Cheap and Accessible Ad-hoc Mobile Mesh Networks

Adam Hultman, Aomi Jokoji, Isaac Donaldson

A paper presented for the class Advanced Networking

CSC 466

Computer Science

University of Victoria

Canada

April 11, 2022

Contents

1 Abstract **3**

2 Introduction **3**

3 Background **4**

4 Related Work **6**

5 Project Goals **7**

6 Setup Choices (routing, hardware, software) **8**

6.1 Setup Choices 8

6.1.1 Routing 8

6.1.2 Hardware 8

6.1.3 Software 9

7 Measurements **9**

8 Topology **9**

9 Demonstration **11**

10 Results and Analysis **12**

10.1 Qualitative Measurements 12

10.2 Quantitative Measurements 13

10.2.1 TP-Link Router to Raspberry Pi (Wireless) to EdgeRouterX (Wired) 14

10.2.2 Raspberry Pi to EdgeRouterX (Wired) 15

10.2.3 EdgeRouterX to Internet 16

10.2.4 TP-Link Router to Internet 17

10.2.5 Raspberry Pi to Internet 18

11 Discussion **18**

11.1 Hardware 19

11.2 Software 20

11.2.1 Bootstrapping 20

11.2.2 Babel 21

12 Future Work **21**

13 Conclusion **23**

References **24**

A Contributions **27**

A.1 Aomi Jokoji 27

A.2 Adam Hultman 27

A.3 Isaac Donaldson 28

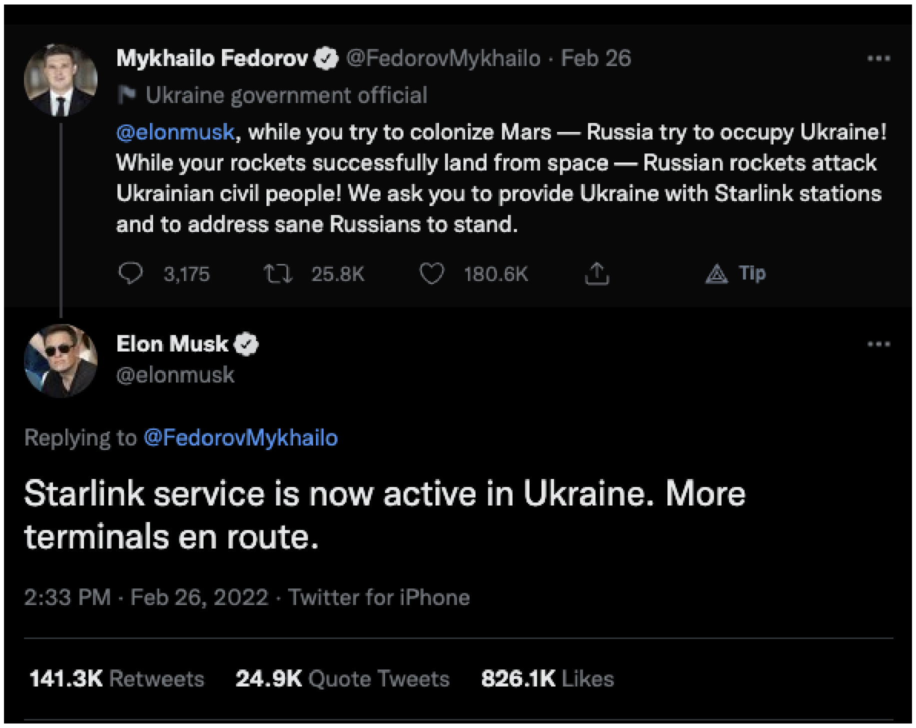
1 Abstract

The current solutions to supply remote, impoverished, or war-torn communities with internet are either expensive, sluggish, or unreliable. With new satellite internet options like Starlink and popularization of mesh networking in communities, these communities not have the possibility of viable internet access. We used commodity hardware and open source software to create an Ad-hoc Mobile Mesh Network where neighbors can share a single internet connection. We found the network to be capable of normal internet usage, including various communication methods. We found that the available material to help setup the network was lacking and documentation was difficult to find, so we propose some additions to the documentation to enable others to replicate our setup and make their own changes. We believe that networks like the one we created are a step towards having usable and affordable internet in communities previously unable to have the same access as those elsewhere in the world.

2 Introduction

Mesh networks have been around for a while now, and they are starting to become popular in home devices like the Google Nest product line. In addition, there are recent scenarios, like the war in the Ukraine, where creating a mesh network in conjunction with SpaceX Starlink satellites can be critical for supporting crucial communications [1]. Testing the ease of setup and usability is an important step to allow for useful internet communications in critical situations.

