# 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 transmission quality (TQ) field with it’s own estimation of the link quality between itself and the node that sent the originator message. Note that the TQ field is not simply replaced by every hop, but is updated using several factors that boil down to a single field in the originator message for which ’higher is better’ for more details see appendix A in [3].

So 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 TQ advertised by the received originator

message is better or the sequence number of the message is higher. Since originator messages contain the last hop to forward the message this data combined with TQ and sequence number is used to maintain an up to date routing table with the best first hop to any other node in the network.

To send data to any other node it is sent to the best first hop, who forward that data in turn to it's own best first hop to the destination and so on until the data reaches it's destination node.

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 TQ field as the sole criteria 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.

Furthermore, for the time being all bandwidth in either direction is paid for by the node initiating the connection, this is useful in that it simplifies

the billing structure for the intended use of last mile consumer connectivity but is not a requirement of the protocol design.

### 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 the most appealing, so it is the first we will address. A malicious node can falsely advertise incredible routes that it is unable or unwilling to complete, grinding the network to a halt as traffic, and in this case payment as well, is directed into a 'black-hole' that seems to be an incredibly optimal route. [9].

For the sake of our protocol we will define a black-hole attack as any node which increases the TQ or decreases the price on any originator message it receives and rebroadcasts. Attempting to detect this falsifying of information is impossible without embedding O(N), where N is the size of the network, information into originator messages or other communication, attempts to reduce this to some constant overhead are faced with the possibility of a single malicious node impersonating an arbitrary number of nodes until whatever sliding window of verifiable TQ and price values has passed.

To prevent this attack we propose measures at both the routing and billing levels. Nodes receiving connections must send a unicast signed ack containing the number of packets received and sent since the last ack the ack period can either be a timer or based on number of packets sent. This allows for direct computation of the actual TQ of a connection, in the case of a malicious node simply dropping packets nodes would within one ack period begin broadcasting the actual TQ of that route, zero. On the billing level the malicious node is never paid because services where never rendered. In fact the largest concern for this form of attack is using it to disrupt connections with pure malicious intent and no profit motive.

We can address pure malicious intent at the billing level. To function

the billing level must be able to determine which adjacent nodes can be communicated with, by separating the ability to block originator messages from the ability to block all traffic the billing code can allow an adjacent node to buy bandwidth without ever routing any of it's own traffic over it. Different trust levels can be required for selling to a node and buying from a node, these values can initially be the same but when a malicious route is found, specifically a route where the node fails to observe an ack after the given period and some tolerance, the node participant in the malicious route can be marked as 'sell only' on a timer and the minimum trust threshold for 'sell/buy' relationships increased. Note that it's possible the node participant in the malicious route is still above the sell/buy threshold for trust. The goal of these measures is that for nodes far away from the malicious node these values will slowly trend back to normal as ack period pass without any malicious routes found, but the area of the network with the attacker will see another malicious route immediately once the timer expires, resulting in the timer being instituted again with exponential backoff for that node. Since the minimum sell/buy trust has also been increased if the malicious node where to reconnect under a new identity it would find it impossible to start the attack again without building trust. For stationary attackers this would quickly become infeasible, for mobile attackers the entire network threshold continues to increase, this isn't ideal as it provides a way to create a barrier to entry for new routes but we prioritize network functionality over fairness.

The single remaining case is where a malicious node advertises better TQ than it can achieve but does actually forward the packets as promised. Since actual TQ can be determined through ack packets and the values in the rebroadcasted originator messages for that route are updated as a matter of course with actual values the route will quickly normalize to the actual transmission quality. As a added bonus the throughput of a connection can be added as a criteria for TQ when ack packets are available allowing nodes to increase or decrease the TQ based on measured bandwidth which can't be inferred from packet loss of normal broadcast traffic.

### 4.4 Key exchange bootstrapping

Cryptography is needed to both identify adjacent nodes for trust and billing relationships as well as sign protocol messages to prevent tampering, a key field will need to be added to originator messages so that each node can share their public key. It's unfortunately infeasible to prevent man in the middle attacks in any way that provides hard guarantees such as a Diffie Hellman key exchange due to the infeasibility of initiating a unicast connection channel

with every other node on the network. Billing is the only area where strong guarantees of trust are required and directly performing a key exchange with adjacent nodes is perfectly doable.

The one area of concern is a malicious node attempting to impersonate another node on the network and receive some or all of their traffic. We will assume there are multiple paths that an originator message containing a given nodes chosen MAC address and pubkey can travel, in the case where only a single route exists or all routes will perform an attack there is no possible corrective measure except presharing of keys.

If more than one node is observed trying to use the same MAC address nodes will drop the later originator messages, in the case of a race condition a node will not forward the second MAC/key pair it receives and then select whichever MAC/Key pair is in the majority of originator messages with that MAC several message periods later, essentially the better connected node will win and the loser will simply pick a new MAC address and try again.

Therefore it's safe to trust the keys included in originator messages for routing purposes. At the routing levels originator messages and ack messages are both signed and must be verified when received to prevent the forwarding of incorrect information.

