Ad-Hoc Wireless Network Development and Performance on Raspberry Pi 3

Final Course Project - Internetworking of Things, Spring 2017

Christina Paolicelli

Introduction / Background

In this project Wi-Fi has been used to establish a Wireless Ad-Hoc Network among 3 nodes using Better Approach to Mobile Ad-Hoc Network-Advanced (BATMAN-adv). BATMAN-adv operates as a protocol on layer 2 which allows BATMAN-adv to handle data traffic, which is encapsulated and forwarded until it reaches its destination. BATMAN-adv divides best path knowledge among all nodes in the mesh by maintaining only information about best next hop towards all other nodes. In order to avoid routing loops an event-based flooding mechanism prevents contradicting topology information and avoid overhead of control traffic by limiting the amount of topology messages flooding the mesh.

Execution of such a flood in the BATMAN-adv implementation is described here. Each node broadcasts an originator message (OGM) to inform neighbors about its existence. The OGMs are then re-broadcast following specific rules to inform their neighbors and so on until the network is flooded. OGMs contain at minimum the address of the originator, the address of the transmitting node, a TTl and a sequence number. OGMs travel on paths of both good quality wireless links and poor quality wireless links, the packets which follow a poor quality wireless link, however, will experience a delay and decrease in reliability. The OGM's sequence number is used to determine if it has been received more than once, if not the node re-broadcasts if it was received from the neighbor which was identified as the next best ho towards the original initiator. In this way the OGMs are flooded selectively through the mesh and inform the receiving nodes about the originator's existence. The best next hop is determined by determining which of the originator messages the node received quicker and more reliably via a single hop neighbor, based on this the node configures the routing table respectively.

Node Dispersal

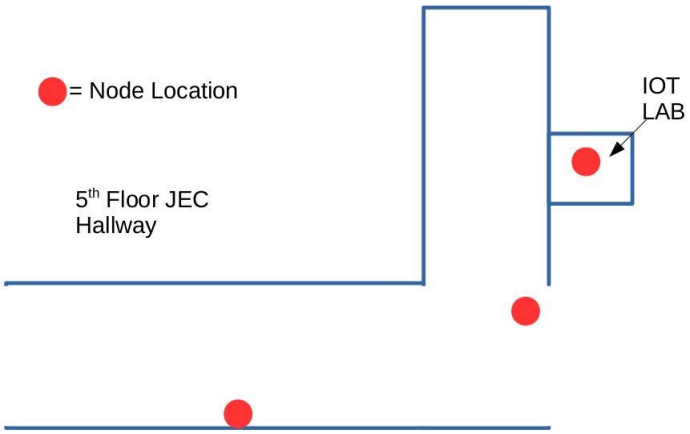


Figure 1: Node Configuration 1, end nodes outside direct connection range

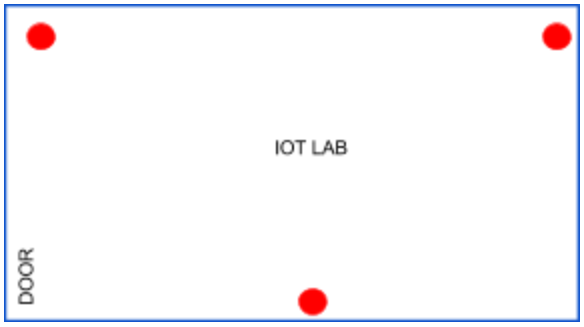


Figure 2: Node Configuration 1, all nodes can reach each other

Testing & Results

The testing methodology were developed based on methodologies presented in relevant literature, in particular [1] as many other paper's methodologies relied on Ipref which was not able to be implemented on this system.

Latency and Jitter

Each node in turn sent 100 pings to every other node. This was done to obtain the round trip delay times between pairs of nodes. A ping was used over broadcast packets because pings closely represent unicast messages which are common in wireless sensor networks. The packet interval time was sent to 0.01seconds for a sending rate of 1Mbps.

