Dynamic Mesh Network for Telemetry Propagation and Communications in Coordinated Drone Swarms

Eric A. Cai
Donald Bren School of Information and Computer Science (*Computer Science and Engineering*)
University of California, Irvine
Irvine, CA , USA

Davis K. Furukawa
Donald Bren School of Information and Computer Science (*Computer Science and Engineering*)
University of California, Irvine
Irvine, CA , USA

Dylan C. Leighton
Donald Bren School of Information and Computer Science (*Computer Science and Engineering*)
University of California, Irvine
Irvine, CA , USA

Gustavo A. Velazquez
Henry Samueli School of Engineering (*Computer Science and Engineering*)
University of California, Irvine
Irvine, CA , USA

Haowei Zhang
Henry Samueli School of Engineering (*Electrical Engineering*)
University of California, Irvine
Irvine, CA , USA

Davide Callegaro
Donald Bren School of Information and Computer Science (*Graduate Student (Ph. D)*)
University of California, Irvine
Irvine, CA , USA

Marco Levorato
Donald Bren School of Information and Computer Science (*Professor*)
University of California, Irvine
Irvine, CA , USA

***Abstract*—Unmanned aerial vehicles (UAV) are emerging as a new technology for innovative applications. A dynamic mesh or dynamic ad-hoc network is one interesting network model that is applicable to the high mobility of nodes within these UAV swarms. *BATMAN advanced* was chosen from multiple routing protocols as its flexibility and portability creates the perfect foundation for our own more application specific protocol. *BATMAN adv* allows for any transport layer protocol to be implemented on top of it. Our project compares the performance of TCP and Reliable UDP (RUDP) as the transport layer protocol for the network. Our project aims to create a dynamic mesh and implement a protocol that is application specific to data transfer in networks with high mobility nodes.**

The current design contains 5 Raspberry Pi's that are configured with the BATMAN adv protocol. We have conducted a Static Node Test that is without interference and Aerial Interference Test with simulated aerial interference by gently shaking the Raspberry Pi's. The results of the test show that the aerial interference tests for bandwidth is better than the tests with no interference. The general range of interest for distances and their respective bandwidths is 0 to 60 meters. However, the maximum effective one hop range of BATMAN adv. is approximately 130 meters. Our network has been proven to support multi-hop data transfer and has been tested on implementations of TCP and RUDP. It is clear from the data that multi-hop routes are better than 1-hop routes in terms of RTT for image transfer. It also seems that RUDP is slightly better than TCP for RTT of image transfers in a 2-hop route.

***Index Terms*—Mesh Network, Dynamic Mesh Network, BATMAN, TCP, UAV, Drones, Raspberry Pi, Wireless, Wi-Fi, Bandwidth, Iperf, Reliable UDP (RUDP), High Mobility, Real-Time, Grafana, InfluxDB**

I. Introduction

Drones or unmanned aerial vehicles (UAVs) are becoming more commonplace in today's technologically integrated society. Their uses range from search and rescue in natural disasters, law enforcement, firefighting, mapping, agriculture, etc. However, as the use of drones becomes more prevalent, there is an increasing demand for more efficient and effective communications between these aerial devices. To satisfy the necessary specifications of a robust communication network, research is required for various topological solutions. One specific solution to approach this issue is the use of a mesh network to establish communication between the devices.

A mesh network is a local network topology in which its nodes are connected directly to one another. This interconnectedness avoids the dependency on a central or critical link like that present within a STAR topology. In any communication network, delay and congestion will always be present which facilitates the need for proper routing algorithms to ensure the effective and timely transfer of data. Depending on the numerous scenarios, making use of the proper routing algorithm is essential for the functionality of the mesh network.

Due to the fact that drones are highly mobile devices, the need for a mesh network that compensates for this phenomenon is necessary. Thus, a dynamic mesh network or a dynamic ad-hoc network is the most appropriate network topology for this situation. However, high node mobility

causes the configuration of the network topology to constantly change. This constant change of configuration increases the difficulty for routing and data transfer as previously used routes become meaningless. Thus, there was a need for constantly updating proactive routing protocol to handle the bulk of the routing within the dynamic mesh. Better Approach To Mobile Ad-hoc Network advanced (BATMAN adv) is one proactive routing protocol that is commonly used as a basis for routing and data transfer within a mesh network. BATMAN adv also only connects the devices at the device level, allow for the implementation of any transfer protocol on top of it to optimize the performance of the network. Due to the issues presented by high node mobility, such as lost connections, the implementation of real-time protocols, such as Reliable UDP (RUDP) was considered and compared to an implementation of a regular transfer protocol such as TCP.

This project aims to ensure productive communications and data transfer between UAV's and/or to other devices outside the associated wireless ad hoc mesh network, such as a computer. This is will be built upon inspiration from existing protocols of network configuration, such as B.A.T.M.A.N (Better Approach to Mobile Ad-Hoc Networks) Advanced, to create a configuration that is application specific to high mobility nodes, such as drones. One particular prior example of work with mesh networks was an experiment done by Benjamin Sliwa et al. [1] on the use of multiple relay nodes, a server node, and a mobile node and running routing tests with various routing protocols. This experiment used a vehicular ad-hoc network (VANET) and tests the performance of BATMAN V routing in comparison to other routing algorithms in this configuration. This experiment showed that BATMAN V was comparatively better than its counterparts in tests for packet delivery against speed, traffic load, and number of parallel bit streams. Our project aims to allow all the nodes to mobile, but the performance and experimentation on a single mobile node in the mesh network is a perfect foundation for our own experiments and design.

In this paper we will give:

1) A background on:
   - Wireless Mesh Networks (WMN)
   - Ad-hoc on demand Distance Vector (AODV)
   - Optimal Link State Routing
   - BABEL
   - BATMAN
   - TCP
   - UDP
   - QUIC
   - RUDP
2) The Project Design including:
   - Materials
   - Standards
   - Current Design
3) The Experimental Design including:
   - Testing
   - Metrics Used
   - Data
   - Results
4) Summary and Conclusion

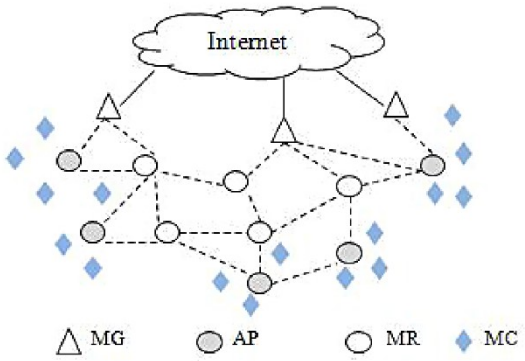

Fig. 1. Topological design of a mesh network (MG: Mesh Gateway, AP: Access Point, MR: Mesh Router, MC: Mesh Client) [3]

II. Background Information

A. Wireless Mesh Networks/Dynamic Mesh Networks:

Wireless Mesh Networks are a form of wireless ad-hoc networks which means that the network is not centralized in nature. Mesh networks tend to be strongly connected, which means that all nodes within the network are connected to each other in one hop. However, this is not a necessity as long as all nodes can reach every other node through some path and there is not a centralized node that connects to every node, like that of a star topology. Another important component is that mesh networks are self-healing and self-configuring which allows them to be much more fault tolerant [2]. The self-configuring and self-healing qualities allow for the nodes in the network to remain connected when a node within the mesh is disconnected or disappears from the network. Specially configured nodes that are called gateways and bridges are needed to facilitate communications between devices that are outside the mesh with devices that are in the network. A gateway is connected to some network, i.e. a local area network (LAN), and allows any device that is connected to the LAN to communicate with devices that are in the mesh. A bridge allows devices, that are outside the mesh network and not connected to networks that the gateways of the mesh are connected, to communicate with devices within the mesh network [3].