One of the main use cases that inspired this paper is the news from Russia's current invasion of Ukraine. A Ukrainian official reached out to Elon Musk on Twitter asking for Starlink to provide connection to the country, and Musk agreed [2]. Shortly after, Starlink terminals were sent to Ukraine and service was enabled for the country [Figure 1]. When we looked at how to achieve an Ad-hoc Mobile Mesh Network with a satellite back-haul connection using everyday equipment, we found the instructions and information to be outdated or lacking. We want to go through the process ourselves and report on the feasibility of this as a real world solution to enabling internet connection to communities using a single connection.



Mykhailo Fedorov @FedorovMykhailo · Feb 26
Ukraine government official
@elonmusk, while you try to colonize Mars — Russia try to occupy Ukraine! While your rockets successfully land from space — Russian rockets attack Ukrainian civil people! We ask you to provide Ukraine with Starlink stations and to address sane Russians to stand.
3,175 25.8K 180.6K Tip

Elon Musk
@elonmusk
Replying to @FedorovMykhailo
Starlink service is now active in Ukraine. More terminals en route.
2:33 PM · Feb 26, 2022 · Twitter for iPhone
141.3K Retweets 24.9K Quote Tweets 826.1K Likes

Figure 1: Elon Musk confirming delivery of Starlink terminals to the Ukraine [2]

3 Background

Mesh network WiFi devices have been made popular to consumers in the last couple of years thanks to Google Nest and Google Home suite of home devices, as well as many other home WiFi network solutions that use mesh networking. The benefits of these mesh networks are that they can extend the connectivity in a house to areas that are out of reach from the routers access point, and they can use a single network instead of having multiple access point as was required in the past. Because these networks use the same hardware, the setup is very easy and is usually plug and play. Although these networks are becoming popular, there are many downsides that would restrict usage for our target networks, namely the fact that these networks depend on hardware specific capabilities to enable the node to node communication, they do not allow configuration and customization of the network, and that they are not flexible enough to work with the needed network topology (geographic distance and both wired and wireless communication between nodes). These popular home networks use radio waves to communicate between nodes, and thus require radio support on the nodes and the base router. Since we are targeting situations that will not be able to rely on specific hardware capabilities, we need the ability to create the network from commodity hardware with off the shelf devices. This also extends to the software available to the users of this network, it should be free and open source, as well as available by download or by USB and other storage mediums. Since the network will always require an internet connection, we can rely that the user will have the ability to download required software and look at the provided documentation.

The current world requires an almost constant internet connection, and that is very difficult or expensive to accomplish in communities destroyed by natural disaster, or remote, impoverished, or war-torn communities. The past options were limited to: single ground-based back-haul connections which could be slow or prone to destruction, satellite connections which are very expensive, slow, and prone to brown outs, or other less reliable methods like using cell phone hot-spotting which in Canada is prohibitively expensive as a home WiFi solution [3] [4]. Recently, there are new internet network providers like SpaceX's Starlink that have the possibility to allow users to have a fast, (relatively) cheap, and resilient internet connection. Combining new internet alternatives with an Ad-hoc Mobile Mesh Network allows the communities that have had little access to the modern internet an ability to share an internet connection locally, reaching more people. This also allows the users to run and maintain their own network, granting governance and customizability to the users of the network.

4 Related Work

There has been numerous papers on the various measurements of mesh networks. Such papers have compared performance in terms of throughput, latency, and jitter in mesh networks [5], and energy efficiency of the different routing protocols [5]. A large concern when evaluating these papers was that they used a network simulator and the simulators used either were incapable of properly representing a mesh network [5], or did not account for the difference of hardware behaviours. As with most simulations, results should be validated in the field, and when we endeavoured to replicate the setups, we found that the setups were undocumented and they had not been validated on real hardware. It is because the papers did not actually create the setups that we went searching for real life projects where mesh networks are in use to bring the internet to a community of people. We focused on 3 projects that we found: Freifunk, NYC Mesh and Helium Network [6] [7] [8]. None of these target the same use case as us, but are interesting in their own respects. Freifunk is a grassroots movement to provide a network that delivers free internet to its constituents [6]. It uses hardware with specific mesh technologies like radio transceivers placed on top of buildings that are positioned towards similar nodes on other buildings. This creates a mesh networks over the participating German cities that allows citizens of these cities to connect to the mesh network. Some of the solutions to technical hurdles that Freifunk had to overcome have since been released as open source software. One of these projects is B.A.T.M.A.N, a routing protocol created by Freifunk member's that aims to improve over OLSR [9]. NYC Mesh is a similar project as Freifunk but resides in New York City. NYC Mesh has gained worldwide attention and has been a great marketing opportunity for free mesh networks. It is very similar to Freifunk in its goals and usage, but relies more on line of sight WiFi connections of up to 1 mile [7]. Helium Network is different from the other two previously mentioned networks. It is powered by the Helium Blockchain and focuses on high decentralization of its network. The network relies on a subset of users to setup transceivers so the network can propagate connection to the users [8]. Like Freifunk and

