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 it's 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 it's 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-streamed 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 Addition of cost criteria

Specifically we propose a new field be added to Batman-Adv originator messages, this field is simply a half-precision floating point value for the cost of transmitting a pre-arranged number of packets in a single direction, an originator creating a new originator message for broadcast into the network would initialize this value to their 'hop cost'. A node rebroadcasting a received originator message would update this field by adding their own hop cost to the existing value before forwarding.

The process for selecting which originator message to retain and rebroadcast is likewise updated to account for the cost field. Instead of choosing the best originator message using the throughput field alone the ratio of cost to TQ is chosen.

Broadcast based protocol overhead, such as originator messages, will be unbilled, this provides an unfortunate avenue for the use of stenography to encode free data. We propose a quick solution by limiting the frequency of overhead packet forwarding to the default issuing rate. In this way a malicious client could in theory reduce their broadcast rate to encode data for free, but in exchange said client would actually be reducing the overhead on the network rather than adding to it.

4.2 Payment chains

The design of Batman-Adv, as well as any reasonably low overhead mesh protocol, specifically avoids requiring any node to have full knowledge of a route it is using. From the perspective of any given member of the mesh only the next hop and the cost required to reach the destination are known. To negotiate with each node along any given route to figure out who to compensate for routing packets is not only unsupported by Batman-Adv but

is impossible to do efficiently. Mesh nodes would spend a significant amount of time and funds paying intermediate nodes to forward negotiation packets to every link in the route until a connection was finally open. To make a bad situation worse, real world testing shows that in worse case scenarios routes can flip as frequently as once every several seconds [5]. While protocol modifications can improve this alarming figure it's not realistic to stop and renegotiate paths until a billing relationship is established with every node any given route may traverse. Furthermore paying each node in a route individually raises the issue of very infrequent paths, as routes move due to network conditions it's entirely possible many nodes will owe funds to each other in such small quantities as to be infeasible to transfer without the protocol overhead or transaction fees negating the benefit.

As noted in section 4.1 each node adds their own price on top of the existing one as they forward originator messages, from the perspective of any given node the only known cost is the cost of the entire route being advertised by that originator message. A node that wants send traffic would select the best route from the originator messages it has seen and pay the first node in that path the full cost of the route. Each node would then be expected to take off their advertised share and pay forward.

This obviously introduces a level of trust that must be balanced on the other side of the seller/buyer relationship, if the bandwidth buyer parts with their funds and has only reduced trust to threaten a seller that does not deliver the relationship imbalance and transitory nature of each nodes identity makes a situation ripe for exploration.

To counterbalance the risk of a zero trust node attempting to take the money and run all bandwidth will be purchased on credit. With payment to be delivered at some future time if services are in fact rendered. In the zero trust version of this relationship the seller must deliver a connection, at least for a very brief period, before demanding payment from the buyer. As trust is built the payment interval and the size of the credit can be increased and the threat of reducing trust becomes more significant as the efficiencies of delaying billing increase.

Perhaps the greatest advantage of this method is the ability to separate routing and billing as components of the network. From the perspective of the routing code cost is just a criteria to optimize and trust does not exist, from the perspective of the billing code only knowledge of and communication with adjacent nodes is a concern.

This separation allows billing to operate as a user level program while routing code operates the kernel level with only a few hooks in between to facilitate cooperation. Most of the following sections addresses routing level

requirements, billing is more fully specified in section 5.

4.3 Weak black-hole attack protection

With the addition of profit incentive a so called 'black-hole' attack becomes very appealing to potential attackers. A malicious node can falsely advertise routes that it is unable or unwilling to complete, or can complete but only in a worse fashion than advertised [9].

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 is opened every fixed number of packets received, or some fixed time interval whichever is shorter, a node must produce a signed ack, this message will contain a timestamp, the bandwidth metric to the destination node, and a bit to indicate if the ack was sent via the timer or packet number trigger. If for some reason this ack is not delivered in a reasonable amount of time a node initiating a connection can assume it's packets are not arriving at the destination and that some node along the path is dropping packets.

While this information is useful it is not necessarily delivered to the nodes that need to have it. It is entirely possible for a duplex connection to not share any nodes between the send and receive paths. In this case we need to prove to the nodes attempting to send packets to a destination that their route to that destination does actually exist. To do this we will forward ack messages in a circular fashion, when a node sends an ack message it will chose the last received ack from the other end of the connection and append it to it's own ack before sending. This way any node that is part of the routing loop receives data about both sides of the connection it is participating in.

This comes together to provide enough information for the network at large to take action against false routing metrics, with the minimum billing interval set to one ack period a node attempting to advertise routes and simply drop packets will never be paid because each node participating in the connection will note the lack of an ack message and discontinue the connection without billing.

If the originator message period is set to some fraction of the ack message period the network should converge onto a new route faster than the ack period requires. Meaning a missing ack heavily implies malicious behavior or large scale network failure, for a nodes participant in multiple connections with the same destination (as neighbor nodes to the malicious node likely are) can easily observe with near certainty that no acks are being produced

from specific routes and blacklist the involved neighbor for those routes for some random period.

If the same behavior is noticed again after the blacklist period an exponential random back off can be used, the reasoning for this is that nodes adjacent to malicious nodes are likely to continue to see the same behavior when their timers expire, while further away nodes have a lower chance of doing so. This slowly isolates the blacklisting until it converges on the guilty node.

The same method can be used when it's observed that the throughput metric is higher or the price is lower in the ack than in the originator message a node is forwarding as it can be inferred that some node along the route to the destination must be malicious.

In conjunction with blacklisting nodes the minimum trust to forward traffic over a new neighbor can be raised such that a blacklisted neighbor can not return under a new identity to perform the attack again quickly.

Provided tests demonstrate that the convergence period for this strategy is feasible it should provide robust protection against stationary attackers. Mobile attackers will eventually run into raised trust levels required to route traffic over new neighbors across the entire network. The major downside of this method of security is that hostile networks create hostile communities trust wise, where it is difficult to build enough trust with neighbor nodes to participate in the network rather than simply buy bandwidth from it.

5 Payment models

The billing program, preliminarily named Scrooge (backronym to be determined), requires read only access to the routing table, the ability to send messages to adjacent nodes, and the ability to block originator messages, traffic or both from adjacent nodes.

During the initial key exchange a number of billing codes can be encoded in the message, advertising what methods of payment any given node can accept. It's then up to the node initiating traffic to either find a mutually acceptable payment method of blacklist the node. Billing info messages will have a hash signature as well as a symmetric one, allowing for easy verification although not easy generation. In the worst case a well connected node with many neighbors that has not yet built any trust may have to perform a large number of asymmetric key operations until trust has been established and the period of billing messages can increase.

5.1 Cryptocurrency

TODO read about lightning network

5.2 Semi centralized cryptocurrency

5.3 Semi centralized traditional currency

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).