There are three main model for mesh networks: the infrastructure model, client model, and the hybrid model which is a combination of the infrastructure model and client model.

Infrastructure WMNs: This type of WMN forms a backbone of mesh routers that create a mesh of self-configuring, self-healing links that can be connected to the internet and other networks through routers configured as gateways or bridges. This allows clients outside the mesh network to communicate with the devices that are present within the

router but only through the specific gateway or bridge routers. This is the most common type of mesh network that is used. Fig. 3 is a visual representation of this WMN model [2].

Client WMNs: This type of WMN provides peer-to-peer (P2P) networking among the devices within the mesh network. As the client nodes facilitate the routing and configuration functionalities there is no requirement for mesh routers in this model. Packets that are transferred within this network tend to travel in multi-hop paths through multiple nodes within the network. Since Client WMNs provide a P2P network, the devices commonly use one type of radio to form the network. However, since there are no designated mesh routers in this network the end nodes are responsible for calculations for routing and configuration which in turn increases the requirements of these end nodes as compared to an infrastructure model. Fig. 4 is a visual representation of the client model [2].

Hybrid WMNs: This type of WMN is a combination of the infrastructure and client WMNs. Hybrid WMNs allow devices outside the mesh network to interact with devices inside the network by accessing a gateway or bridge node. However, other devices that are connected to a client that is connected to the mesh network can also use the client as a gateway and access devices in the network. The infrastructure of the hybrid model allows for connections to other networks such as Internet or Wi-Fi. Interestingly, the addition of clients and their routing capabilities provides improved connectivity and coverage within the network. Fig. 5 is a visual representation of the hybrid model[2].

Characteristics:

- **Efficiency:** WMNs have very minimal needs in terms of physical infrastructure. This allows for WMNs to be perfect for applications in developing countries or during natural disasters as a network created with minimal physical infrastructure can still be deployed or used after physical damage is done to most other infrastructure. The movement or changing of nodes and routing protocol that are implemented within the mesh network affect the overhead within the mesh network and can affect the efficiency of the network as a whole. How one approaches the splitting and transmission of data will also affect the efficiency of the network. For example, transmission over too many hops can lead to large delays and decreased throughput.
- **Limits of Usability:** The nodes within the mesh network and the protocols implemented are the main sources of limitations for a mesh network. The capacity or bandwidth capabilities and the memory that is available in each node is a limiting factor. These factors are fairly obvious as a node with more memory and better bandwidth is most likely better than one with less memory and bandwidth. Another limiting factor is that the mesh network needs a minimum number of nodes to function. This implies that under certain circumstances, it is not a viable solution if one does not have the requisite number of devices. Protocols and other algorithms implemented will also cause limitation due to inherent need for calculations and storage.
- **Fault Tolerance:** Since mesh networks are decentralized networks, there is not single point of failure within the mesh network. Some mesh networks use techniques such as flooding, broadcasting packets/information to every

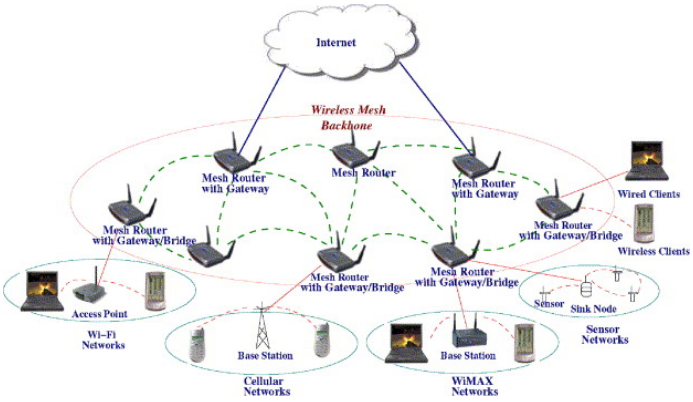

Fig. 3. Representation of Infrastructure Mesh[2]

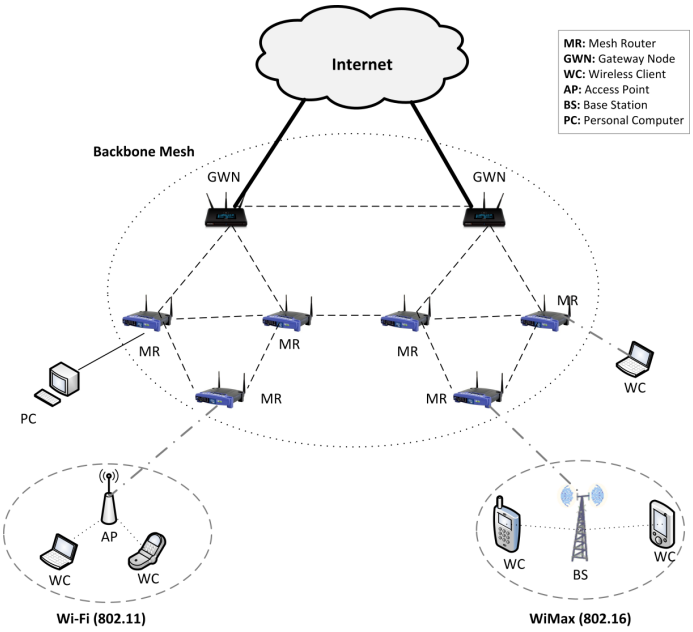

Fig. 2. More Visual Representation of Mesh Network

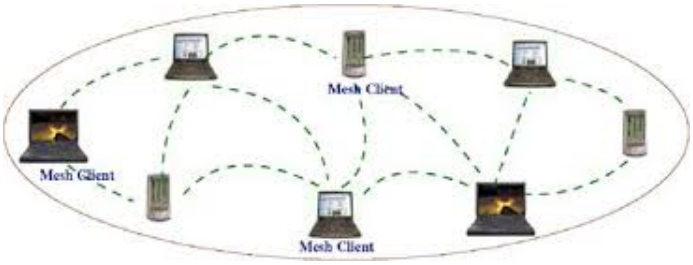

Fig. 4. Representation of Client Mesh[2]