NYC Mesh, Helium Network relies on mesh networking hardware to communicate with the devices of the network. Users of the network pay for the usage with a cryptocurrency of the Helium Network, and as such the users who provide nodes and connections are payed for the usage of their hardware (to connect to the network). Unlike Freifunk and NYC Mesh, Helium Network focuses on IoT devices and as such uses LoRa based communication protocols for low powered devices [8]. After finding and evaluating these projects and other smaller but similar ones, we concluded that each project was missing a key functionality. All of the projects were deemed unfit by either being too expensive for users, relying on specific hardware, or requiring previous infrastructure. These projects were key in evaluating the need and interest in creating a mesh network that supplies communities with freedom, and access to the internet.

5 Project Goals

Our main goal for this project is to assess the possibility of setting up an Ad-hoc Mobile Mesh Network using commodity hardware, no previous internet infrastructure, and a single internet connection to share. More specifically we will assess the feasibility of using commodity hardware in the configuration described in the following sections. We will assess the possibility of using all open source software to configure, setup and run the network. We will assess the state of the community around setup guides and documentation. We will also run a battery of both qualitative and quantitative tests and measurements to see if the configured network is usable in ways users would expect. After setting up and assessing the network, we will describe the pain points we encountered, and we will describe what we believe are possible areas of improvement and next steps in the network.

At the end of the project we hope to have accomplished the following: deployed an Ad-hoc Mobile Mesh Network, used commodity hardware (in particular 2 different consumer grade routers and a singular Raspberry Pi), configured and deployed open source software (including the Babel routing protocol) to the network, tested various qualitative measurements on the network to see usability, and perform a small set of quantitative network performance tests.

6 Setup Choices (routing, hardware, software)

6.1 Setup Choices

As stated in the project goals, we build an understanding of current mesh-network technologies in addition to achieving this on commodity hardware.

6.1.1 Routing

We ended up choosing Babel for the routing protocol to build our mesh network. There are a number of key reasons why we chose Babel over other routing protocols. First and foremost, Babel is a "loop avoiding" distance-vector routing protocol. This means we can use it in a mesh network topology and not have any loops. Furthermore, Babel is aware of various link types and can optimize accordingly. In practice this means differentiating between wired and wireless links but could also be used over a VPN link. The underlying implementation and specifications are still actively updated and developed as well as the RFC being readily available appealed to our motivations [10].

6.1.2 Hardware

The setup uses two consumer-grade routers and one Raspberry Pi 3B [11]. One router was a Ubiquiti EdgeRouter X (er-x) [12]. It was used as the backhaul connection and as a purely wired node. The second router used was the TP-Link Archer C5 [13]. It was used for both wired and wireless connections (to the er-x and Raspberry Pi, respectively). The Raspberry Pi was also a hybrid wired/wireless node. It also functioned as a wireless access point.

6.1.3 Software

Given our usage of commodity hardware and the lack of any advanced routing protocols on their stock firmware, we decided to use OpenWrt on all nodes. OpenWrt is a Linux-distribution focused on low-power embedded systems [14]. All the devices that were to participate within the mesh network supported OpenWrt on the same version. The Babel routing daemon can be installed onto OpenWrt as well.

7 Measurements

One of the main motivations of doing a real life implementation of the network, as opposed to using a network simulator like the papers we reviewed, was to see how the network would behave in conditions not properly simulated. To properly test the network, we have devised a set of both qualitative and quantitative tests and measurements. The quantitative measurements will be typical performance measurements and will include measuring the throughput, latency, and jitter in the network between nodes, and between nodes and the gateway. The qualitative measurements we take will be less exact (as is their nature), nonetheless we will measure the network under the following circumstances: group audio calling (VoIP), group video calling, group video calling while screen sharing, and group video calling while screen sharing and video streaming. These tests and measurements represent a wide swath of potential load issues, and connection issues that will test if the network is usable, even under very stressful loads.

8 Topology

The network topology will consist of a singular super node that has a back-haul internet connection, and 2 regular nodes that are connected with each other and the super node in a mesh configuration. The super node is a Ubiquiti EdgeRouter X with wired Ethernet connections to the outside internet, the other TPLink router that acts as a node, and the Raspberry Pi 3B that acts as the last node. The wired connections use Cat5E Ethernet cable. Between the Raspberry Pi 3B and the TPLink router there is a wireless WiFi connection. Furthermore, because of the Raspberry Pi's dual wired/wireless ability, it acts as an access point for devices wanting to use the network. Figures 2 and 3 are both network diagrams for our mesh topology.

