B.A.T.M.A.N Unpacked: A Guide to the Protocol's Fundamental Concepts

José Afonso Gastaldi de Araújo Teixeira, Leander Seidlitz*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: joseafonsogat@gmail.com, seidlitz@net.in.tum.de

***Abstract*—B.A.T.M.A.N is a routing protocol designed for ad-hoc mesh networks. Throughout its development, various versions of the core algorithm were released that addressed issues of previous ones. This paper explains the core ideas behind the B.A.T.M.A.N protocol and highlights the differences between its different versions.**

***Index Terms*—B.A.T.M.A.N, ad-hoc mesh networks**

1. Introduction

Mobile ad hoc networks are often used as a replacement for traditional structured wireless networks where a client has to communicate directly to an access point that alone holds all the routing knowledge. In a mobile ad hoc network, each device (usually called a node) that is part of the network follows a protocol that allows it to relay information (packets/frames) to its local-link neighbours, which comprise all nodes that are close enough for direct communication. This creates a path across various nodes that ultimately allows two mutually distant nodes to communicate.

This type of network was first developed for military applications in the early 1970s [1] but as technology advanced and commercial devices such as cellphones and laptops got smaller, cheaper and more powerful, there was also more motivation to develop protocols that handle self-creating, self-organizing and self-administering networks that do not rely on fixed structures to function.

The B.A.T.M.A.N (Better Approach to Mobile Ad-hoc Networks) was developed by the Freifunk community in Berlin and since the release of kernel version 2.6.38 it is part of the official Linux kernel. This paper aims at explaining the core components of the B.A.T.M.A.N protocol and summarize the improvements that have come along with every generation of the protocol.

2. B.A.T.M.A.N

The development of the B.A.T.M.A.N protocol started around 2006 and, as of completion of this paper, includes 5 major versions/generations. One can think of generations I to V as the abstract ideas, the blueprints of the protocol, whereas B.A.T.M.A.N Deamon (batmand) and B.A.T.M.A.N Advanced (batmanadv) are the actual implementations/programms that turn the idea into reality.

B.A.T.M.A.N is a mobile ad hoc protocol well suited for mesh networks with unstable links. Naturally, other protocols such as the OLSR (Optimized Link State Routing) protocol exist, but what makes B.A.T.M.A.N stand out is that in this environment there is no need for nodes to gather knowledge about the state or topology of the network any time. Even though the algorithms of different generations differ from one another, the basic idea remains the same. Each node must only rely on the metadata received (or the lack of it) by Originator Messages (OGMs) that are broadcast in the network to decide about the best next hop to forward packets. How the nodes use the information from the OGMs varies from generation to generation because of the neighbour ranking system used in each one.

Next, we dissect the Originator Message and formally present each field contained in such packets used in B.A.T.M.A.N III. Later versions of B.A.T.M.A.N introduced new fields, but the ones from B.A.T.M.A.N III continue to be the backbone of newer OGM versions.

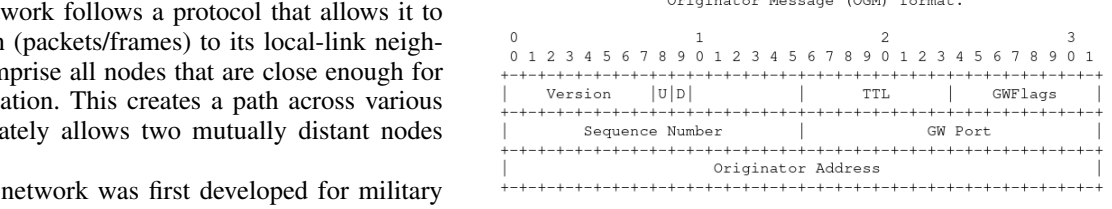

Figure 1: Layout of an Originator Message [1]

