Hocnet a payment based mesh network protocol built on Batman-Adv

2016-12-10

1 Work in progress

This paper is incomplete, feel free to provide feedback via Github issues

Contents

2 Abstract

As the number of connected individuals and devices expands the ‘last mile‘ continues to be the greatest challenge both in the connected and developing worlds, representing a disproportional portion of the cost and difficulty of connecting the world.

Ad-Hoc networks are a type of network where there is no dedicated infrastructure such as switches or routers and instead these tasks are handled by the participating users in a distributed fashion. Mesh networks are a derivative of Ad-Hoc networks in that dedicated hardware for routing is used and encouraged but not ultimately required for the functionality of the network.

In this paper we propose a plethora of additions to the existing mesh network protocol Batman-Adv with the purpose of creating a 'decentralized last mile' where each node competes to be paid to carry traffic via the most efficient route it can use. Furthermore we will encourage the use of so called ‘Ad-Hoc infrastructure‘ where users may implement dedicated hardware such as directional antennas, IR links, or simply mundane wires to produce better and more profitable routes for their nodes to advertise.

We hope that by lowering the barrier of entry to providing connectivity services we can both decrease the cost of connecting the developing world and provide an environment that fosters competition in otherwise monopolistic last mile infrastructure in developed countries.

3 Batman-Adv

Batman-Adv is the layer 2 version of the original BATMAN protocol outlined in [1]. Conceptually the protocol is very simple. Every originator period every node participating in the network broadcasts an originator message containing a MAC address, which identifies the node that ‘originated‘ the message as well as the MAC address of the last node to forward it. When an originator message is broadcast every node that receives the broadcast updates the expanded bandwidth field with its own estimation of the bandwidth between itself and its neighbor if and only if the predicted bandwidth is lower than the value already in the originator [3].

As the originator messages spread through the network each node checks each originator message it receives against its own local routing table, either to be added or updated if the throughput advertised by the received originator message is better or the sequence number of the message is higher. Since originator messages contain the last hop that forwarded the message this data combined with throughput and sequence number is used to maintain an up to date routing table with the best first hop along the path to any other node in the network.

To send data to any other node it is sent to the neighbor node listed in the local routing table, who will forward that data in turn based on it's own copy of the routing table. This process continues until the data is delivered to its destination.

Batman-Adv is implemented as an up-stream Linux kernel module with several optimizations from the original concept, including network coding to batch the broadcast of originator messages and several optimizations focusing on allowing hybrid networks containing traditional Ethernet links as well as wireless links [2]. Perhaps the most promising feature of Batman-Adv is that is remains competitively performant when compared to other mesh protocols [4]

4 Modifications to Batman-Adv protocol spec

4.1 Weak black-hole attack protection

For our purposes we will define a black-hole attack as any node which increases the throughput metric or decreases the price on any originator message it receives and rebroadcasts.

When a connection between two nodes across the network is opened, both nodes regularly produce a signed äckm̈essage. This ack message contains:

- A timestamp.
- How much data has been received from the other node.

When each side of the connection receives the ack message, they compare the information about how much data was received with their records of how much data was sent. From this, they can estimate the throughput of their route to the other node. Other measures are also possible. If the ack message contains the number of packets received, they can estimate the packet loss rate by comparing this with the number of packets they sent.

They then forward the ack on along the route. Other nodes along the route can use the ack message to make the same estimate. They compare the traffic they have forwarded from the source to the traffic that the destination says it received from the source.

This way, all the nodes participating in a connection for at least one full ack period can make an estimate about the true quality of the route.

This provides enough information for the network at large to take action against false routing metrics. Each node participating in an active connection will be able to estimate an accurate quality metric to at least one of the ends of the connection, once every ack period.

4.1.1 Active vs. passive throughput tests

The above is a passive throughput test- it just watches the connection without taking action itself. If a connection is not saturated during an ack period, the throughput measurements may be innacurate. Some methods of estimation can give numbers that are too low, since the link is not being fully utilized. If a node is only trying to send 50Mbit/s, that doesn't mean that 50Mbit/s is the full capacity of the connection. Other methods can give numbers that are too high. If the link is uncongested and experiencing no packet loss, such methods may report the throughput as being the maximum theoretical throughput of the network interface.

Either way, the issue is a lack of information about the total capacity of the route. The solution is to combine passive throughput tests with active throughput tests. Active throughput tests attempt saturate the link to find its capacity. If they are coordinated with the passive throughput test, they can gather valuable information about the link. However, if it is possible to detect when an active throughput test is happening, dishonest nodes on the route could treat test traffic differently from regular traffic. This could lead to inaccurate estimates and needs to be dealt with.

4.1.2 Duration of throughput corrections

Another question is for how long to continue to apply the adjusted metric after receiving an ack. If this period is too short, then nodes will go right back to trusting an inaccurate metric that they recently had to correct. If this period is too long, then nodes will miss out on legitimate updates about newly improved routes.