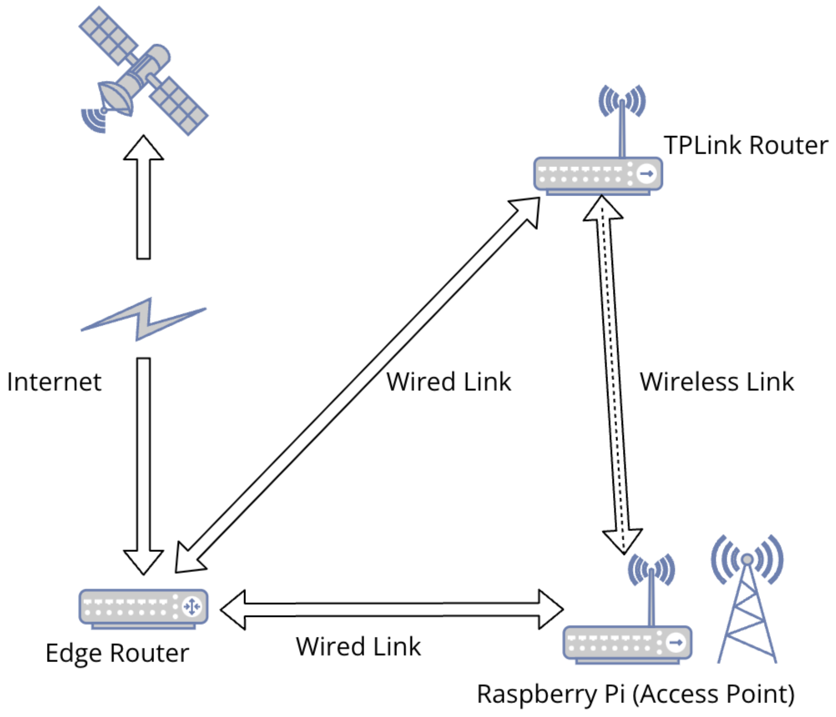


Figure 2: Topology

Below (Figure 3) is the network topology diagram that emphasizes the letter notation used throughout our work.

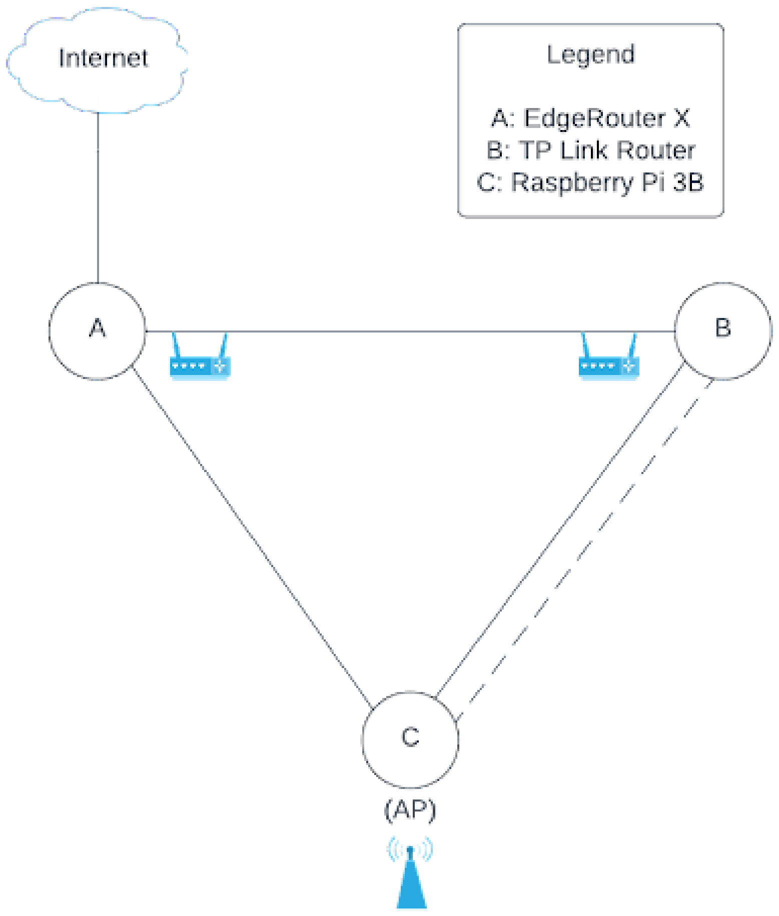


Figure 3: Topology

9 Demonstration

After the network setup was complete, it was used with a variety of applications that are common for the target users. These include VoIP, video calling, screen sharing, and video streaming. The main service used for these applications was Discord. For video streaming, however, YouTube and Netflix were used. A screen capture of this demonstration can be seen in Figure 4 below.