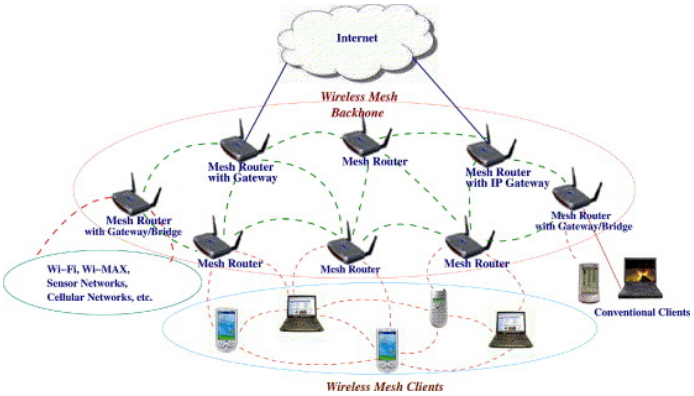

Fig. 5. Representation of Hybrid Mesh[2]

single node within the network, to create redundancy and prevent loss. However, this does not mean that there is not point of failure that can occur. Within infrastructure and hybrid models, if there is a single gateway used, the gateway can be seen as a single point of failure. This is because it is the only link between outside devices and devices within the mesh causing it to be critical. Another interesting characteristic is that the mesh network is highly dependent on routing. This means that the efficiency and effectiveness of the network is dependent on the ability of the routing protocol to handle sudden issues such as a disconnected node.

B. Routing Protocols

There are many approaches to routing within networks which leads to a variety of solutions and methods. In this paper, we are concerned with the difference in performance between common reactive, algorithms that calculate routes after a request for transfer has been made, and proactive, algorithms that calculate routing table before requests, protocols in mesh networks and their usability and effectiveness in a network with high node mobility. With this in mind, we researched and compared the reactive solution AODV and three proactive solutions OSLR, BABEL and BATMAN.

Reactive: Reactive protocols wait until a request is sent before allowing the nodes to initiate a route discovery process. This means that the nodes do not need to constantly update their routing tables to predict the best routes within the network like proactive protocols do. This causes reactive protocols to have higher latency but lower overhead when compared to proactive protocols.

- **AODV:** A reactive protocol that discovers routes as necessary and does not maintain routes from every node to every other[5]. These routes are maintained for a certain period of time, if it is not used again it is considered expired and discarded. This minimizes the number of active routes between source and destination as well as stale routes. This helps with reducing the computations needed for route maintenance in the protocol. Routes are discovered in the network by flooding the network with a route request packet. The nodes maintain a list of precursor nodes for each destination, that should be routed through [4].

Proactive: Proactive protocols use various techniques to maintain necessary routing tables on every node within the mesh to represent the entire topology. This means that the route between nodes can be given immediately when requested but incurs higher overhead as the tables must be stored and maintained. The maintenance of the tables also forces the nodes within the mesh to need to constantly communicate to ensure that the topology of the network has not changed or check if there is a better path.

- **OSLR:** OLSR was developed for MANETs and is a table-driven proactive protocol[5]. The nodes select a set of neighbors to be multipoint relay (MPR). These MPRs are responsible for forwarding control traffic and are an efficient mechanism for flooding control traffic by reducing the number of required transmissions. OLSR provides the shortest path routes by using MPR nodes and their declarations of link-state information for their MPR selectors. MPRs periodically announce their information in control messages and the route is calculated by finding the nodes that are reachable from the various MPRs [6],[8].
- **BABEL:** Proactive routing based on distance vector approach to routing that is an evolution of the Expected Transmission count ETX algorithm [7]. This selects routes more intelligently than using single hop-count approach and has two distinctive characteristics that optimize relay performance. BABEL uses history-sensitive route selection to ensure that nodes do not continuously change preferred route between source and destination as it can lead to route instability. BABEL will prefer past routes over routes that have just been established when there is a choice. BABEL also executes a reactive update and forces a request for routing when a link failure is detected in the network [8].
- **BATMAN:** Proactive routing protocol that is more aligned to the minimal resources available in embedded hardware [8]. There is no central knowledge of routing information such that no node knows the topology of the network. They also do not know the routes to all other destinations, but only have a list of nearest neighbors as well as their best neighbor. Information is routed by forwarding through nodes and their subsequent best neighbor. BATMAN also prefers better links on the idea that better links provide faster and more reliable communication [12]. The best neighbor of a node is determined by how well the link is between the node and that neighbor. The protocol seems to be less complex than link-state calculation and has modest hardware requirements.

There are various factors that must be taken into account when deciding the type of solution to use for routing. For example, mesh networks with high number of nodes may find

that a reactive solution is better than a proactive solution as there is a high amount of overhead in proactive routing tables when node counts are higher. For our project, we have a relatively low node count, between 5-10 nodes which is favors a proactive approach. Since BATMAN is built upon the logic of an OSLR approach and has many configurations that can be modified, we decided that BATMAN would be the best option to base our solution on.

C. B.A.T.M.A.N.

BATMAN was born out of a response to shortcomings that OLSR presented [9],[13]. As the number of nodes increased in the network, OLSR had a tendency to flush routing tables unnecessarily causing routes to regularly go up and down and routing loops due to out of sync routing tables. BATMAN combats these issues with the use of Originator Messages (OGM) that are flooded throughout the network to notify other nodes within the mesh of the existence of a node and to also test the strength of the links. The OGMs are used in BATMAN's calculations of the best neighbor for each node [9],[10],[11]. The Transmission Quality (TQ) of a link is based on the number of received OGMs from different nodes to get the receive quality (RQ) and the number of their own OGMs that are resent by their neighbors or the echo quality (EQ). EQ is divided by RQ to get TQ and the higher the TQ the better quality the link is[10]. This allows the nodes within the mesh to know which link is the best and therefore know to which node it needs to propagate data when needed. The nodes within a mesh configured with BATMAN use TQ to find its best neighbor and maintains a routing table of these best neighbors and nearest neighbors. This means that none of the nodes within a BATMAN mesh know the topology of the network but can still propagate data in the correct direction. BATMAN also considers number of hops in the calculations by removing a set percentage for every hop from the TQ of a particular node. For example, every extra hop besides the first could incur a four percent (4%) penalty to the TQ. This helps prevent routes that are stronger in strength but have significantly larger hop counts [11],[12].