Jitter was calculated by taking the sum of the differences between the sample RTTs and dividing by the number of samples (minus 1). Jitter was considered tolerable if it is below 15% of the average latency. [2]

Results

Setup 1

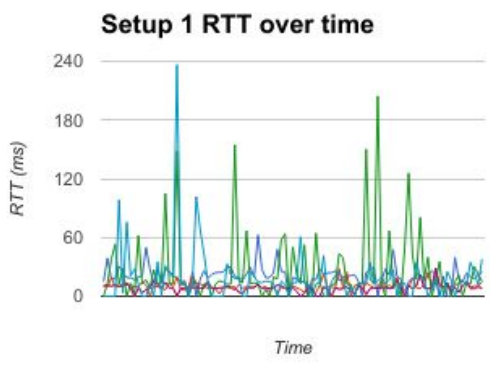


Figure 3: RTT values over time

In this configuration the two external nodes did not have a direct path of communication and therefore a two hop path had to be taken. For a two hop path it was found that the average RTT was 29.35 ms. This is compared to a single hop path between these nodes and the center nodes which had an average RTT 16.348 ms. Under this configuration and these conditions, therefore a two hop path has over twice the RTT as a single hop path. However in this configuration all of the RTTs were fairly low, it was found that the wireless of the raspberry pis did not have the best range and therefore the network was restricted to a single hallway. D. Koutsonikolas and C. Hu [1] suggest that an RTT below 100 is acceptable and this configuration meets this threshold for single hop links, however there were spikes in the RTT for the two hop link up to 236.5 ms.

Maximum jitter was found to be 33ms between the two external nodes. Minimum jitter was found to be 3.875 between the IOT lab node and the center node. All jitter in this configuration is above the 15% of latency threshold, which is not considered good, at the worst case scenario it is almost 127% of latency. Although these levels are quite high for simple data transfer it has negligible impact, however if the network was attempted to be used for video streaming or VOIP this would have more significant impacts.

Setup 2

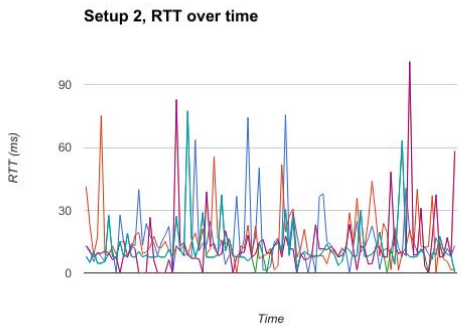


Figure 4: RTT over time for setup 2

In this configuration all nodes were within range of each other and therefore all paths are single hop. Average latency among all paths was 14 ms. In this setup there was a larger variation in RTTs, and there was one RTT above 100 ms

which is considered the acceptable threshold, however as this is a negligible amount in the scope of all messages sent this was disregarded.

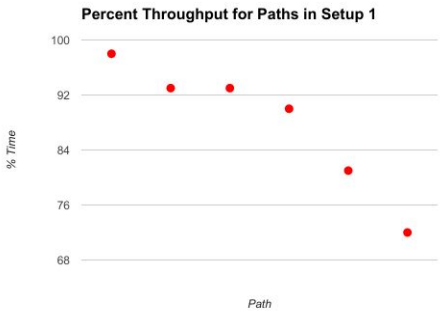
Despite having an overall better latency than the first configuration, configuration 2 has high levels of jitter, up to 75% of average latency. None of the path's met the 15% metric as the minimum jitter path was only 65% of average latency. Although these levels are quite high for simple data transfer it has negligible impact, however if the network was attempted to be used for video streaming or VOIP this would have more significant impacts. An additional note of this setup compared to the previous is that because this was done purely in the IOT Lab, on a busier day there was more possible interference than the performance in the hallway on a quiet day, the potential interference was not quantified so this is only a hypothesis which may explain the increased network jitter.

Loss and Throughput

To measure loss the same ping messages as latency were used and number of packets lost for 100 pings was recorded for each path.

Results

Setup 1



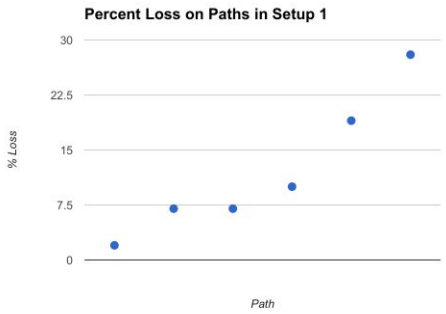


Figure 5: Percent Throughput and Loss for Setup 1

Loss rates range from only 2% all the way to 28%, the 28% loss was observed between the two edge nodes. The single hops, however, showed promising levels of throughput all displaying at least 90% throughput. While this has the potential to be improved it is noted that this setup was designed to place nodes at the the limits of range .

