Connecting AREDN meshes using Supernodes

Tim Wilkinson (KN6PLV)

Proposal

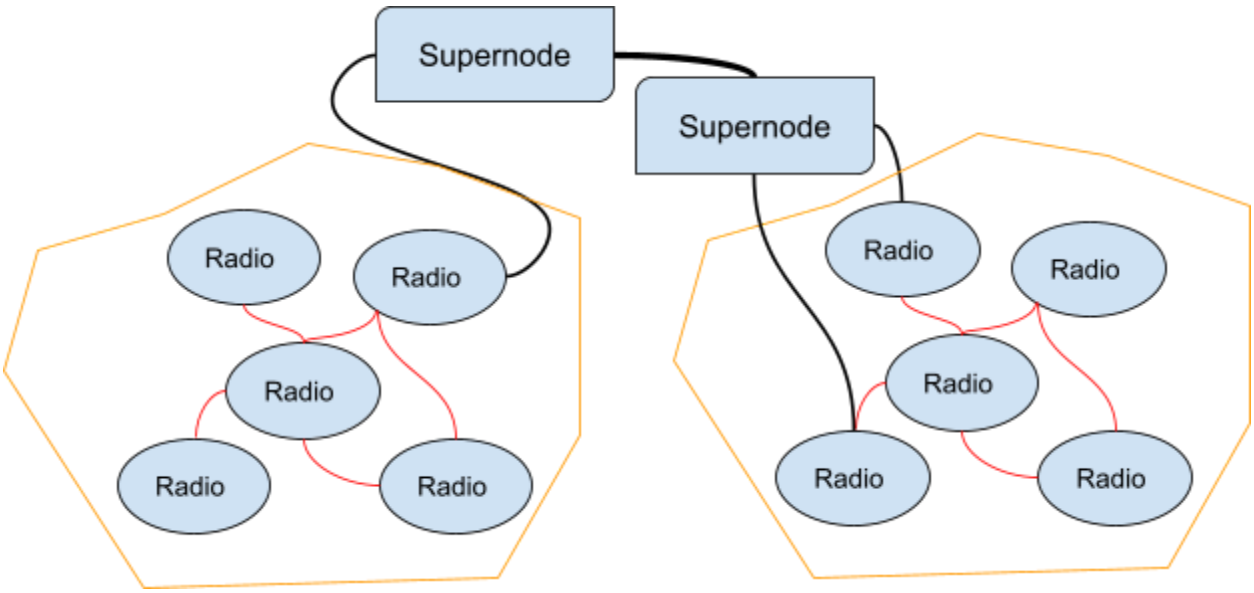
In this proposal we present a way to create a unified, global, AREDN mesh without making any changes to the current AREDN infrastructure. It requires no hardware or software changes and uses no extra memory, cpu, power or bandwidth on individual AREDN nodes. Instead we propose creating a group of “supernodes” which connect the current AREDN networks, route data between the meshes but isolate OLSR traffic to avoid mesh overload. These supernodes act as a second-tier mesh. The approach is cooperative, peer based, and opt-in with every mesh maintaining ownership and control. Importantly, it has none of the drawbacks of the AREDN cloud tunnel server.

AREDN Technology

Each node in AREDN manages its network routes using the OLSR software (described here). The size of AREDN networks are limited by the memory, cpu and bandwidth requirements of the OLSR software. Node resource requirements scale linearly with the number of nodes in the network which limits the total network size. Consequently, large state, country or world sized networks are not possible.

Supernodes

Supernodes are designed as bridges between mesh networks, allowing multiple networks to be connected without increasing per-node costs.



Supernodes do not merge networks into one big mesh (AREDN cloud tunnel server) but instead provide connections between discrete meshes.

Local and Remote

For a mesh network with supernodes, we define a node to be in our local mesh network or in an unknown remote network. Local networking is unchanged. Connections to unknown remote nodes are routed to a local supernode.

Addressing

In a traditional AREDN network, each node in the network knows the address of every other node. But what happens to an unknown node address? Each node also has a default gateway; a node in the network to send unknown traffic to. This is usually a WAN gateway node which bridges the AREDN mesh to the Internet. But AREDN also supports other gateways. We include a supernode gateway for an unknown mesh node address; an address which is not definitively known but has the form 10.X.X.X. Unknown traffic is sent to a supernode gateways. These supernodes announce themselves to the local mesh using “Host and Network Associations” (part of the OLSR protocol) so no changes are necessary to local mesh software.

Supernodes

A supernode is a server running the OLSR software and knows how to reach any nodes in any of the mesh networks it connects to (directly or via other supernodes). Once traffic arrives at a supernode, it is forwarded to the mesh containing the final destination node. A supernode might know about 10,000 nodes (unlike a typical AREDN mesh which knows perhaps 200) and requires more memory, bandwidth and cpu than a mesh node. These requirements are not particularly large and could run on a small server.

