AREDN Bandwidth: Where did it go?

And how do we fix it?

Tim Wilkinson - KN6PLV (tim.j.wilkinson@gmail.com)

AREDN mesh networks don't appear to have the bandwidth you might expect. We look at what may be happening and how to fix it.

TL;DR

Low SNR links between nodes break the Linux “auto distance” algorithm resulting in poor bandwidth utilization on node links. This document proposes an experimental application - Link Quality Manager - which is installed on nodes to better manage links. 10x bandwidth improvements have been observed.

Expected link speed vs actual link speeds

We’ve all pointed an AREDN node at another node, found the sweet spot for the best SNR, and then been underwhelmed by how much bandwidth there seems to be. WiFi is famous for over-reporting how much bandwidth is available vs what you actually get (think 1/6th in many cases), but somehow we all expect a 2 mile link with SNR > 20 to do more than 1 Mb/sec. So what’s actually going on here? Unfortunately actual performance data is difficult to come by, and experiments to see what might improve (or not) are difficult to coordinate. But there are some theories.

Performance theories

Hidden nodes

The classic problem with CSMA networks (AREDN WiFi is CSMA) is that of “hidden nodes”. A better explanation of this can be found here, but in short CSMA works by nodes listening for transmitters before themselves transmitting. This can fail when nodes are spread out so only some nodes can hear others resulting in many transmitting simultaneously, corrupting data, requiring retransmissions and wasting bandwidth.

And while this could be a real issue for AREDN, it only has an effect when there are a lot of collisions. And for there to be a lot of collisions there has to be a lot of traffic ... and there just isn’t[1]. AREDN networks are largely idle. Probably the biggest traffic generator is OLSRD, and on

[1] This is an assumption; but a reasonable one given the services currently offered on the mesh. The highest bandwidth demand is for video (of which there is little) and photos (mostly monitor cameras). People are not staring at these constantly.

the SF network (as of today) it accounts for about a few kilobytes/second. Statistically it is difficult to ascribe the bandwidth problems to hidden nodes.

Bandwidth decimation

Bandwidth decimation is where too many radios are using the same channel between similar points, and so rather than adding bandwidth, each radio ends up with a share of the original.

As noted above, AREDN doesn't currently use much of its available bandwidth. There is some overhead in having multiple radios on the same channel, even if they're not very active, but it is not significant. Compare this to a home wifi setup; if many devices are constantly using a lot of bandwidth then it is definitely noticeable; otherwise, not so much.

Checking the status of various Omni antennas (picking one type of node at random) on the SF mesh there are rarely more than 5 neighbors to each node, and never more than 10. If everyone was transmitting constantly there would be a problem; but they're not.

CSMA vs TDMA

TDMA is more efficient than CSMA and avoids many of its problems. However, Linux currently has no implementation of the protocol; it is restricted to proprietary radios. And while there's nothing wrong with this, it means that AREDN as currently envisaged cannot support it. Which doesn't mean TDMA radios don't have their place in an AREDN network, but they seem better suited for planned, backbone operation, rather than for random Ham/Hacker deployment.

That said, racing to embrace TDMA without understanding why the current CSMA network is failing is problematic. If hidden nodes aren't the issue, and network utilization is too low for bandwidth decimation, how would TDMA fix this? What actually is the problem?

Alternate theory

Coverage Class

The WiFi standard "IEEE Std 802.11TM-2007, Part 11" briefly discusses "Coverage Classes". This defines the air time propagation for the wifi signal. For in-home networks this parameter is unimportant as devices are close together, and the actual time it takes for packets to transfer between devices is essentially zero. For long distance networks this parameter is more important. A wifi signal propagates at approximately 1 mile every 5 microseconds - fast, but a significant time especially over 10 and 20 mile links. The coverage class is designed to make allowance for this propagation time. If the coverage class is too high, devices will wait longer than necessary to retransmit failed packets. If the class is too low, devices will retransmit packets unnecessarily. Having an appropriate coverage class for a long distant network link is very important for optimal performance.

