Improving Shared Channel Access Techniques for Amateur Packet Radio

Brian Lloyd, WB6R QN

Amateur Packet Radio has come of age. There are now many packet radio users in the Amateur community, well over 20,000 at this point. In the "olden days" when there were few users on a channel at any one time the need for effective channel access techniques was much less than it is now. The problems all result in the same symptoms: poor throughput and lost connections.

Here is a list of the problems:

1. Collisions due to hidden terminals
2. Collisions due to jump-on
3. Unnecessary retransmissions
4. Congestive collapse

The first problem is called the hidden terminal. A hidden terminal is one that shares the channel with your station but cannot hear or be heard by you. If both stations attempt to use a shared resource, for instance a digipeater, the result is numerous collisions at the digipeater due to the failure to wait for the other station to cease transmitting. If you can't hear them you won't wait.

Collisions due to hidden terminals are impossible to eliminate unless you can eliminate the hidden terminals. It is possible to permit all stations to hear all other stations if you improve the radios and antennas used at packet radio stations or you can utilize duplex digipeaters that forward packets in real-time. This permits all users within range of the duplex digipeater to hear when any other station is transmitting (just like we do when we talk on our local voice repeater). This will eliminate hidden terminals.

The second problem, collisions due to jump-on, is caused by the 1-persistent nature (always transmit when the channel becomes clear) of the Carrier Sense Multiple Access (CSMA) software commonly found in our TNCs. If we examine what happens when a station is transmitting and several others become ready we readily see the problem.

1. Station A begins transmitting
2. Station B becomes ready but waits for A to stop sending
3. Station C becomes ready but waits for A to stop sending
4. Station A stops transmitting
5. Stations B and C both "jump-on" and collide

The problem here is that once a station becomes ready to transmit it will, as soon as the channel appears to be clear. The solution here is to reduce the probability that a station will begin transmitting when the channel goes clear when there is more than one station ready to transmit. This technique is known as p-persistent CSMA. The probability for transmission 'p' is varied in order to reduce the chance for collisions without undue effect on throughput. Here is how it works:

1. A station has data to send and waits for the channel to become clear
2. When the channel becomes clear the station generates a random number and compares it with the value of p (the transmission probability). If it "wins" (the random number is less than p) it transmits. If it "loses" it waits for one slot time (a tunable parameter) and repeats this sequence.

This sequence of events guarantees that the station will eventually transmit. The random factor will cause significant variability between the time the channel goes clear and a given station transmits. This provides greatly improved channel sharing.

Let us look at our original scenario and see what happens if the two stations were to use a value of 0.5 for p (50% probability of transmission when the channel goes clear). There are four possible states when the channel goes clear: neither B nor C transmit, B transmits but C does not, C transmits but B does not, both B and C transmit. The result is that the probability of a collision drops from 100% to only 33% and the probability that a station will transmit without a collision is now 66% (after some number of slot times). This is a considerable improvement and will significantly reduce the number of retransmissions required.

Another subject of interest is high incidence of unnecessary retransmissions due to the sending stations failing to wait for a sufficient period of time before retransmitting an apparently lost packet. The firmware that is currently installed in TNCs uses a very simplistic algorithm to set the value of the retransmission timer. This time, known as FRACK, is set to a fixed value, usually 3 seconds. This value is multiplied by n+ 1 where n is the number of digipeaters in the address chain. The result is that the retry timer will be set to 6 seconds when packets are being routed through a single digipeater regardless of all other channel activity.

The problem with this simplistic algorithm is that the retry timer at the sending station may expire while waiting for an ACK when the channel activity is so high that the receiver is unable to respond for a period of time that exceeds the period of the retry timer. The result is that the sending station retransmits the packet when it really should be waiting longer.