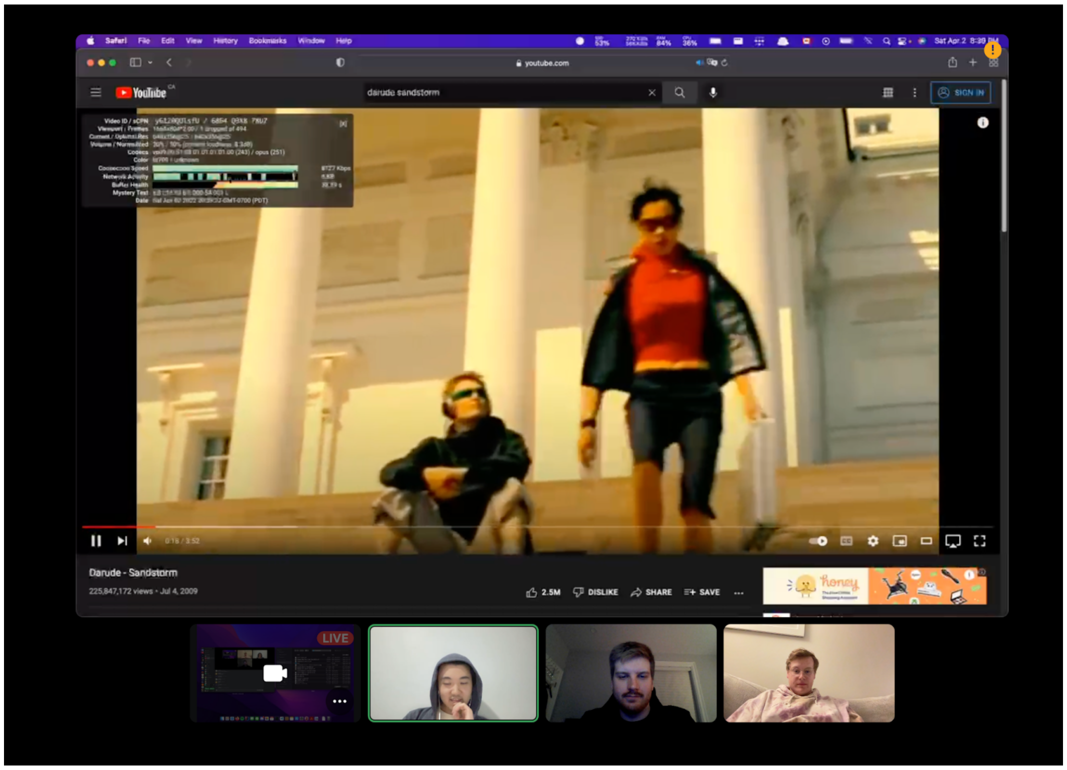


Figure 4: Video call with screen share and YouTube VoD over mesh network

In the rest of this section, we will discuss our qualitative and quantitative measurements of these network applications.

10 Results and Analysis

The network was subjected to a series of tests. The Qualitative Measurements section will discuss the user experience from our demonstration, followed by the Quantitative Measurements which will display and briefly analyze the results from various network tests that were performed on the mesh network.

10.1 Qualitative Measurements

The applications over the mesh network were largely usable. In the "live" applications (VoIP, other forms of calling/streaming), the transmission would disconnect for 3 to 5 seconds while the network switched nodes (triggered purposefully).

Using VoIP over the mesh network provided an overall good experience. Quality was very good and jitter was only noticeable a few times over a call length of several minutes. There was no call dropping and the network quickly converged during node switching.

While video calling, similar qualities to VoIP were observed. The calls had similar audio quality but the video quality was less than expected (Figure 5). There was noticeable jitter and quality fluctuated throughout the calls but it was overall usable. There were still no calls that were dropped and the network converged quickly while switching nodes.

Figure 5: Video call over mesh network with Discord

The introduction of screen sharing into the video call forced a decrease of video quality but it was still very usable. The screen share video feed had high quality but some noticeable jitter.