BATMAN was originally a layer routing protocol and used UDP to handle the exchange of routing information. However, BATMAN adv is a layer 2 routing protocol and increased efficiency by pushing most of the overhead calculations into the Linux kernel. This shift moved the routing protocol from the network layer to the data link layer. This change caused a shift from the use of IP addresses to MAC addresses and allows the protocol to be easily portable between IPv4 and IPv6. This feature makes the protocol more lightweight and futureproof. Another advantage of moving routing to the data link layer is that any desired transfer protocol such as TCP or UDP can be implemented on top of the current BATMAN routing protocol, increasing the versatility of the protocol [11],[12].

BATMAN adv has default settings that run the devices within the network on a single frequency channel and offers TDMA as the solution to communication. Based on the 802.11 standard, the frequency channels that would be used are 1, 6 or 11. However, BATMAN adv also allows for changes in the default configuration to allow multicast and the use of multiple channels instead of the default broadcast and single channel approach [11],[12].

BATMAN adv is a routing protocol that is extremely versatile in nature and allows for the tweaking of configurations to fit the needs of the user. This makes BATMAN adv the perfect foundation for our own application specific protocol to ensure data transmission in high mobility mesh networks.

D. Transfer Protocols Analysis

- **TCP:** TCP (Transmission Control Protocol) is one type of transport layer protocol that has flow control, congestion control, reordering of packets, and retransmission. The problem with TCP is that it is slow and not "real-time", that is, it assumes a connection for the duration of the transmission/ACK time which is not guaranteed in our use case. We want the reliability part of TCP without the slowness of TCP which is why we are looking for a new option.
- **UDP:** UDP (User Datagram Protocol) is another transport layer protocol, however, it has no reliability or guarantees for packet delivery. It is however, fast and "real time" in that it will not try to keep retransmitting a packet or packets if the connection is lost. We want the speed of UDP and the reduced redundancy when the connection is lost that packets do not continue getting (re)transmitted and instead the node flushes the transmission buffer and move on.
- **QUIC:** QUIC is a google implementation of a protocol on top of UDP that controls flow, congestion, and security [18]. The goal is to reduce the amount of handshake that goes on when creating a connection, more specifically an http connection. The main point of this being that QUIC is made as a "security layer protocol" on top of an http connection. More can be read in the report above. Another major advantage of QUIC is that it uses multiple streams to transmit requests and responses on the connection allowing for parallel request/response streams. A major stopping point with QUIC that we ran into is that it is heavily dependent on current http protocol standards and is being developed and documents for use with http connections which are not what we are doing. In addition to this, QUIC is so new there is very little documentation on it so recreating it for our use is also extremely difficult.
- **RUDP:** Reliable User Datagram Protocol is a lightweight protocol in the application layer (on top of transport protocol UDP) that has some aspects of the TCP reliability, but quicker and better for "real-time" uses. RUDP allows for flow/congestion control and retransmissions and possibly out of order packets and since it is a protocol that exists (in our case) on the application layer we can manipulate the parameters like timeout time and number of retransmission attempts. This is important because we want to find an optimal number of retransmission

attempts and timeout for our ad-hoc network use case. Problems that have arisen are that there is not too much research papers or standardized documentation on RUDP as it is not a standardized protocol. We are currently looking further into this protocol as it has promise in that we can control the parameters. Our current implementation is in Python (slow) so our next step, assuming a go ahead by our advisor, will be to translate our Python implementation into C++. We are looking for other options as well as more studies on RUDP that discuss its performance.

Fig. 6. Raspberry Pi Model 3b

E. Reliable UDP (RUDP)

The Reliable User Datagram Protocol (RUDP) was proposed in an Internet Draft in 1999 as a transport protocol to provide the reliability offered in TCP while providing the "real-time" feature UDP provides. The most notable, recent, and relevant implementation, as well as, demonstration of this protocol can be found in the MILCOM 2009 – 2009 IEEE Military Communications Conference [15] in which they clearly detail and document the steps and processes that were taken to properly implement and utilize this protocol in an airborne network. The outlined performance and simulation results demonstrate that "RUDP... can improve the application performance significantly over TCP, UDP...". Furthermore, given that our project is considered to be airborne and dealing with high-mobility nodes and expected to have a real-time protocol, RUDP would be a heavy contender for the backbone of our transport protocol.

Another paper [16] discusses their design and implementation of an "embedded" RUDP. For the purpose of their project (2011), RUDP was implemented in an embedded system. From their testing and application, their results show that "...[RUDP] is very effective for the high-speed transmission of data in embedded systems..." And while their paper does not go into detail beyond embedded systems, it is worth noting that their main objective was to implement a "reliable data transmission [UDP]-based protocol for reliable transmission."

The remainder of the two papers document the layout and functionality of RUDP. One being the actual 1999 IETF Internet Draft [14] and another [17] being an analysis and comparison of RUDP against the usual TCP/UDP protocols. Currently, our implementation closely follows that of the mentioned 1999 Internet Draft. However, changes to satisfy our project's requirements can be made, if needed.

III. Project Design

A. Materials

Hardware:

- Raspberry Pi 3b
- Microsd Cards (8, 32, 64 GB)
- HDMI Cables
- Ethernet Cable
- UAVs - Drones
- Router/Network Switch
- Computers

Software:

- Python 3.6.8
- BATMAN-adv (Better Approach to Mobile Ad-Hoc Network)

B. Standards

- IEEE 802.11: IEEE Standard for Information Technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 10: Mesh Networking
- Wi-Fi 2.4 GHz (IPv4)
- MicroSD
- USB 2.0 / HDMI 2.0
- TCP

C. Current Design

Nodes:

- 4 Raspberry Pi Model 3b
  - 1 Raspberry Pi Model 3b is the Gateway (GW)
- 1 Raspberry Pi Zero

The design of our project is a simulated UAV mesh network with Raspberry Pi's as the stand ins for real UAVs. We represent these UAVs with Raspberry Pi's in the current design.

Five Raspberry Pi's have been configured to function as nodes in our mobile Ad-Hoc network. To relay messages

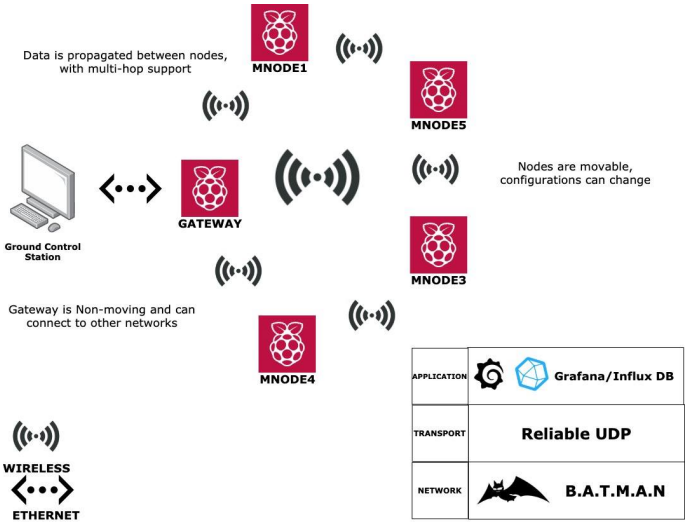

Fig. 7. Final Design of Network

and establish connection(s) between these nodes we utilized an open-source protocol known as B.A.T.M.A.N advanced to configure the Raspberry Pi's. One Raspberry Pi serves as the gateway that allows access between devices on the Wireless Network that the gateway is connected to via Ethernet and the nodes within the Mesh network. The remainder of the Raspberry Pi's behave as nodes to our Mobile Ad-Hoc Network. All these mesh nodes are connected to one another in the same fashion as the Mesh networks that were previously described. Figure 7 illustrates the final design of the mesh network that was implemented.