### 4.5 Optimizations using hash-chains

Asymmetric encryption is a costly operation, even though we only propose security features on network overhead packets and leave security of the actual traffic to the layer 3 protocol our current proposal involves message verification operations proportional to the network size every originator message period, even using RSA with small keysizes a modest node node will struggle as network size increases.

Using one way hash-chains we can provide a similar level of security with a much less costly cryptographic primitive. In a one way hash chain a node chooses a random seed S and uses a secure hash function H to create a hash chain of length n.

$$H(S) = V_n$$

Where $V_n$ is the end of the chain. From that point $V_{n-1}$ ... $V_0$ are computed by.

$$H(V_{n+1}) = V_n$$

Using this method a node generates an arbitrary but finite chain of length n. Data can be symmetrically signed or otherwise encrypted with $V_i$, which is

then transmitted to the network. Some period of time later $V_{i-1}$ is revealed to the network and can be used to verify the authenticity of the earlier message. By maintaining a longer history a node can verify that a given series of messages must originate from the same sender.

By embedding a chain $V_n$ and $V_{i+1}$ in each originator message $V_n$ can be used to to verify the previous originator message sent by that node, secured with $V_{i-1}$ and $V_{i+1}$ can be used to verify the originator message containing it once $V_{i+2}$ is released by the next originator message. When $V_n$ is finally reached a new chain is generated and it's $V_0$ is placed in an originator message that is verified using S.

In the name of additional efficiency we propose the use of the same secure hash function H for generating the message signatures. Where $M_{sig}$ represents the originator message signature and || represents binary append.

$$M_{sig} = H(Timesamp||MAC||V_n||V_{n+1}....)$$

Once $V_{n+1}$ has been revealed in the next originator message this signature can be computed and verified. By requiring the next originator message to verify the current one we effectively add a latency of one originator message period to all overhead operations if we are to verify the contents of each message before using them. It is possible that the contents of the message could be used initially and then discarded if the message fails verification, at the very least it can be confirmed that.

$$H(V_{n-1}) == V_n$$

Which proves that some valid version of the originator message from that node exists in the network, although it's possible that the received version has been modified in transit until verification is complete. To reduce this latency we propose that if nodes only see one version of a given originator message with a valid $V_n$ it should be accepted immediately but placed in a queue for verification [7, 8].

### 4.5.1 Persistence of Trust

While hash chains offer significantly better performance they do not provide an easy way to verify that you are speaking to the same node after instances of power loss or even intermittent connection loss. Resuming verification is possible after receiving more than one correct originator message but without a constant pubkey there is no way to verify that the identity of those node has not changed while the node was away. Since trust is an integral part of the

billing system this means that we can not eliminate the need for asymmetric key cryptography entirely.

On the other hand, unless you have a billing relationship with a node trust is not affected, meaning pubkeys only need to be shared with adjacent nodes. Even better only one asymmetric key operation needs to be performed to verify that the owner of the pubkey is the owner of a given hash chain, after which all billing messages can be verified using only the current $V_n$ for any given node.

### 4.5.2 The large network problem

In a case where the longest path through the network is longer than the originator message period it's possible to perform a rather complex attack. An attacker would have a low latency connection across the diameter of the network, they would pick up originator messages at one end of the network, wait for the next message to reveal the key, then use that key to forge a false originator message and broadcast it before the original message reaches that point in the mesh. Victim nodes would be faced with two valid, conflicting originator messages.

A proposed solution is to use the checkpoints previously proposed in 4.3, when a node producing a checkpoint in an originator message produces their signature $M_{chk_n}$ which is the $n^{th}$ checkpoint signature.

$$M_{chk_n} = H(V_{chk_n}||V_{chk_{n+1}}||Timestamp_{chk_n}...Price_{chk_n}||TQ_{chk_n})$$

By signing the original contents of the message as well as the checkpoint contents no additional space is used because the hash output is a fixed size, but it becomes possible to verify what version of each originator message each saw exactly, as opposed to only the price and TQ observed. Provided the checkpoint interval and maximum number of checkpoints remains properly tuned for the maximum network diameter controlled by the originator message TTL a checkpoint should exist on any far flung originator message either before the next key reveal or too shortly after to reliably attack.

### 4.5.3 Dealing with attackers

Using asymmetric key cryptography it was feasible to simply drop originator messages that failed to validate, while it's still possible to drop obviously invalid originator messages where the revealed $V_n$ does not match $H(V_{n-1})$ various race condition attacks are possible.

A new verification queue, that only updates the routing table after verification for nodes with originator messages that failed to verify is possible, but opens up the possibility of using the solution as an attack in and of itself to delay updates about a specific node in the network. Since we assume ad-hoc infrastructure will make up the backbone of the network increased originator message periods are unlikely to re-route traffic, which would be the major motivation for such an attack. For the time being we leave this problem open.

# 5 Payment models

The billing program, preliminarily named Penguin (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 neighbours 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).