There is a solution to this problem; use an adaptive algorithm to adjust the value of the retry timer to suit the channel conditions. In order to make this possible, the stations must collect information about actual round-trip packet time. Each time a packet is transmitted, the sending station should start a timer. When the ACK is received, the timer is stopped. These times should be run through a filtering algorithm to derive a smoothed round trip time (SRTT). The retry timer should then be set to a value greater than or equal to the SRTT (in fact it should be set much longer than the SRTT to accommodate transient peaks in the round trip time). Since the software will arrive at a value for retry time that is based on current channel conditions rather than an arbitrary value, excessive retransmissions will be avoided in the case of heavy channel loading and excessive delays will be avoided in the case of light channel loading. Here is the formula for smoothed round trip time:

$$SRTT' = (1 - \alpha) * RTT + \alpha * SRTT$$

Where:

$SRTT$	= the previously calculated value for the smoothed round trip time
$SRTT$	= new value of $SRTT$
α	= a parameter ranging from 0 to 1; if not adjustable, a suggested fixed value is 0.9
RTT	= round trip time measurement for the current packet

This formula will produce a smoothed calculation of the round trip time. The value of α determines how much effect the new round trip time (RTT) will have on the new value for SRTT $(SRTT')$. Large values of α will give a slow response to changes in RTT while small values of α will give more weight to RTT. In essence, this formula is a low-pass filter for changes in the value of RTT.

So far I have discussed solutions for many problems but I haven't dealt with the problem of "prime time," that period of time in the evening when so many stations are on the air at once that even with p-persistence and a round trip timer, many packets will be lost due to collisions anyway. It is the case with any sort of shared channel that if an excess of traffic is offered, the throughput will drop, leading to more transmissions, more collisions, and even less throughput. This is a downward spiral leading to a malady known as congestive collapse. The only solution to congestive collapse is to reduce the amount of traffic offered to the channel. Round trip timing will help by responding to the delays and slowly increasing the amount of time between retransmissions. There is another, short-term solution known as backoff.

The purpose behind backoff is to have the stations using the channel very rapidly increase the amount of time between retransmissions. In the case of binary exponential backoff, the time between retransmissions is doubled each time a packet is retransmitted. If our initial retransmission timer is set to ten seconds, the time between first and second retransmissions would be 20 seconds, 40 seconds between second and third retransmissions, 80 seconds between third and fourth retransmissions, and so forth. The result of this is to have the stations retransmitting data rapidly reduce the amount of traffic offered to the channel, permitting the channel to recover and begin moving information again.

The value of all the above suggestions has been proven in the Washington, DC, area. In this area, there are many stations using the KA9Q TCP/IP networking software. The KA9Q TCP/IP networking software in conjunction with the KISS TNC implements p-persistence, round trip timing, and binary exponential backoff. During periods of heavy channel activity when both TCP/IP and "ordinary" TNCs were using the channel, TCP was in all cases able to move traffic when the TNCs were losing connections. The only symptom visible to the operators of the TCP/IP stations was periods of longer delay in the delivery of traffic. Status information garnered from the TCP/IP software showed the round trip time increasing and examination of the transmission characteristics showed that binary exponential backoff was called into use when the traffic was exceptionally heavy.

While this test was not performed scientifically, the results were obvious; TCP/IP, using the above mentioned channel access techniques, delivered traffic when ordinary TNCs were losing connections. A compromise was finally reached when persistence was dropped at the TCP/IP stations to a level below 20% to permit ordinary TNCs priority on the channel. This had no effect on the operation of TCP/IP other than increased delays.

In conclusion it appears that a solution can be found to the problems of collisions due to hidden terminals, collisions due to jump-on, unnecessary retransmissions, and congestive collapse. The problem of hidden terminals can be addressed through the use of duplex digipeaters or guaranteeing that all stations can hear all other stations in a given local area. Collisions due to jump-on can be avoided through the use of p-persistent CSMA. Unnecessary retransmissions can be avoided by the use of round trip timing in the setting of the retransmission timer. Congestive collapse can be avoided through the use of a retransmission backoff algorithm such as binary exponential backoff.

RF spectrum for the amateur radio operator is limited and we must do as much as possible to use what we have more effectively. These techniques will permit us to make the greatest use of whatever shared channel resources are available.