The Mesh Network that is currently in use is of the infrastructure model as we are trying to communicate between the mesh devices as well as allow an outside device to send commands. Fig. 8 shows an implementation of our mesh network with 3 nodes that was established for testing and proof of concept.

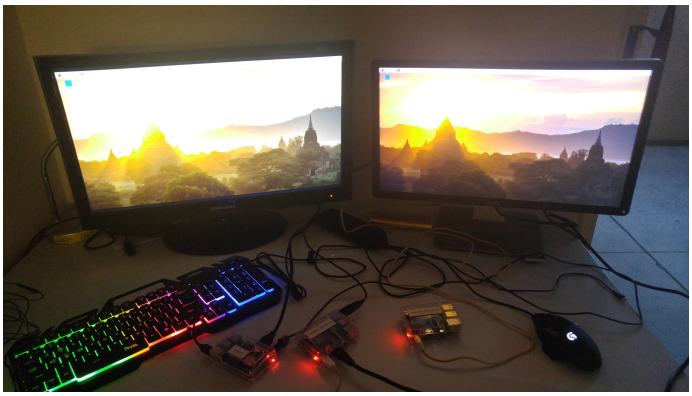

Fig. 8. Static Mesh Network Using 3 Nodes (Raspberry Pi)

IV. Experimental Design

A. Tests

1) **Static or no Interference Node Bandwidth Test:** The first test that was conducted was the using 2 nodes, one stationary the gateway and one mobile node. The gateway acted as the server and the mobile node was the client for the TCP Iperf test. The test was conducted by having a stationary GW and walking the mobile node away from the GW. An Iperf test was conducted at zero-distance from the gateway and in intervals of approximately ten steps. At every stop, a GPS coordinate was taken and an Iperf test was conducted. The test was conducted in a Line of Sight (LOS) manner and outside on an open field. The nodes were held above the ground to simulate drones off the ground and to decrease interference. These nodes were supposed to be static so the Raspberry Pi's were held as still as possible. Table 1 has the raw data for this test. Table 3 is the processed data for this test. Fig. 9 is a graphical representation of the processed data for this test. Fig 10 is a graphical representation of the processed data without the zero distance bandwidth to allow for more readability for the rest of the data in this first test.
2) **Aerial Interference Bandwidth Test:** The second test that was conducted used the same exact methods that were used in the first test. However, to simulate the interference that a drone would experience in flight we gently shook the Raspberry Pi's. Table 2 has the raw data for this test. Table 4 has the processed data for this test. Fig. 11 is a graphical representation of the processed data for this test.
3) **1 Hop RTT Tests:** This test was used to test the performance of our network when sending an image file over a 1 Hop connection. The image file was 58KB in size. Each test consisted of data collection at set intervals of 10 ft starting at 0 ft and ending at 80 ft. At each distance, there were five iterations of data collection where each iteration consisted of the image being sent 100 and getting RTTs for each image transfer for a total of 500 RTTs that were averaged to get the final data point. The test also was done using both TCP and RUDP as the transport layer protocol to allow for comparison and test which protocol was better for the network as a whole.
4) **2 Hop RTT Tests:** This test was used to test the performance of our network when sending an image file over a 2 Hop connection. The 2-Hop connection was achieved by having a node in the exact middle of the distances, for example if the distance was 10 ft between source and destination, the middle node is situated at 5 ft. Before doing any data collection for this test, multi-hopping was checked using the traceroute command that is native to BATMAN's control interface. The image file was 58KB in size. Each test consisted of data collection at set intervals

of 10 ft starting at 10 ft and ending at 80 ft. There is on 0 ft test because it is unnecessary to have multi-hop configurations at 0 ft. At each distance, there were five iterations of data collection where each iteration consisted of the image being sent 100 and getting RTTs for each image transfer for a total of 500 RTTs that were averaged to get the final data point. The test also was done using both TCP and RUDP as the transport layer protocol to allow for comparison and test which protocol was better for the network as a whole.

5) **1-Hop to 2-Hop Throughput Test:**
This test was conducted by having a middle node situated at 100 inches from the source node and taking throughput measurements at set distances separated by 33 inches. This test simulates the motion of a device that will decide to switch from a 1-hop connection to a 2-hop connection after passing the node that is at the set position. The throughput was taken at each distance using the throughput meter command that is native to the BATMAN control interface.

B. Metrics Used

1) **Distance:**
Distance in meters between two points; Calculated with a distance formula for two GPS coordinates. Distance calculated by gps-coordinates.org
In the second iteration of testing, the distances that were used were calculated using a tape measure that was of total length 25ft.
2) **Bandwidth:**
Kbits/sec Bandwidth report given by Iperf software tests
3) **Round Trip Time:**
This was calculated by subtracting the time right after sending the file from the time that the program on the sending node received the acknowledgment that the file was received from the receiving node. This eliminated the need for all devices in network to be time synchronized as all calculations were done based on the times from the same device.
4) **Throughput:**
This was given by the throughput meter that was present within the BATMAN control interface.
5) **Route:**
This was given by the traceroute command that was present within the BATMAN control interface.

C. Data

Table 1 has the raw data for the static node test.
Table 3 is the processed data for the static node test.
Table 2 has the raw data for the aerial node test.
Table 4 has the processed data for the aerial node test.
Other raw data can be found at [19].

D. Results

Fig. 9 is a graphical representation of the processed data for the static node test.

TABLE I
RAW DATA FOR STATIC NODE TESTS

Static Raw Data			
GPS Coordinates	Interval (s)	Size (MB)	Bandwidth (kbits/sec)
33.650119,-117.8340	0.0-10.2	23.9	19,700
33.650389,-117.8337	0.0-12.8	0.896	575
33.650543,-1178337	0.0-20.8	0.181	71.4
33.650670,-1178835	0.0-22.1	0.271	100
33.650663,-1178336	0.0-15.2	0.375	202
33.651008,-117.8333	0.0-390.2	0.0691	1.45

TABLE II
RAW DATA FOR AERIAL NODE TESTS

Aerial Raw Data			
GPS Coordinates	Duration (s)	Size (MB)	Bandwidth (kbits/sec)
33.650119,-117.8341	10.8	3.0	2320
33.650438,-117.8337	10.6	0.896	692
33.650319,-117.8337	11.0	1.380	1050
33.650565,-117.8337	10.9	2.880	2200
33.650657,-117.8336	10.5	0.640	502
33.650663,-117.8336	11.9	0.181	124
33.650872,-117.8334	608.7	0.102	1.38

Fig 10 is a graphical representation of the processed data without the zero distance bandwidth to allow for more readability for the rest of the data in the static node test.