- **Version**: Value defined by the protocol; if a packet has a different `VERSION` number than expected, the packet is ignored.
- **Is-direct-link Flag**: Indicates whether the neighbour node is also a direct neighbour of the Originator or not.
- **Unidirectional Flag**: Indicates whether the neighbor node is bidirectional or not.
- **Time to Live (TTL)**: The maximum number of hops an OGM can undergo before it is dropped.
- **Gateway Flags (GWFlags)**: Set to 0 if the node is not an internet gateway; otherwise it contains the connection bandwidth (upload/download).
- **Gateway Port (GWPort)**: Set to 0 if the node is not an internet gateway; otherwise, it specifies the tunneling port to use.
- **Sequence Number**: The number that uniquely identifies the OGM in a given timeframe.
- **Originator Address**: The IPv4 (MAC address in B.A.T.M.A.N Advanced) of the B.A.T.M.A.N interface that generated the OGM.

These data are essentially all that is required by the protocols to establish the correct and efficient paths among nodes in the mesh, allowing them to make well-informed and timely routing decisions. (Fields related to internet gateways are explained in more detail in 6).

 doi: 10.2313/NET-2024-09-1_02

In the next sections of this paper, each neighbour ranking system and the differences in the format of the OGMs are explained in detail. It is also important to note that generations I and II are mentioned in B.A.T.M.A.N III's section since they are considered to be initial flawed prototypes in the early development of the protocol [2].

3. B.A.T.M.A.N III

In this section, a comprehensive explanation of how Originator Messages flood the network precedes the description of the neighbour ranking system of the third generation of B.A.T.M.A.N. [1], to facilitate a better understanding of the protocol.

3.1. How Does the Protocol Operate?

Every node in the network broadcasts Originator Messages periodically, with the purpose of announcing its existence to other nodes. Every OGM contains a sequence number which uniquely identifies it, enabling the distinction between new and duplicates OGMs. That way an OGM is only counted once.

An Originator (an interface of a node) will create an OGM with all the metadata necessary, as listed in section 2, and broadcasts it to all its local-link neighbours. These neighbours will again process the OGM and foward it to all of their neighbours. It is easy to see that the network will be flooded with OGMs until the packets are lost or the Time To Live (TTL) reaches 0 which will make a receiving node silently drop the packet.

A neighbour Ranking System was implemented based on the number of unique OGMs a neighbour has relayed from a specific Originator, where the neighbour with the most relayed OGMs is considered the current Best Link to the Originator. This ranking process is very dynamic and can change path as new OGMs are broadcasted.

Later when a node receives a standard non-OGM packet, it will know immediately which neighbour is the next best hop towards the destination and will forward the packet to that chosen neighbour, ultimately completing the purpose of the protocol.

3.2. Neighbour Ranking System

Each node maintains a list of Originators [1] it knows of, with some basic data collected about them from OGMs received. For each known Originator, an entry is created that contains all important information about this Originator, such as address, gateway capabilities, Host Network Announcement (HNA) list, and the current sequence number. For each Originator there is also a frame called a NBRF (Neighbour Ranking Sequence Frame) [2], nicknamed "sliding window", in which a fixed number of the last OGMs received from that specific Originator is stored (as an ordered list).

When a new OGM arrives from that specific Originator, a new entry is posted in the sliding window along with the identification of the neighbour from which the OGM came. If the window is full, the last entry is dropped. If the OGM's sequence number is greater than the current sequence number (current greatest), this field is updated with the new incoming sequence number.

The best ranking neighbour for that Originator is constantly re-evaluated, selecting the one with the most entries (OGMs received from) in the sliding window, accredited as the best next hop for communication with that specific Originator.

3.3. Generation I and II Design Flaws

Generations I and II are considered to be initial prototypes in the early development of the protocol [2]. Generation III is a direct upgrade that fixed the design flaws. Below are the main flaws of these versions.

3.3.1. Generation I. B.A.T.M.A.N does not check if a link is bidirectional or not. This is a design flaw because B.A.T.M.A.N only uses bidirectional nodes; this means that if a node receives packets from another node, it must also be able to send packets back. Unidirectional links are useless and a burden to the network under Generation I.

