallest distance such that a transmitting station S cannot capture a receiver separated from it by at least C(x), when the transmitting station S' nearest to S is separated by x + 2C(x). (That is, S' is x further away from the receiver than S.) We assume that C(x) is a monotonic increasing function of x.

For convenience, we define

$$c = C(d)$$

Consider two transmitting stations S1 and S2 located at points p1 and p2, respectively, where the distance between the two stations is

$$|p1 - p2| = d + 2c$$

Let the distances of a point p from the points p1 and p2 be denoted

$$r1 = |p - p1|$$

$$r2 = |p - p2|$$

Also consider the three regions shown in Figure 6, which are bounded by a hyperbola:

Region 1: $r1 - r2 <= -d$

Region 2: $r1 - r2 >= +d$

Region 3: $-d < r1 - r2 < +d$

Note that p1 is in Region 1 and p2 is in Region 2.

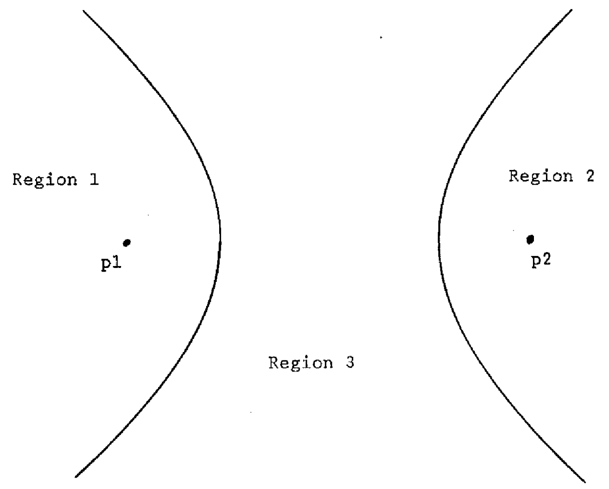


Figure 6: Three regions bounded by a hyperbola.

Assume that S1 and S2 transmit colliding packets where S1 begins transmitting at time t1 and S2 begins transmitting at time t2.

Lemma 6: If t1 >= t2, then every station within Region 2 detects a collision.

Proof: Every point p in Region 2 satisfies

$$r1 >= d + r2$$

The packet from S1 arrives at point p at time

$$\begin{aligned} t1 + t(r1) &\geq t1 + t(d + r2) \\ &= t1 + t(d) + t(r2) \end{aligned}$$

The packet from S2 arrives at point p at time

$$t2 + t(r2)$$

Therefore, the packet from S1 arrives at least

$$t1 - t2 + t(d) \geq t(d)$$

later than the packet from S2. It follows from Lemma 5 that the collision is detected by any station within Region 2.

Lemma 7: If t1 <= t2, then station S1 detects the collision.

Proof: The packet from S2 arrives at S1 at time

$$\begin{aligned} t2 + t(d + 2c) &> t1 + t(d) + t(2c) \\ &> t1 + t(d) \end{aligned}$$

By Lemma 5, S1 detects the collision.

Lemma 8: When S1 and S2 transmit packets which collide, then either S1 detects the collision, or every station within Region 2 detects the collision.

Proof: This follows directly from Lemmas 6 and 7.

Lemma 9: Station S2 cannot capture receivers outside of Region 2.

Proof: Electromagnetic radiation decays as 1/r where r is the distance from the transmitter. Therefore, by the definition of c, if only S1 and S2 are transmitting, the region in which S2 can capture receivers is given by

Region C: $r1/r2 \geq (c+d)/c$

which can be written as

$$r1 >= (c+d)r2/c$$

or

$$r1 - r2 >= d*r2/c$$

On the boundary of Region C, we have

$$r2/c >= 1$$

Therefore, the boundary of Region C satisfies

$$r1 - r2 >= d$$

which is the definition of Region 2. It follows that Region C is contained in Region 2, thus any station captured by S2 must lie within Region 2. If any other stations are transmitting (besides S1 and S2), then the region in which S2 can capture receivers is smaller, and therefore still within Region 2.