Fig. 11 is a graphical representation of the processed data for the aerial node test.

Fig. 12 is a graphical representation of the processed results for 1-Hop TCP image transfer RTT.

Fig. 13 is a graphical representation of the processed results for 1-Hop RUDP image transfer RTT.

Fig. 14 is a graphical representation of the processed results for 2-Hop TCP image transfer RTT.

TABLE III
PROCESSED DATA FOR STATIC NODE TESTS

Static Processed Data	
Distance (m)	Bandwidth (kbits/sec)
0	19700
45.61	575
58.33	71.4
74.56	202
80.92	100
130.43	1.45

TABLE IV
PROCESSED DATA FOR AERIAL NODE TESTS

Aerial Processed Data	
Distance (m)	Bandwidth (kbits/sec)
0	2320
40.91	1050
49.37	692
66.03	2200
73.05	502
74.56	124
130.43	1.38

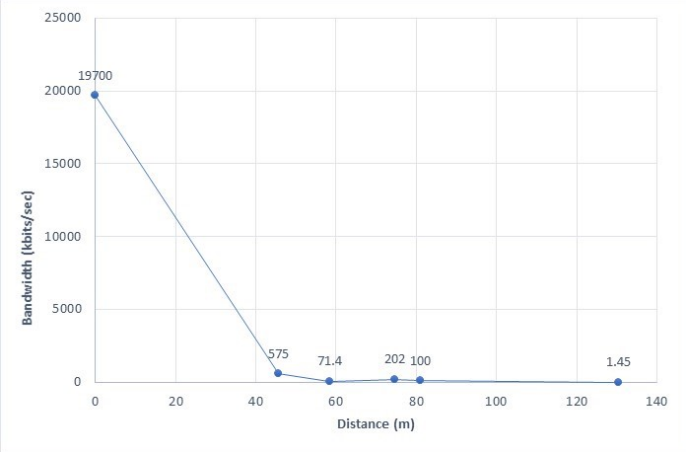

Fig. 9. Static Node Bandwidth Test

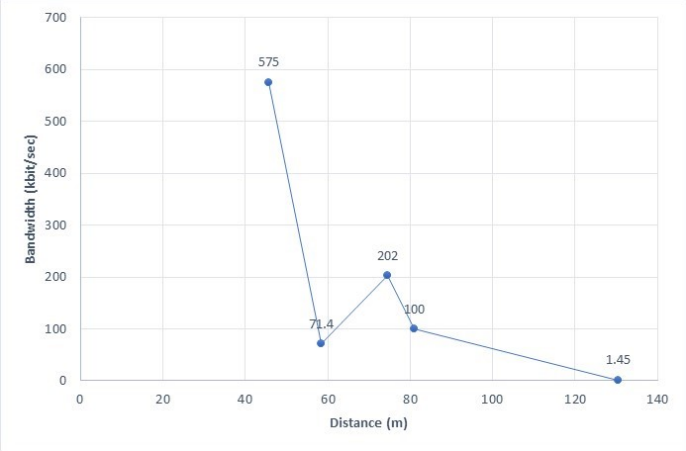

Fig. 10. Static Node Bandwidth Test without zero outlier to make the other data more readable.

Fig. 15 is a graphical representation of the processed results for 2-Hop RUDP image transfer RTT.

Fig. 16 is a graphical representation of the percent increase normalized to the first time measurement at 0 ft based on the processed results for 1-Hop TCP vs RUDP image transfer RTT.

Fig. 17 is a graphical representation of the percent increase normalized to the first time measurement at 10 ft based on the processed results for 2-Hop TCP vs RUDP image transfer RTT.

Fig. 18 is a graphical representation of the processed results for the 1-Hop to 2-Hop switch measured in Throughput.

V. Summary and Conclusion

Using the results from the first series of tests, we were able to identify that the range of interest for the effective distance seems to be between 0 and approximately 50 to 60 meters for both the Static Node Test and the Aerial Node Test. At the 130.43 meter mark, the results for the bandwidth were 1.45 kbits/sec and 1.38 kbits/sec for the static node test and aerial node test respectively. From this metric, we know that 130.43