The simple fact of receiving a self-originated packet from a neighbour already suggests that that link is bidirectional, but to confirm this, a node will keep a record of the sequence number of its self-originated OGMs when broadcasting it to its neighbours. When the node possibly receives a self-originated OGM from one of its neighbours, it will check whether the sequence number is within a certain range (BI_LINK_TIMEOUT) of the node's current sequence number. If it is, and the Unidirectional Flag (UDF) is not set, then the check is successful. In Generation II a bidirectional link check was implemented.

3.3.2. Generation II. The inherent flaw with this version is that a node will retransmit all OGMs it receives from its neighboring nodes. This might not look like an issue at first, but when a node forwards OGMs from all its direct neighbours, it may create an illusion of quality that is not real, and can even create loops. Here is an example:

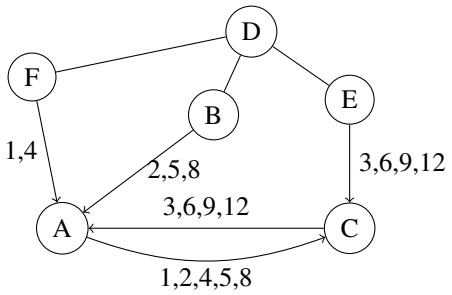

Figure 2: Not Best-Ranking Neighbour Problem [2]

The numbers next to the arrows correspond to the sequence numbers of OGMs originated from D. If a packet has to be forwarded from node C to D, it is easy to see that the best path towards D is through E. That is because E has relayed the most amount of OGMs from D, 4 in this case, whereas nodes B and F have just relayed 3 and 2 OGMs, respectively, which indicates packet loss through these paths. The issue is at node A, because it relays OGMs coming from both nodes, B and F, to C, resulting in A relaying 5 OGMs in total to C, which creates an illusion of it being a better path than it actually is from C's point of view. Generation II chooses A as the next best hop towards D, even though A is connected to weaker links.

To solve the issue a node must, upon receiving an OGM, only foward it if the neighbour the packet came from is the current best next hop towards the Originator, otherwise the packet is processed and then dropped.

4. B.A.T.M.A.N IV

B.A.T.M.A.N III's main problem concerns the asymmetric links between nodes. An asymmetric link between two bidirectional interfaces occurs when packet loss in one direction is different from that in the other direction, which may lead to a false analysis of the perfect next hop towards an Originator.

An example would be a network with nodes A, B, and C where all nodes are connected. OGMs from Node A propagate to B and to C. Since the links are asymmetric arbitrary packet loss values are chosen. B receives 100% of packets from A but A receives only 5% of the packets from B. C receives 90% of A's packets, which in turn also receives 90% of C's packets back. Finally, B and C receive 80% of the packets sent to each other. If the network is using B.A.T.M.A.N III, node B would think it has the perfect link towards A, which would be a single hop, but in fact this is not the best choice to make, as there is a huge packet loss from B directly to A. The better choice would be to hop to C and only then jump to A.

B.A.T.M.A.N IV [3] introduces a new concept of how to determine the best next hop towards an Originator, the quality of transmission, i.e., how likely it is a neighbour node will actually receive the packets sent. The key variables are Receive Quality (RQ), Echo Quality (EQ) and, of course, Transmit Quality (TQ) [4]. These metrics are always between a node and a specific neighbour.

- RQ: number of OGMs received from a specific neighbour (generated by this neighbour) divided by the number of expected OGMs in a given time frame.
- EQ: number of self-originated OGMs received back from a specific neighbour divided by the total number of self-originated OGMs in a given time frame.
- TQ: the node calculates this value by dividing the Echo Quality by the Receive Quality from its neighbour $\rightarrow EQ \div RQ$. It estimates the reliability of a transmission to a neighbour.

