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:

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

Where C_1 and C_2 are constants to be adjusted and hardcoded. The goal of this formula is to provide a longer adjustment time based on nodes stopping quickly returning to adjusting the same route and further increased by the magnitude of the route adjustment since these characteristics fit the expected profile of a malicious node that can't provide any actual use to the network.

4.1.3 Duration of throughput corrections

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).