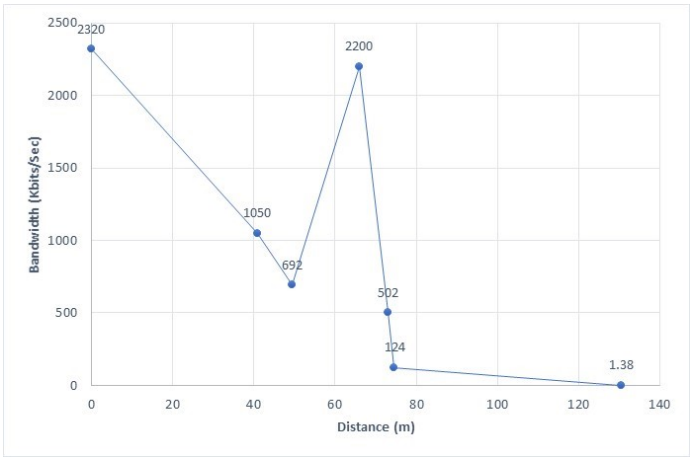

Fig. 11. Aerial Interference Node Bandwidth Tests

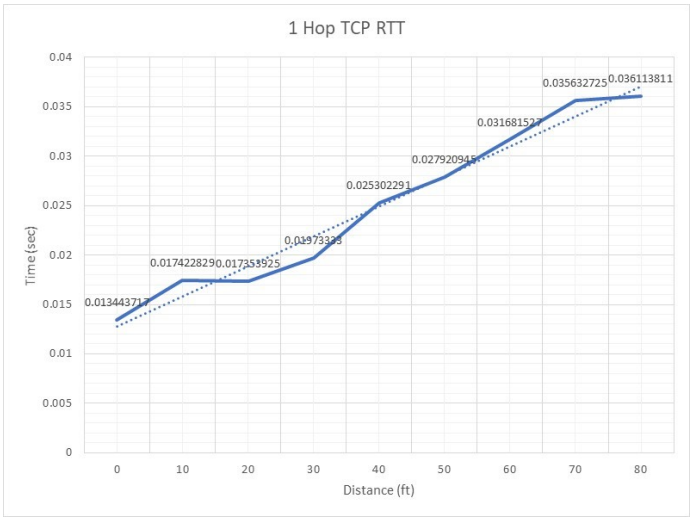

Fig. 12. 1 Hop TCP RTT VS Distance

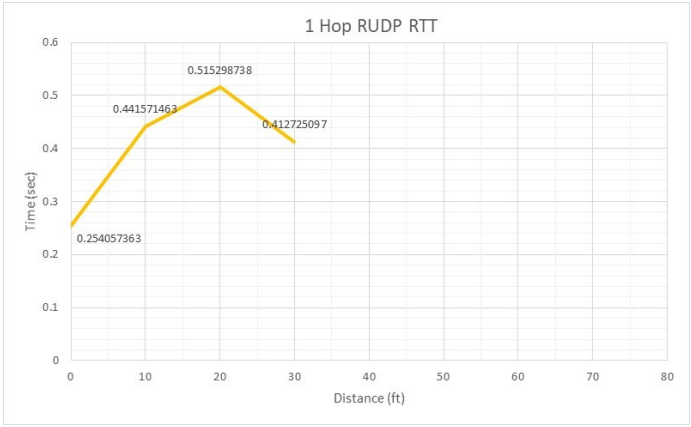

Fig. 13. 1 Hop RUDP RTT VS Distance

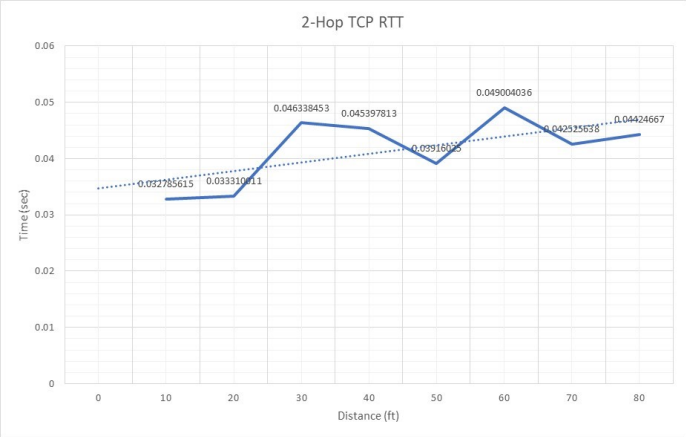

Fig. 14. 2 Hop TCP RTT VS Distance

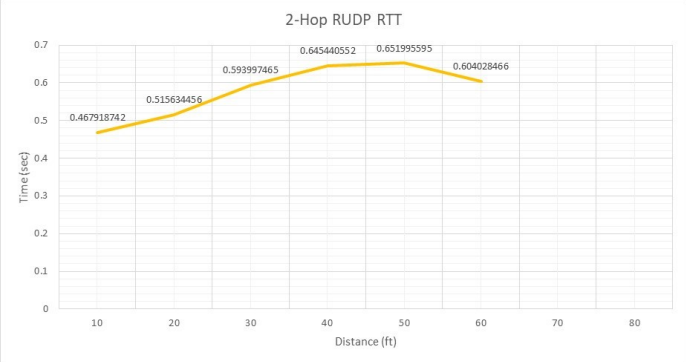

Fig. 15. 2 Hop RUDP RTT VS Distance

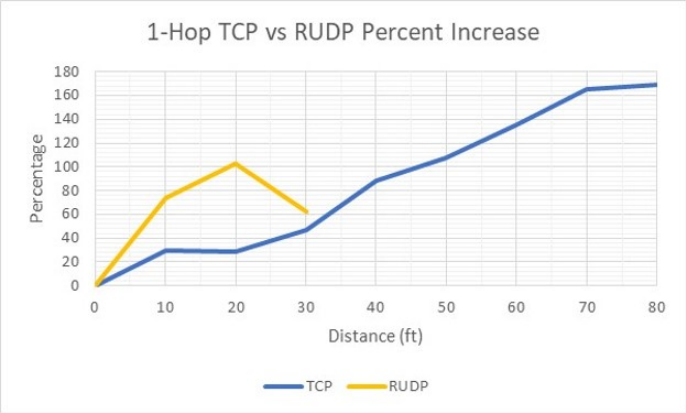

Fig. 16. 1 Hop TCP VS RUDP RTT VS Distance Percent Increase in Time

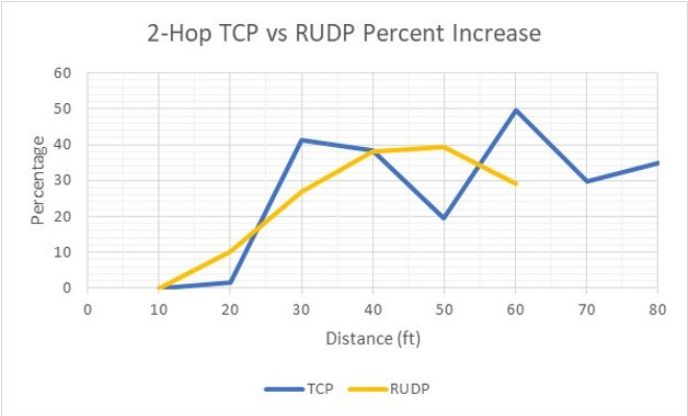

Fig. 17. 2 Hop TCP VS RUDP RTT VS Distance Percent Increase in Time

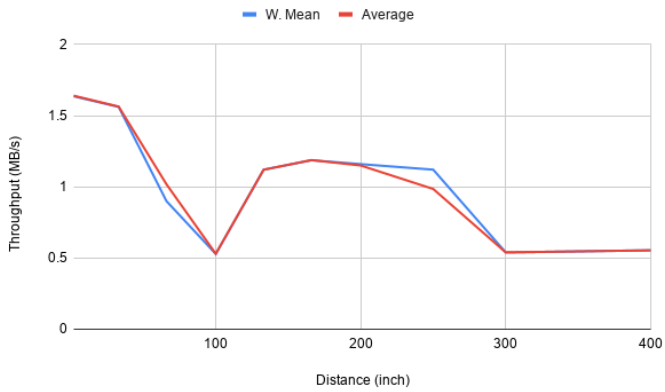

Fig. 18. 1 Hop to 2 Hop Throughput VS Distance

meters is still within the one-hop distance for the BATMAN adv routing protocol but it is not necessarily a useful distance. This helps us to conclude that approximately 130 meters is the maximum range for one hop routing for BATMAN adv implemented on Raspberry Pi's without supplemental enhancers like a WiFi Antenna or Adapter.

A throughput test was necessary to prove that our network was able to support multi-hop and allowed us to compare the performance of network when there are 1-hop connections and multi-hop connections. The test results for this test are shown in Figure 18, and the 'hump' in the graph shows that the network started multi-hopping after 100 inches from the source, which was the position where the middle node was placed. This proves that multi-hop is supported in our implementation of the network.