Auto-distance

AREDN provides a “Distance to FARTHEST Neighbor” setting which, indirectly, allows the Coverage Class to be set (the Linux kernel calculates the coverage class from the distance setting). Temptingly, an “auto” option is provided and is used by most (all?) people. What does “auto” do?

“Auto” enables a “dynamic ack” algorithm in the kernel which automatically adjusts the coverage class. The adjustment is based on the timings of packets sent to, and acknowledged from, other devices. The class will always be large enough to handle the most distant device. This sounds great in theory but it is in fact a big problem in AREDN.

AREDN is an open, ad hoc, network; any node can associate with any other node so long as they use the same channel and bandwidth. This results in distant nodes, with very low SNRs, being associated with each other. Unfortunately the dynamic-ack algorithm does not care that these links are, essentially, unusable in the network, and still adjusts the coverage class to accommodate them. The result; a higher coverage class than actually required for network operation, resulting in longer delays in packet retransmission. This compounds the already increased retransmissions inherent in longer links and further reduces bandwidth.

Experimental solution: Link Quality Manager

The Link Quality Manager is a platform for testing this “broken auto-distance” theory. It can be installed on any current nightly build and runs automatically in the background. Once running it performs a number of tasks:

1. Blocks radio links which are also DtD links
2. Blocks radio links which have too low an SNR
3. Blocks radio links which are too distant
4. Sets the node’s Coverage Class based on the most distant non-blocked node which is a direct, routable neighbor.

What does that mean?

1. Occasionally nodes will have directly (DtD) connected nodes which are also on the same channel (perhaps there are multiple directional nodes close together). Although the DtD link should be preferred by OLSRD, this block makes sure any radio link between is always ignored.
2. We want to ignore links with SNR too low to be useful. The application uses an adjustable deactivate and active level; drop below the deactivate SNR and the link is blocked until the SNR is above the activate level (the hysteresis avoids links bouncing in and out of a blocked state). This stops OLSRD from using poor links (if for some reason it thinks this is a good idea).
3. We want to limit how far a node can be from this node and still be acceptable, even if there is a high SNR. The more distant a node, the lower the bandwidth of the link

between us and them. In addition the entire link's bandwidth is affected by the most distant node we communicate with. We may want to limit this. We automatically determine the distance between nodes using the latitude and longitude information available from each node's api.

4. The node's automatic distance detection is disabled; we manage it here. The manager selects any link to a node which has not been blocked, and checks the link is routable (ie. there is at least one route in the mesh which uses this link). From the nodes remaining, it selects the greatest distance and uses this to calculate the coverage class.

The Link Quality Manager refreshes its state every 60 seconds and adjusts the blocked nodes and coverage calculation. A simple UI is available at `/cgi-bin/lqm` which shows the state of each node and link and allows adjustment of various parameters.

What it doesn't do

The Manager blocks nodes by blocking traffic from the appropriate MAC address. What it doesn't do is block nodes from associating with the radio. It would be ideal to either ban "poorly performing" nodes from associating with a radio, or alternatively telling our node not to associate with distant radios. However, the ad-hoc wifi mode used in AREDN does not appear to support this[2].

Results

Without deploying the Link Quality Manager across a cluster of nodes which are immediate neighbors, it's difficult to know how well this works in practice, or if this theory is crushed by reality. In small scale desk testing between a few nodes, it provides improved bandwidth. On a single two mile link, an improvement of 10x was observed.

Next step

The next step is to try this experimental code on a larger scale; perhaps 10 interconnected nodes where bandwidth tests can be conducted before these changes and after.

Other thoughts

Given the way the coverage class in WiFi works, there is probably some merit to separate out long and short distant links onto different channels, assuming this isn't already being done for different reasons.

[2] If it is, and someone knows how to do this (not just read the manual where it suggests it works, but has actually succeeded in doing this … I did try) please let me know.