To strike the right balance, exponentially increasing time-out will be used for bandwidth corrections beyond a small tolerance. A node that has participated in a correction will record the time it last performed a correction for a given route T_{last} and the size of the correction δ. When participating in another correction on the same route the amount of time to apply the bandwidth correction after the ack has expired will be determined as:

$$T_{adjustment} = \frac{C_1}{T_{now} - T_{last}}^{\delta C_2}$$

Where C_1 and C_2 are constants to be adjusted and hardcoded. This formula will make it take longer to go back to using a neighbor's advertised metrics if the advertised metrics needed to be corrected recently, and/or required a large correction.

4.1.3 Local reputation

The adjustment timing mechanism above provides some defense against nodes making bad route advertisements. If a node makes an inaccurate advertisement, its future updates will be disregarded for a time. The more inaccurate the advertisement, the longer the time. However, if the node simply changes its public key, it will appear as a new node. Its updates will again be taken at face value.

Also, the entire mechanism so far only verifies routes that are being used. The power of distance-vector routing protocols is that they proactively propagate information about all routes on the network. When a node receives a packet, it is able to route the packet to the best next hop immediately. There needs to be some way to gain information about the reliability of routes that have not yet been verified.

If a node has seen that a given neighbor's route advertisements are generally very accurate, it can be confident of two things. One, that the neighbor itself will not be the source of any route advertisement innacuracies. Two, that the neighbor is able to correct for the innacuracies of nodes further away.

These assumptions hold for routes through that neighbor which have never been used or verified. In this way, we can transfer the information gained from the verification of a few routes to many other unverified routes through that neighbor.

Because of this evidence of accuracy, it makes sense for to have a preference for routes through reliable neighbors. A naive way to do this would be to simply improve the routing metrics of reliable neighbors. For example, a reliable neighbor would get 30% extra throughput added to routes that they advertised, if throughput was the metric being used.

However, this would fail to produce the desired effect in the presence of new unreliable neighbors advertising wildly inaccurate routes. If a new neighbor pops up advertising an (inaccurate) throughput of 1gbps, the extra 30% added to a reliable neighbor's accurate advertisement of a route with a throughput of 100mbps would not make a difference. The new neighbor's inaccurate route would still be chosen, and would be used until it was verified and discovered to be inaccurate.

One way to solve this is to use a tiered system. New neighbors with no routes that have been used and verified go in the lowest tier. The remaining tiers are made up of neighbors with higher degrees of reliability. During route selection, routes through neighbors from higher tiers are chosen more frequently than those in lower tiers. This would prevent very inaccurate routes from circumventing the system, while still allowing new neighbors a chance to prove themselves.

Each tier is used a certain percentage of the time. For example the lowest tier is used 5% of the time. When routing traffic to a destination, 5% of the time, the best neighbor from the lowest tier is used. The rest of the time, neighbors from higher tiers are used.

If a node makes inaccurate route advertisements it is moved into a lower tier.

This is still vulnerable to sybil attacks on the lowest tier. There is nothing stopping a node from making 1000 fake new nodes to increase the odds that that one of its fake nodes will be chosen instead of other new nodes. There's also nothing stopping neighbors in the lowest tier from advertising inaccurate metrics.

The higher tiers are not as vulnerable, since to get into a higher tier, a node needs to have demonstrated accurate route advertisement already. Also, nodes in the higher tiers are obviously incentivized to make sure that their advertisements are accurate to avoid losing their status.

So, in this scheme, there needs to be some minimum of cost to get into the lowest tier to protect against sybil attacks. We will leave this unspecified, as different use-cases could use different mechanisms.

References

[1] David Johnson, Ntsibane Ntlatlapa, and Corinna Aichele. *A simple pragmatic approach to mesh routing using BATMAN* (2008)

[2] Cigno, R., and Daniele Furlan. *Improving BATMAN Routing Stability and Performance* (2011)

[3] Quartulli, Antonio, and Renato Lo Cigno. *Client announcement and Fast roaming in a Layer-2 mesh network* (2011)

[4] Murray, David, Michael Dixon, and Terry Koziniec. *An experimental comparison of routing protocols in multi hop ad hoc networks* Telecommunication Networks and Applications Conference (ATNAC), 2010 Australasian. IEEE, 2010.

[5] Britton, Matthew, and Andrew Coyle. *Performance analysis of the BATMAN wireless ad-hoc network routing protocol with mobility and directional antennas* Military Communications and Information Systems Conference (MilCIS), 2011. IEEE, 2011.

[6] Hu, Yih-Chun, David B. Johnson, and Adrian Perrig. *SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks* Ad hoc networks 1.1 (2003): 175-192.

[7] Perrig, Adrian, et al. *SPINS: Security protocols for sensor networks* Wireless networks 8.5 (2002): 521-534.

[8] Hu, Yih-Chun, Markus Jakobsson, and Adrian Perrig. *Efficient constructions for one-way hash chains* International Conference on Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2005.

[9] Lundberg, Janne. *Routing security in ad hoc networks* Helsinki University of Technology, http://citeseer. nj. nec. com/400961. html (2000).