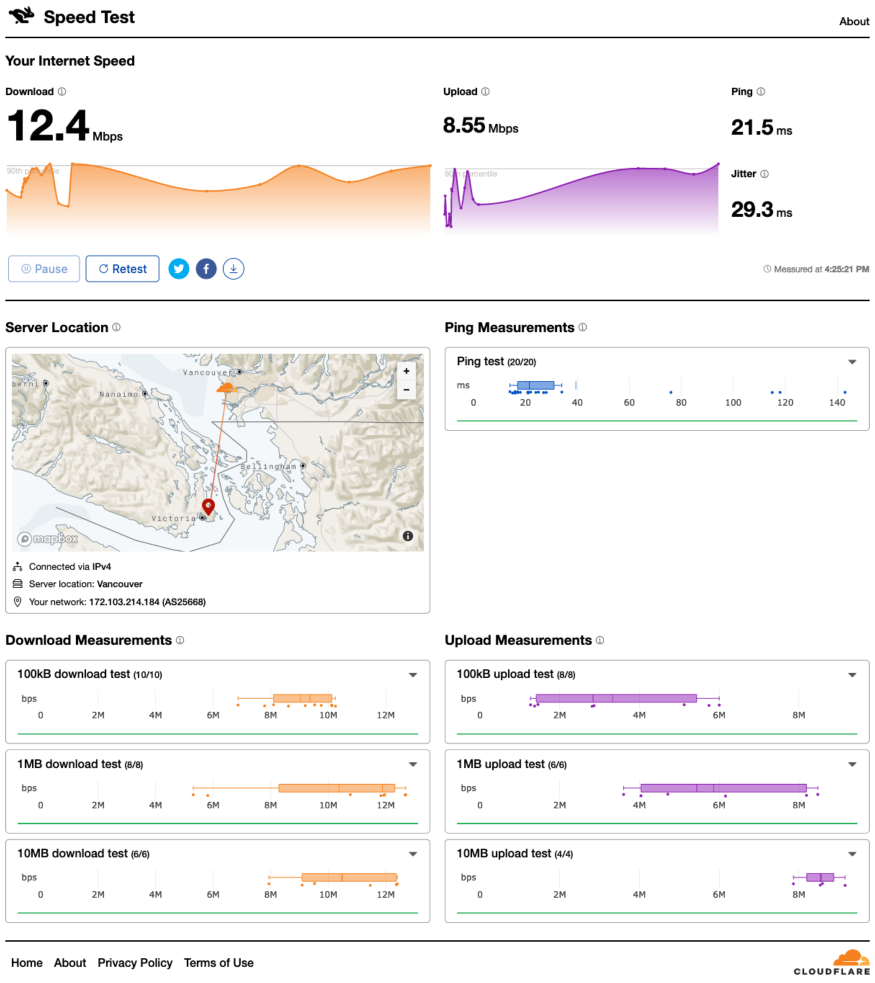
Overall, the mesh network handled multiple workloads very well. There were some dropped calls and jitter, but only for instances of video calling, screen sharing, and video streaming.

10.2 Quantitative Measurements

Various quantitative measurements were collected on the mesh network. The two main types were Internet speed tests with Cloudflare and network performance measurement with iperf3. Overall, these measurements are consistent with the qualitative measurements and what we saw in the demonstration in terms of quality and performance.

10.2.1 TP-Link Router to Raspberry Pi (Wireless) to EdgeRouterX (Wired)

Figure 6 shows the Cloudflare Internet speed test from the TP-Link router wirelessly to the Raspberry Pi, wired to the er-x, then out into the Internet.



Your Internet Speed

Download 12.4 Mbps

Upload 8.55 Mbps

Ping 21.5 ms

Jitter 29.3 ms

Measured at 4:25:21 PM

Server Location

Connected via IPv4

Server location: Vancouver

Your network: 172.103.214.184 (AS25668)

Ping Measurements

Ping test (20/20)

Download Measurements

100kB download test (10/10)

1MB download test (8/8)

10MB download test (6/6)

Upload Measurements

100kB upload test (8/8)

1MB upload test (6/6)

10MB upload test (4/4)

Figure 6: BCA Speed Test

As seen below in Figure 7, the iperf3 measurements through the network show a mean bitrate of 10.3 Mbps over a period of 10 seconds.