Isolation

To allow a supernode to keep a map of the entire, multi-mesh network, there are some minor differences to the OLSR software a supernode runs. These changes keep the meshes connected to supernodes isolated from each other; OLSR traffic can flow out of a mesh into a supernode, but OLSR traffic never flows from a supernode into a mesh. From the perspective of a mesh node, a supernode looks like one more mesh node. From the perspective of a supernode, everything is one big mesh. This distinction is important and allows meshes to connect without being overloaded.

Multiple Supernodes

A single supernode would become a single point of failure, but because supernodes also use the OLSR protocols, we can have many, all operating as peers.

Multiple supernodes are connected to each other. Multiple meshes can be connected to multiple supernodes. Supernodes can be connected to meshes in multiple places. The only requirement is that all supernodes can reach all other supernodes directly or via other supernodes.

Supernode Requirements

The hardware requirements for a supernode are modest. For networking Linux uses a routing technique called LPC-tries. This provides routing lookups with almost constant time regardless of number of routes and consumes only 128MB of memory.

Supernodes need to handle all traffic between meshes. Linux can easily handle 100's of GB/s so the limitation will be the physical network used to connect supernodes to each other and to the meshes.

OLSR also imposes some overhead which grows as the number of nodes grows. These overheads are manageable assuming supernodes are connected using fast (100Mb/s), reliable interconnects.

Overhead

The number of messages a supernode receives scales linearly with the total number of nodes in all connected meshes. How much traffic is this?

A supernode receives a management message from every node in the network (all nodes in all meshes) every 5 seconds. With a typical message size of 100 bytes, a supernode receives 20 bytes per second per node. At the time of writing (August 2021) there are 4,300 AREDN nodes registered world-wide, so a supernode for this network would receive 84KB/s (or 0.7Mb/s); not a particularly large bandwidth requirement.

Topology

We propose the following topology for AREDN supernodes. Each local mesh is connected to one or more supernodes. A supernode is only ever connected to a single mesh and may be connected at multiple points. All supernodes are connected to each other, directly or via other supernodes.

Ownership

By having only a single mesh connected to each supernode, the owners of each mesh can be responsible for their own supernodes. This simplifies management and maintenance. There is also some fault isolation as a failed supernode will only affect the connection of one mesh. Additionally, bandwidth shapers can be used between the supernode and mesh to limit external traffic throughput.

Name Services

AREDN uses standard DNS to locate hosts and services. OLSR broadcasts host and service names through the mesh, with every node maintaining a local list. Supernodes prevent these messages from flooding between meshes (overwhelming mesh nodes). This also means that, by default, a local mesh node cannot lookup a name on a remote mesh. Local name lookups are unaffected.

Supernodes have two ways to manage names across meshes; give each mesh its own unique domain name (or zone), or unify all names under the familiar **local.mesh** suffix.

Multiple Zones

With this implementation, supernodes do not share host and service names between supernodes which avoids mixing similar names from different meshes (**X.local.mesh** could exist in multiple meshes). Instead, each supernode maintains its own DNS Zone with a unique zone name (e.g. **sfwem.mesh**). Supernodes run standard DNS services (e.g. BIND9) to manage this global set of names.

Unified Zone

With this implementation, each supernode creates a single unified zone containing all the known names from all supernodes. This means that duplicate names will result in multiple IP addresses being returned from a query (which might cause confusion).

Configuration

It is not possible for nodes to use these global names automatically. Instead a device needs to opt-in and set their default nameserver to an appropriate supernode. For a mesh node, the default nameservers (usually 8.8.8.8 and 8.4.4.8) can be changed on the appropriate configuration page.

Pilot

To demonstrate this approach to connecting multiple meshes, a pilot study was conducted using two supernodes to connect two meshes; the Bay Area mesh and the Southern California Mesh. The goal of the pilot was to validate the claims made in this proposal. The pilot used two Debian machines, each connected by a tunnel to one of the two mesh networks, and connected to each other over ethernet.

Results

The connected supernodes and meshes performed as expected. Traffic was successfully routed between meshes, the supernodes maintained copies of the topology of both meshes, but neither mesh became aware of the other (no node number explosion). In total, 600 nodes were connected together, and the traffic between the supernodes to support this was 15 bytes/sec/node (a little better than the original estimate).

Conclusion

This proposal describes a mechanism to expand the reach of a local AREDN mesh network to all AREDN networks. This is achieved without changes to the hardware or software of the current networks and without increasing resources used by nodes. Meshes are connected using supernodes which route traffic between meshes without imposing additional costs on the meshes themselves.

References

AREDN cloud tunnel server

Optimized Link State Routing Protocol - OLSR

Linux routing performance

DNS Zone Files

BIND9

Supernode code (GititHub)