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

The largest challenge in providing internet services to both already connected and emerging internet populations is the so called 'last mile'. With the explosion number and dramatic decrease in cost of connected devices ad-hoc networks and by extension mesh networks have looked increasingly attractive as solutions to connectivity problems.

In this paper we explore the possibility of a for-profit mesh service for participation and use of the general population as a last-mile internet service provider with infrastructure created from a dynamic mix of of semi specialized consumer hardware and professional equipment inserted at key points.

3 Feasibility of mesh networking as a primary ISP

3.1 Participation

WiFi enabled devices are ubiquitous and often spend a significant amount of their active time idle, the potential of using a network of normal consumer devices to replace expensive, difficult to construct, and often monopolistic last mile infrastructure is enormous.

The challenge is both motivation and enablement for the average consumer to participate in such a network. We propose a protocol in which a node will route data to an exit gateway and each node along this path will be paid at a rate of that nodes own choosing. In later sections we will examine the techniques for providing such a payment infrastructure, in this section we focus only on the feasibility of the network itself. Feasibility being defined as reasonable cost, connection speed, and complexity for a network base on mostly consumer hardware, with the possibility of professional hardware being profitable at network choke points being left open.

3.2 Network throughput

Throughput in wireless mesh networks is limited both by packet loss and channel availability. While each individual node may be capable of several

hundred megabits per second of wireless throughput actual throughput is cut in half immediately if the transmission must be rebroadcast on the same channel. Having two devices operating on different channels can mitigate this issue at the cost of compounding the problem of available channels.

(TODO the below is very rough, missing citations and possibly misleading, determine a good way to model a hybrid network with hard links put in at choke points due to profit incentive)

There are three non-overlapping 2.4ghz WiFi channels available in the United States and 6ish channels with varying bandwidths available in the 5ghz spectrum. While other channels are available (most interestingly high power wireless A for rural users and 60ghz wireless AD for ultra dense urban environments) we will restrict ourselves to these bands as they are the most common. Many of the available 5ghz channels require DFS compliance to be used and are ignored by most consumer hardware. As this is a software issue more than a hardware one we will make an assumption in our favor there.

At 300mbps single direction maximum per 2.4ghz N channel and 800mbps single direction maximum per 5ghz AC channel. This is a grand total of 4.8Gbps per 5ghz signal range and 900Mbps per 2.4ghz signal range, in this case smaller signal ranges due to walls or other physical interference work both for and against the network. By isolating nodes total bandwidth available to each node increases, but the number of potential hops to reach an internet gateway increases.

Batman-Adv has a maximum of three retires for wireless interfaces in its ARQ scheme. (mix of worst case and best case estimates, maybe move everything to average case? Insufficient data for average case)

3.3 Network latency

(determined by number of hops and network coding for batched packets, which might only apply to originator message's)

3.4 Cost of operation

(power consumption of a node use to estimate floor price of a hop, this combined with avg number of hops to gateway estimate should determine floor price of bandwidth, should be cost feasible by a mile)

4 Modifications to Batman-Adv protocol spec

4.1 Challenges to payment

To understand this section you should be familiar with the BATMAN protocol in broad strokes [1]. It's a short and very accessible paper, please read it if you have not already. TODO create BATMAN protocol summary section.

While the BATMAN protocol has significant inherent overhead this is addressable through optimization [2, 3] and it provides real world performance comparable to more conceptually complex protocols [4]. This allows for the addition of payment in a relatively straightforward manner.

4.2 Addition of cost criteria

Specifically we propose a new field be added to BATMAN 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.

Protocol overhead 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.

4.3 Payment chains

The design of BATMAN, 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 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 expect good performance out of a protocol that requires negotiating with many nodes individually. 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 without the individual amounts ever exceeding the fees for issuing the transactions.

The solution we propose involves having route prices advertised as 'lump sums', where the cost of the entire route is paid to the first node in that route and it's expected that each node will pay forward all except it's own share. Since nodes are much more likely to send significant amounts of traffic over each adjacent node as opposed to any given node in a network this should result in the vast majority of traffic being feasible to pay out even with a significant amount of route flipping. We will spend the next several sections exploring what guarantees and security we can provide before addressing payment in further detail in 5

4.4 Weak black-hole attack protection

As noted in the previous section we refer to nodes advertising false routes, this is just one subset of what is known as a block-hole attack. A malicious node can falsely advertise routes that it is unable to complete, grinding the network to a halt as traffic is directed into a 'black-hole' from which is does not escape [9]. While we can easily prevent the aforementioned attack as well as many other scams by having all bandwidth purchased on credit we can not use this method to prevent a well connected node from attracting more than its share of traffic by fudging a better TQ on each originator message it passes.

Since Hocnet is expected to use and encourage the use of Ethernet bridges, IR bridges, and other ad-hoc infrastructure it's possible for a node to have a near perfect TQ to another node at an arbitrary distance. Making it impossible for any given node to determine the validity of an originator message without direct communication with the originator.

To resolve this problem using constant originator message space we propose the addition of time stamps in originator messages. The originator will place a signed time stamp into originator messages. After some predetermined period t a node forwarding an originator message will be required to sign the current TQ and route cost with its public key and attach this data to the originator message. This process will be repeated for each time period up to a small but arbitrary number of times m.

originator messages found to be lacking these values after the prescribed time will be dropped. This allows any node receiving an originator message to verify that the route cost and TQ have decreased and increased respectively since the last checkpoint. The strength of the security this provides against malicious originator messages is inversely proportional to t, where a small enough time period will devolve into the same O(n) scheme discarded earlier in this section.

While this algorithm provides no strong guarantees we hope to tune it such that black hole attacks aren't effective on anything but very small subjects of the network.

4.5 Clock synchronization bootstrapping

The above section requires a that all nodes maintain a synchronized clock. TODO describe method to bootstrap clock sync passively using observed neighbor originator message time stamps. TODO replace originator message number with time stamp?

4.6 Key exchange bootstrapping

To prevent impersonation it's critical that two nodes sharing a MAC address but using a different pubkey can not exist. We propose a simple solution where any originator message using an identical MAC address to an existing originator message but a different key will simply be dropped by the receiving node. Combined with a reasonable policy for evicting old entries for nodes not seen on the order of hours should be sufficient.

TODO think more about possible attacks here

4.7 Potential 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 node modest 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].

4.7.1 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.

4.7.2 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 4.4, 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.

4.7.3 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

Payment negotiation at the mesh level was mentioned in 4.3 without much detail beyond the method. This is an intentional chose to abstract as many of the details of billing as possible into a user-space program, in this way we can embed a payment negotiation protocol into Hocnet at a higher level instead of being forced to write it into the kernel itself. 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 routing to a given adjacent node by removing it from the routing table.

5.1 Cryptocurrency

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).