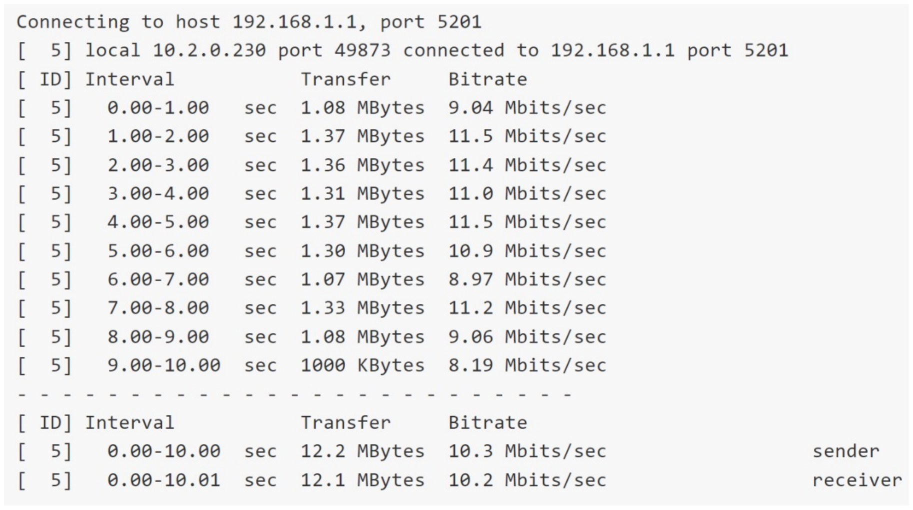

```
Connecting to host 192.168.1.1, port 5201
[  5] local 10.2.0.230 port 49873 connected to 192.168.1.1 port 5201
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  1.08 MBytes  9.04 Mbits/sec
[  5]   1.00-2.00   sec  1.37 MBytes  11.5 Mbits/sec
[  5]   2.00-3.00   sec  1.36 MBytes  11.4 Mbits/sec
[  5]   3.00-4.00   sec  1.31 MBytes  11.0 Mbits/sec
[  5]   4.00-5.00   sec  1.37 MBytes  11.5 Mbits/sec
[  5]   5.00-6.00   sec  1.30 MBytes  10.9 Mbits/sec
[  5]   6.00-7.00   sec  1.07 MBytes  8.97 Mbits/sec
[  5]   7.00-8.00   sec  1.33 MBytes  11.2 Mbits/sec
[  5]   8.00-9.00   sec  1.08 MBytes  9.06 Mbits/sec
[  5]   9.00-10.00  sec  1000 KBytes  8.19 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.00  sec  12.2 MBytes  10.3 Mbits/sec                  sender
[  5]   0.00-10.01  sec  12.1 MBytes  10.2 Mbits/sec                  receiver
```

Figure 7: BCA iperf3 Measurements

10.2.2 Raspberry Pi to EdgeRouterX (Wired)

This wired connection connects the Raspberry Pi with the EdgeRouterX through a Cat5E Ethernet cable. As seen in Figure 8 below, the mean bitrate was 94.7 Mbps over a period of 10 seconds.