The Originator Message is also updated to carry a 1-byte long TQ value. When a new OGM is created the TQ value is set to the maximum and with each hop, this value is readjusted and tends to decrease. Three main aspects influence the adjustments of the OGM's TQ:

1) The local transmit quality of the node. When a node receives an OGM, it will multiply its local TQ towards the previous sender by the incoming TQ, which is the value found in the OGM. This way, each node receiving an OGM packet knows the TQ towards the Originator of the message. When forwarding the packet to other neighbours the updated OGM's TQ should be updated to $TQ = TQ_{\text{incoming}} \times TQ_{\text{local}}$
2) Penalties due to asymmetric links. Networks that use Wi-Fi, for example, use acknowledgment messages (ACKs) when a packet is sent, so the sender gets the confirmation that the message actually arrived, otherwise the sender will continue resending the message until a timeout occurs. If a node knows that the Receive Quality of packets towards a neighbour is low, it will adjust the TQ of the OGM before sending it to that specific node, as to: $TQ_{\text{new}} = TQ \times \text{asym}$ where $\text{asym} = (100\% - (100\% - RQ)^3)$
3) Penalties due to the number of hops. Even in a perfect scenario where every path from A to B holds a TQ of 100% it is still advantageous to choose the path with fewest hops, since more hops usually translate into more latency. BATMAN IV makes sure that at every hop the node receiving the OGM will decrease its TQ value by a fixed value, regardless of any other penalty before forwarding the packet.

Each node keeps "sliding windows" of types local and global. A node will have a local sliding window for each of its neighbours so it can track the last $TQ_{\text{LOCAL\_WINDOW\_SIZE}}$ (the amount of entries that the window can hold) OGMs received from that neighbour to calculate the current link quality. A node will also have a global sliding window for each originator that the node has knowledge of. Within this window there will be the average TQ values of each distinct neighbour leading to that specific originator.

In summary, nodes will use these sliding windows to make a competent analysis of which neighbour is best suited for the next hop towards the destination.

5. B.A.T.M.A.N V

B.A.T.M.A.N V [5] is the latest generation of the protocol by the time this paper is completed. It deviates from the packet loss metric implemented in the previous generation, as this time the focus is on packet throughput. It was observed that B.A.T.M.A.N IV is not optimal in dealing with meshes with nodes that present little to no packet loss, but diverse throughput capabilities. Furthermore, to diminish the nodes' overhead due to processing so many OGMs with such short intervals between them, a "divide and conquer" approach was used. The original OGM, used until generation IV, had the purpose of finding neighboring nodes through bidirectional checks and, most importantly, to flood the network with link quality information so that nodes could make better routing decisions. B.A.T.M.A.N V divides these functionalities into two distinct types of packets, ELP (Echo Location Protocol) packets and OGMv2 packets. Each of these two types of packets will be described in the following subsections.

5.1. ELP

These packets, similarly to OGMs, possess version numbers, sequence numbers, and the Originator's address. Once a neighbour receives such a packet, it will process it by first performing the necessary checks to ensure the validity of the packet, and then updating its Neighbours List with new information. If the neighbour is new and is not yet in the list, the list gains a new entry; otherwise, the `Last Time Seen` and `Last Sequence Number` fields of the existing entry for this neighbour are updated.

The key distinction with ELP packets is that after their processing, they will not be re-broadcast as an OGM would; their path ends after one single hop. (It is only used to keep the receiving node's Neighbours List updated.)

5.2. OGMv2

It is similar to the previous generation OGMs, but instead of carrying the TQ value, there is a new field for the throughput measurement. When a node interface receives this type of packet, it will perform the usual validity checks on version, source, destination and self-originated

message. If the OGMv2 passes the initial checks, its sequence number is checked if it is within the range of a "protection window". The packet is dropped if the sequence number is too old or unexpectedly new.

The interface will then update the information it has about the Originator, such as the sequence number and last seen timestamps. Because the throughput of the path towards an Originator is as high as that of its weakest link, the throughput value may need to be updated if the throughput of the neighbour that receives the OGMv2 is lower than the throughput value in the OGMv2. There is also a 5.8% hop penalty applied to the throughput value in order to create a decreasing metric over multiple hops.