A station is said to have received a <u>false</u> <u>ACK</u> if it receives (and does not ignore) an ACK signal when the intended receiver did not send an ACK. By Lemma 4 and Assumptions 1-3, a false ACK can be generated only by a station whose receiver is captured by another transmitter.

<u>Lemma</u> 10: Station S1 cannot receive a false ACK from a station whose receiver is captured by S2.

<u>Proof</u>: By Lemma 9, any station S that is captured by S2 lies within Region 2. Therefore, by Lemma 8, either S detects the collision and does not send an ACK, or S1 detects the collision and ignores the ACK.

A similar argument can be used to prove

Lemma 11: Station S1 cannot receive a false ACK from a station captured by a transmitting station more distant than S2.

Theorem: Provided stations are placed no closer than c(2c/d + 1), it is impossible for a transmitting station to receive a false ACK.

Proof: By Lemmas 10 and 11, a false ACK can only be generated by a station whose is receiver is captured and is receiving a packet from a transmitting station within a radius of d + 2c. (Note that c(2c/d + 1) is the distance of the farthest point p from S2 in Region C defined above.) Because C(x) is a monotonic increasing function, a transmitting station within a radius of d + 2c can only capture stations closer to it than c(2c/d + 1). Because stations are not placed closer than c(2c/d + 1), it is impossible for a receiver to be captured by a transmitting station within a radius of d + 2c.

Corollary: False ACKs cannot occur in networks using non-capturing transmitters (e.g., using a coaxial cable medium and a properly designed receiver/transmitter), regardless of the spacing of stations.

Proof: For this case, c = 0.

A station can be removed from the network at any time without invalidating any of the preceding arguments.

When a new station is added to the network, the station's CSM must become synchronized with the other station's CSMs. The new station cannot transmit or receive packets until its CSM becomes synchronized. When no "1" mits have been received on the channel for a period of duration T + 2A, the CSM goes into SYNC WAIT state, and is synchronized. It is unsafe to assume the channel is idle after a shorter period of "0" mits because it is possible (though improbable) for two or more colliding packets to result in a long sequence of "0" mits. Of course, in an overloaded network, a station may have to wait a long time for its CSM to become synchronized.

4. Practical Considerations

The function C(x) must be determined empirically for a given implementation. The only implementation to date uses coaxial cable and a receiver/transmitter for which C(x) = 0. For this case, the above Corollary is of considerable practical importance. For each implementation, the function C(x) will determine the minimum station separation distance c(2c/d + 1). If stations are placed closer than this distance, then the above Theorem does not apply, and the algorithm may not always function correctly. However, the probability of a false ACK may be extermely small in such cases. It is interesting to note that the minimum station separation distance c(2c/d + 1) decreases as the speed of data transmission increases.

Our transmission algorithm has been designed specifically for local networks of up to 5000 ft. diameter. Such local networks will rarely have more than a few hundred stations, and utilization of the network bandwidth is expected to be relatively low. For larger radio networks, it may not be desirable to avoid capture. Several authors [4,5,6] have shown that capture can be used to significantly increase the capacity of a packet radio network through <u>spatial reuse</u> by allowing several successful transmissions to occur simultaneously. We feel that spatial reuse is of little value in a local network compared to the benefits of higher performance and simpler higher-level protocols afforded by automatic acknowledgements.

It seems likely that the above theorem can be extended to the case where different stations in the network transmit using different power levels. However, it is probably difficult to extend the analysis to cases of non-uniform signal propagation that can occur with FM channels.

Radio receivers typically have automatic gain control (AGC). The gain is set to a high level when the channel is idle. When a packet is detected on the channel, the AGC sets the gain to a lower level, as appropriate for the signal strength. An important consideration is that the gain must be reset to a high level immediately after the end of a packet (at the beginning of the CSM's ACK IDLE state), so that channel activity can be detected. Thus, it must be possible to pre-empt the AGC with a <u>gain reset</u> which occurs immediately after the last bit of a packet has been received, otherwise the distance d must be increased to allow for AGC recovery time.