Setup 2

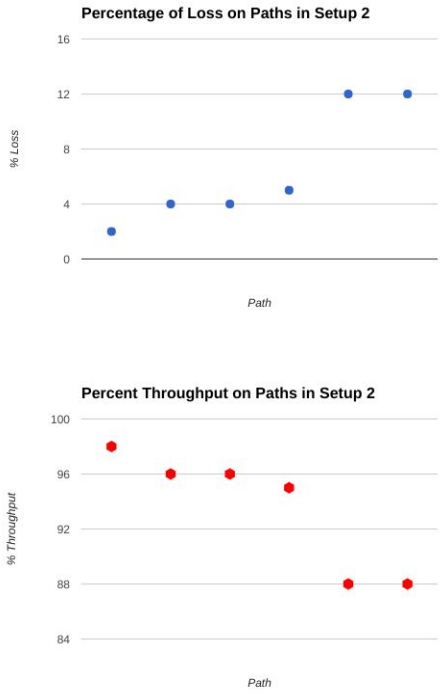


Figure 6: Loss and Throughput for Setup 2

Setup 2 was much more promising in terms of loss and throughput, however it was entirely comprised of single hops. The maximum loss was found at 12% and the minimum loss was 2%.

Additional Work

One hypothesis to improve network performance was that an increase in the interval time would reduce traffic on the network and reduce RTT. When this was implemented, however, it was found that there was no reduction in RTT and an increase in packet loss.

To determine if an application could run on top of the developed network a modified version of the lab 2 server/client code was written and run with success.

Conclusion

In this paper we describe an ad-hoc network developed using the BATMAN-adv protocol for deployment on raspberry pi 3s. Two different configurations were tested one which required the external nodes to communicate over two hops. The most notable observation was that single hop performance more reliably meets performance specifications than two hop performance. Despite this implementation and analysis whether this is a good way of implementing WANETs for raspberry pis, however this adds to the available data for different protocol implementations, especially on the raspberry pi platform for which there is lacking.

References

[1]S. Das, D. Koutsonikolas and C. Hu, "Measurement-based Characterization of a Wireless Mesh Network", 2017.
[2]"What is Jitter?", Pingman Tools, 2017. [Online]. Available: https://www.pingman.com/kb/article/what-is-jitter-57.html. [Accessed: 28- Apr- 2017].
[3]M. Sailash Singh and V. Talasila, "A Practical Evaluation for Routing Performance of BATMAN-ADV and HWMN in a Wireless Mesh Network Test-bed", International Conference on Smart Sensors and Systems, 2015.
[4] M. Lindner, S. Eckelmann, S. Wunderlich et al., "The B.A.T.M.A.N. Project." [Online]. Available: http://www.open-mesh.org/

Appendix

Mesh Configuration

Despite the large amount of information available on designing wireless mesh networks, and even particularly with this method this was still difficult to comprehend and accomplish for someone who is relatively new to networking. Because of the lack of beginner's information that I was able to find as I began this project I would like to provide a step-by-step guide to my network configuration for those who wish to try implementing such a network themselves.

1. Set up nodes with a linux OS (I used 3 Raspberry Pi 3s running Raspbian)
2. Connect nodes to internet to install necessary packages.
3. Install batctl

   sudo apt-get install batctl
4. Install batman-adv

   sudo apt-get install batctl bridge-utils

   sudo modprobe batman-adv

   batctl -v

   (The batctl and batman-adv versions should be displayed)
5. In /etc/network/interfaces modify the interfaces file to read the following

   auto lo
   iface lo inet loopback
   iface eth0 inet static

   auto wlan0
   iface wlan0 inet manual
   mtu 1532
   wireless-channel 1
   wireless-essid <network-name>
   wireless-mode ad-hoc
   wireless-ap: 02:12:34:56:78:9A

   auto bat0
   iface bat0 inet auto
   address 192.168.123.3
   netmask 255.255.255.0
   gateway 192.168.123.1
   pre-up /usr/sbin/batctl if add dev eth0
   pre-up /usr/sbin/batctl if add dev wlan0
6. Restart networking services (*sudo services networking restart)*
7. As root run the following commands to initialize network

   ip link set up dev wlan0

ip link set mtu 1532 dev wlan0
iwconfig wlan0 mode ad-hoc essid my-mesh-network ap 02:12:34:56:78:9A channel 1
batctl if add wlan0
ip link set up dev wlan0
ip link set up dev bat0

8. See if everything is working
   iwlist wlan0 scan