If the OGMv2 already comes from the current best neighbour towards the Originator, the packet is just re-broadcast. If it does not come from the best neighbour, but the OGMv2 throughput is higher than that of the best neighbour, the best neighbour is updated and the packet is re-broadcast. If the packet neither comes from the best neighbour nor has a higher throughput, it is not forwarded.

6. Connection to Outside Networks

Using the B.A.T.M.A.N protocol, standard nodes are able to communicate with distant nodes that are also inside the mesh and use the same protocol. That is, the protocol allows the creation of an isolated network, a bubble with no outside communication. Creating an isolated network is an accomplishment in itself; however, a scenario may arise where a client node desires to communicate with a server or another client located outside the mesh, for instance, on the internet. To allow a B.A.T.M.A.N mesh network to interact with other networks, there can be special nodes put in place in the network to provide additional functionality. These nodes can also function as an internet gateway, which is basically a bridge between two separate networks. To announce that a node is also a gateway, the node must declare this functionality when creating its OGMs and add the correct information in the following fields:

- **GWFlag:** Specifies the upload and download speeds in kbit per second;
- **GWPort:** Specifies the port number for tunnel communication with the gateway node;
- **HNA Extension Messages:** These are appended after the OGM and contain the (outside) network address and a netmask.

A client node can receive OGMs from multiple gateway nodes and decide which one they want to establish a connection with, based on connection quality or speed, or even a mixture of both parameters. It is common for the client to set his/her preferences manually, but an auto-selection mechanism is also available if needed.

For B.A.T.M.A.N generations that work on the layer 3 of the OSI model, a client node needs to encapsulate the Internet data in an UDP/IP datagram. This means that packets destined for the outside network must be wrapped into UDP packets when forwarding the data to the gateway node. As mentioned earlier, gateway node OGMs will specify which port number should be used for gateway functionality. Upon receiving a packet, the gateway node will read the port number from the outer UDP header and if the port number is correct, it will “decapsulate” the packet, keeping only the original IP packet to forward it to its final destination.

7. B.A.T.M.A.N Advanced

One can think of generations I to V of B.A.T.M.A.N as the abstract ideas, the blueprints of the protocol. B.A.T.M.A.N Deamon and Advanced are the actual implementations/programms that turn the idea into reality. B.A.T.M.A.N Deamon is the name of the older implementation of the protocol and operates on layer 3 of the OSI model. In contrast, B.A.T.M.A.N Advanced currently uses B.A.T.M.A.N IVbut also operates on layer 2. This means that instead of using IPs and packets, it encapsulates data and fowards them as raw Ethernet frames using MAC (Media Access Control) addresses to route them across the mesh until they reach their intended destination node. Since this implementation uses MAC addresses, one can make the comparison between the mesh network and a virtual switch. Each node can be interpreted as a port of this switch, so when non-mesh nodes connect to a mesh node, they get the illusion of having connected to a switch port, thus being in a local network. The underlying network topology invisible to them.

7.1. Data Forwarding in B.A.T.M.A.N Advanced

B.A.T.M.A.N Advanced excels at using different network interface types to foward data. Based on the link qualities defined by the algorithm B.A.T.M.A.N Advanced can choose which hard interface of a node is best suited for communication (e.g. Wi-Fi or Ethernet) in order to ensure the best path. This also means another interface outside the mesh can be bridged to a mesh mode with the bat0 interface which will then seamlessly foward the data. B.A.T.M.A.N Advanced uses raw Ethernet frames so it is not possible to just send that type of data over Wi-Fi because Wi-Fi and Ethernet have different formats and protocols. Here is an example of what a communication through a Wi-Fi connection:

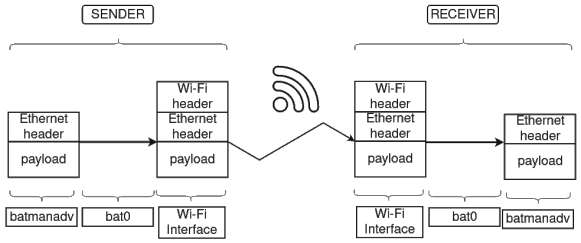

Figure 3: B.A.T.M.A.N Advanced encapsulation process

In the figure above, node A is the sender and B the receiver. B.A.T.M.A.N Advanced in node A will craft the raw Ethernet frame and then the Wi-Fi driver will encapsulate the data into a Wi-Fi frame. The data will then be sent from A’s Wi-Fi card to B’s. The Wi-Fi interface will receive the data, the driver will decapsulate it and deliver it to the bat0 interface. After that the B.A.T.M.A.N Advanced protocol from B will take over and process the data in its desired format.

7.2. Distributed ARP Tables

A mesh network using B.A.T.M.A.N Advanced can be used as an intermediary that connects non-mesh

clients. The problem is that these clients will only know each other's IP address, but not their MAC addresses, which are the key data necessary for communication on B.A.T.M.A.N Advanced networks.

In the traditional case, when a non-mesh client linked to mesh node A wants to communicate with one linked to mesh node B, it send an ARP request to A. A broadcasts the request until it eventually reaches B, which would respond with the MAC, or else the packets are lost, which will eventually result in A having to wait for a timeout before requesting again.

The advantage of using a distributed ARP Table [6] is that after the initial misses, a cache memory of IP to MAC entries is stored in the nodes. This means that groups of nodes are able to cache subsets of entries (IP & MAC) of the non-mesh clients they are linked to. Thanks to a distributed hash fuction, even if a node does not have the needed entry, it will know exacly where the entry could be found. Given an IP address to any node, it will apply the hash function and forward the request to a group of nodes where the key-value pair (MAC,IP) is stored. This allows unicast of the ARP requests, thus the likelihood of packet losses is significantly lower.

8. Conclusion

In conclusion, the B.A.T.M.A.N protocol and its current implementation in B.A.T.M.A.N Advanced as a Linux Kernel driver is a viable option for networks where a fixed infrastructure is not trusted, too expensive or unreliable such as in military operations scenarios or in areas of natural disasters. The protocol has good application in these areas because the protocol is suited for unreliable nodes that can unexpectedly go offline since the protocol will automatically find a substitute route from point A to B without severing the connection, something that makes it an invaluable tool for creating self-healing networks. One of the most relevant examples of its practical and successful application is the Freifunk Community in Germany, which has been using the B.A.T.M.A.N protocol for its mesh networks throughout many German cities in order to provide free Wi-Fi, demonstrating the protocol's potential to facilitate community-driven, decentralized networking solutions on a larger scale.

References

[1] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better approach to mobile ad-hoc networking (b.a.t.m.a.n.)," Internet Engineering Task Force, Internet-Draft draft-openmesh-b-a-t-m-a-n-00, Mar. 2008, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-openmesh-b-a-t-m-a-n-00.txt

[2] A. Neumann, C. E. Aichele, and M. Lindner, "B.a.t.m.a.n. status report," Tech. Rep., Jun. 2007. [Online]. Available: https://downloads.open-mesh.org/batman/papers/batman-status.pdf

[3] "Batman-iv," https://www.open-mesh.org/projects/batman-adv/wiki/BATMAN_IV, 2024, accessed: 26.03.2024.

[4] A. Quartulli and R. Lo Cigno, "Client announcement and fast roaming in a layer-2 mesh network," *Technical Report DISI-11-472, Università degli Studi di Brescia*, October 2011, version 1.0. [Online]. Available: https://www.researchgate.net/publication/265010299_Client_announcement_and_Fast_roaming_in_a_Layer-2_mesh_network

[5] "Batman-v," https://www.open-mesh.org/projects/batman-adv/wiki/BATMAN_V, 2024, accessed: 26.03.2024.

[6] "Distributed ARP Tables," https://www.open-mesh.org/projects/batman-adv/wiki/DistributedARPTable, 2024, accessed: 26.03.2024.