If the automatic acknowledgement signal is to be of value, the software protocol must have certain properties. Specifically, the receiving station must be prepared to process every packet it receives, and the processing must take place in a reasonably small period of time. A station cannot choose to discard arbitrary packets because of a lack of buffer space or processing power. Also, a station cannot leave its receiver control (see Figure 4) in the PACKET UNLOAD state for too long; otherwise another station may erroneously conclude that the station in PACKET UNLOAD state is no longer in the network.

We have developed a software protocol which has the properties necessary to take advantage of automatic acknowledgements, and to avoid erroneously concluding that a station is not in the network. These properties are independent of the station's application software. The software protocol for this network will be described in a future paper.

5. Acknowledgements

We wish to acknowledge the contributions to this project made by David Bryant, Mark Gerhold, Gary Hodgeman, Larry Hopkins, Randy Johnson, and Marsh Parker.

6. References

[1] Kleinrock, L. and Tobagi, F.A., "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and their Throughput-Delay Characteristics", IEEE Trans. on Communications, Vol. COM-23, No. 12, December 1975, 1400-1416.

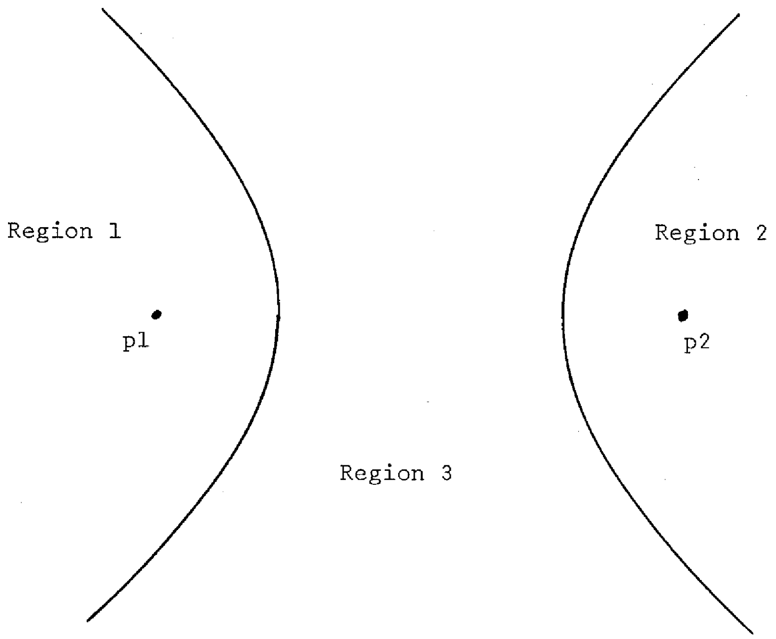
[2] Tokoro, Mario and Tamaru, Kiichiro, "Acknowledging Ethernet", COMPCON Fall 77, IEEE Catalog No. 77CH1258-3C, 320-325.

[3] Donnelley, James E., and Yeh, Jeffry W., "Interaction between Protocol Levels in a Prioritized CSMA Broadcast Network", Computer Networks, North-Holland Publishing Company, 3 (1979), 9-23.

[4] Roberts, L.G., "ALOHA Packet System With and Without Slots and Capture", ASS Note 8 (NIC 11290), ARPA Network Information Center, Stanford Res. Inst., Menlo Park, Calif. (June 1972). Reprinted in Computer Communication Review, Vol. 5, April 1975

[5] Kleinrock, L., and Silvester, J., "Optimum Transmission Radii for Packet Radio Networks", Conference Record, National Telecommunications Conference, December 1978, 4.3.1-4.3.5.

[6] Fratta, L., and Sant, D., "Some Models of Packet Radio Networks with Capture", Proceedings of the Fifth International Conference on Computer Communication, October 27-30, 1980, 155-161.


Region 1
p1
Region 3
Region 2
p2