For the second iteration of the we had focused on using a maximum distance of 80 ft to help ensure that we would be able to collect usable data that was also meaningful. Looking at both the 1-hop and 2-hop tests, Figures 12, 13, 14, 15, it seems to show that TCP is always better than RUDP, our real-time protocol, at least in consideration of RTT as a metric. However, this is most likely due to the fact that our own implementation of RUDP is not the best. Processing the data

to show the percent increase in time over distance normalized to the first data point of each test shows that the difference between TCP and RUDP is much less than a pure time graph can show. This process also eliminates the problems that occur from issues in our implementation and allow for the protocols to be displayed based on their own performance. From Figures 16 and 17 it can be seen that the 2-hop tests have a much lower percent increase as compared to the 1-hop version of this testing. This implies that multi-hop routing is better than 1-hop routing for the RTT metric in our network. Another aspect of the multi-hop that is important to notice is that multi-hopping allowed for our implementation of RUDP to double the usable distance. This can be seen in Figures 13, 15. RUDP does not have any data for one hop past 30 ft because the number of timeouts that RUDP gave when transmitting hit our set limit and was determined to be a lost connection. This phenomenon did not occur in multi-hop routing until after the 60 ft distance as each individual route distance was 30 ft or below.

Looking at Figures 16 and 17 it is hard to conclude on which protocol is better for the performance of the network. The biggest limiting factor for this is the fact that RUDP was not able to produce data past 30 ft for 1-hop and 60ft for 2-hop because the connections were not stable enough. However, from a comparison of the data it seems that 2-hop is always better for both protocols and seems that RUDP may be slightly better than TCP in terms of percent increase in time for RTT. This is something that we expected as we believed that a real-time protocol should work better as it is more adaptable to weak and lost connections.

In summary, this paper gave background information on WMNs and BATMAN adv which is the foundation for the current project design. The experimental design led to two TCP Iperf tests, one for a static or no interference and one for aerial interference. Each of the tests were conducted in a field in an LOS manner and the results were interesting. The maximum one hop range within our network has been proved to be approximately 130 meters in distance and under conditions of little physical interference such as walls. We also proved that our network supported multi-hop data transfer routes and could successfully deliver that data. We then proceeded with a comparison of two different transfer level protocols TCP and RUDP to determine if standard TCP was better or a real-time solution was preferable. The results of our testing seems to imply that RUDP is slightly better than TCP and that multi-hop routes are always better than 1-hop routes.

ACKNOWLEDGMENT

This project is supported by the Henry Samueli School of Engineering and Donald Bren School of Information and Computer Sciences at the University of California, Irvine.

Special thanks to our Project Advisor, Associate Professor Marco Levorato, Davide Callegaro, and the HYDRA (Resilient Computation for Heterogeneous Autonomous Drone sYstems) Lab.

REFERENCES

[1] B. Sliwa, S. Falten, and C. Wietfeld, “Performance evaluation and optimization of B.A.T.M.A.N. v routing for aerial and ground-based mobile ad-hoc networks,” IEEE Veh. Technol. Conf., vol. 2019-April, pp. 1–7, 2019.

[2] I.F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: a survey, Computer Networks Journal 47 (4) (2005) 445–487

[3] Benyamina, D., Hafid, A., & Gendreau, M. (2011). Wireless mesh networks design—A survey. IEEE Communications surveys & tutorials, 14(2), 299-310.

[4] John Hopkins University. Ad hoc On Demand Distance Vector (AODV) Routing Protocol. [Online]. Available: https://www.cs.jhu.edu/ cs647/aodv.pdf

[5] K. Holter, “Comparing AODV and OLSR,” Comparing AODV and OLSR, 28-Apr-2005. [Online]. Available: https://folk.uio.no/kenneho/studies/essay/essay.html.

[6] Clausen, Thomas Heide & Jacquet, Philippe& Adjih, Cedric & Laouiti, Anis & Minet, Pascale & Muhlethaler, Paul & Qayyum, Amir & Viennot, Laurent. (2003). Optimized link state routing protocol (OLSR).

[7] V. Vesel´y, V. Rek, and O. Rysav´y, “Babel routing protocol for omnet++ more than just a new simulation module for INET framework,” CoRR, vol. abs/1609.05215, 2016.[Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[8] M. Abolhasan, B. Hagelstein and J. C. -. Wang, ”Real-world performance of current proactive multi-hop mesh protocols,” 2009 15th Asia-Pacific Conference on Communications, Shanghai, 2009, pp. 44-47. doi: 10.1109/APCC.2009.5375690

[9] D. Johnson, N. Ntlatlapa, C. Aichel, “Simple pragmatic approach to mesh routing using BATMAN,” 2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, Oct 2008, pp. 1–10

[10] D. Seither, A. Konig, M. Hollick, ”Routing performance of wireless mesh networks: A practical evaluation of BATMAN advanced”, Proc. IEEE 36th Conf. Local Comput. Netw., pp. 897-904, Oct. 2011.

[11] A. Neumann, C. Aichele, M. Lindner, S. Wunderlich, ”Better approach to mobile ad-hoc networking (B.A.T.M.A.N.)”, IETF Draft, 2008.

[12] Open-Mesh.net, “B.A.T.M.A.N. (better approach to mobile ad-hoc networking).” [Online]. Available: http://www.open-mesh.net/

[13] J. Pojda, A. Wolff, M. Sbeiti and C. Wietfeld, ”Performance analysis of mesh routing protocols for UAV swarming applications,” 2011 8th International Symposium on Wireless Communication Systems, Aachen, 2011, pp. 317-321. doi: 10.1109/ISWCS.2011.6125375

[14] T. Bova, T. Krivoruchka, ”Reliable UDP Protocol”, IETF Internet-Draft, February 1999.

[15] Tuong Le et al., ”Reliable User Datagram Protocol for airborne network,” MILCOM 2009 - 2009 IEEE Military Communications Conference, Boston, MA, 2009, pp. 1-6.

[16] J. Fang and M. Liu, ”Design and Implementation of Embedded RUDP,” 2011 Second International Conference on Networking and Distributed Computing, Beijing, 2011, pp. 7-9.

[17] A. O. F. Atya and Jilong Kuang, ”RUFC: A flexible framework for reliable UDP with flow control,” 8th International Conference for Internet Technology and Secured Transactions (ICITST-2013), London, 2013, pp. 276-281.

[18] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, Zhongyi Shi . 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In Proceedings of SIGCOMM ’17, Los Angeles, CA, USA, August 21-25, 2017, 14 pages. https://doi.org/10.1145/3098822.3098842

[19] Eric Cai, Davis Furukawa, Dylan Leighton, Gustavo Velazquez, Haowei Zhang. 2020. ”Dynamic-Network-Project-Documentation”. Github. https://github.com/ericcai9907/Dynamic-Network-Project-Documentation