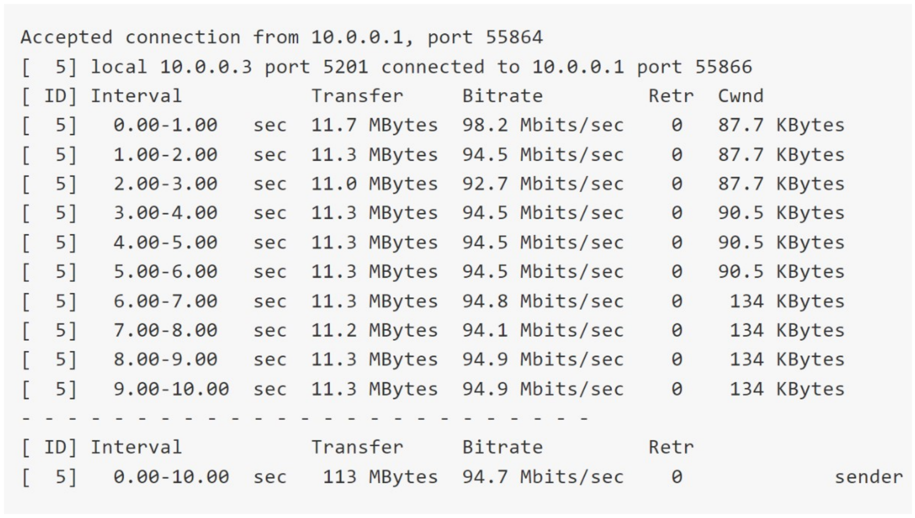
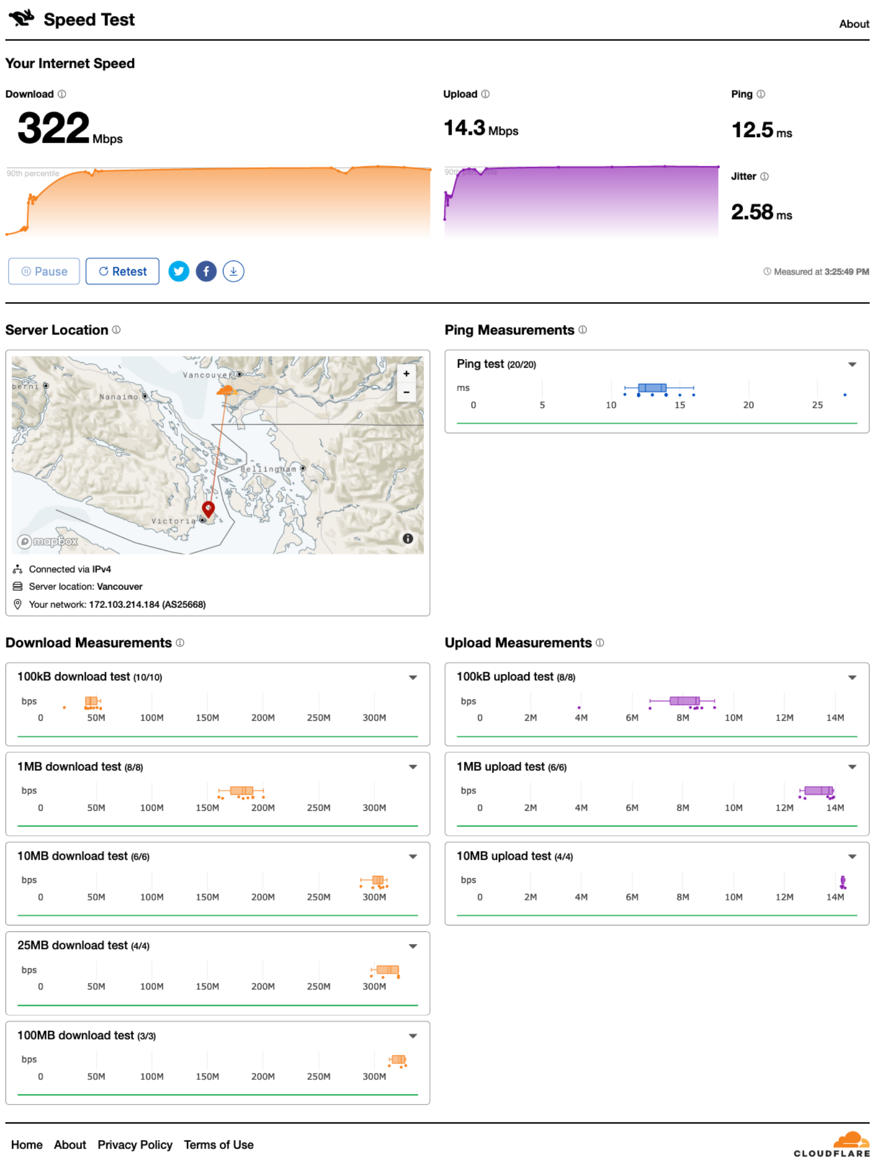

```
Accepted connection from 10.0.0.1, port 55864
[  5] local 10.0.0.3 port 5201 connected to 10.0.0.1 port 55866
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  11.7 MBytes  98.2 Mbits/sec    0   87.7 KBytes
[  5]   1.00-2.00   sec  11.3 MBytes  94.5 Mbits/sec    0   87.7 KBytes
[  5]   2.00-3.00   sec  11.0 MBytes  92.7 Mbits/sec    0   87.7 KBytes
[  5]   3.00-4.00   sec  11.3 MBytes  94.5 Mbits/sec    0   90.5 KBytes
[  5]   4.00-5.00   sec  11.3 MBytes  94.5 Mbits/sec    0   90.5 KBytes
[  5]   5.00-6.00   sec  11.3 MBytes  94.5 Mbits/sec    0   90.5 KBytes
[  5]   6.00-7.00   sec  11.3 MBytes  94.8 Mbits/sec    0    134 KBytes
[  5]   7.00-8.00   sec  11.2 MBytes  94.1 Mbits/sec    0    134 KBytes
[  5]   8.00-9.00   sec  11.3 MBytes  94.9 Mbits/sec    0    134 KBytes
[  5]   9.00-10.00  sec  11.3 MBytes  94.9 Mbits/sec    0    134 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec   113 MBytes  94.7 Mbits/sec    0             sender
```

Figure 8: CA iperf3 Measurements

10.2.3 EdgeRouterX to Internet

This measurement covers the route from the EdgeRouterX to the Cloudflare server in Vancouver. As seen in Figure 9, the download speed was 322 Mbps and upload was 14.3 Mbps. Also, there is a very small amount of jitter (2.58 ms) and a small ping, 12.5 ms.


Speed Test

About

Your Internet Speed

Download 322 Mbps

Upload 14.3 Mbps

Ping 12.5 ms

Jitter 2.58 ms

Pause Retest

Measured at 3:25:49 PM

Server Location

Connected via IPv4

Server location: Vancouver

Your network: 172.103.214.184 (AS25668)

Ping Measurements

Ping test (20/20)

Download Measurements

100kB download test (10/10)

1MB download test (8/8)

10MB download test (6/6)

25MB download test (4/4)

100MB download test (3/3)

Upload Measurements

100kB upload test (8/8)

1MB upload test (6/6)

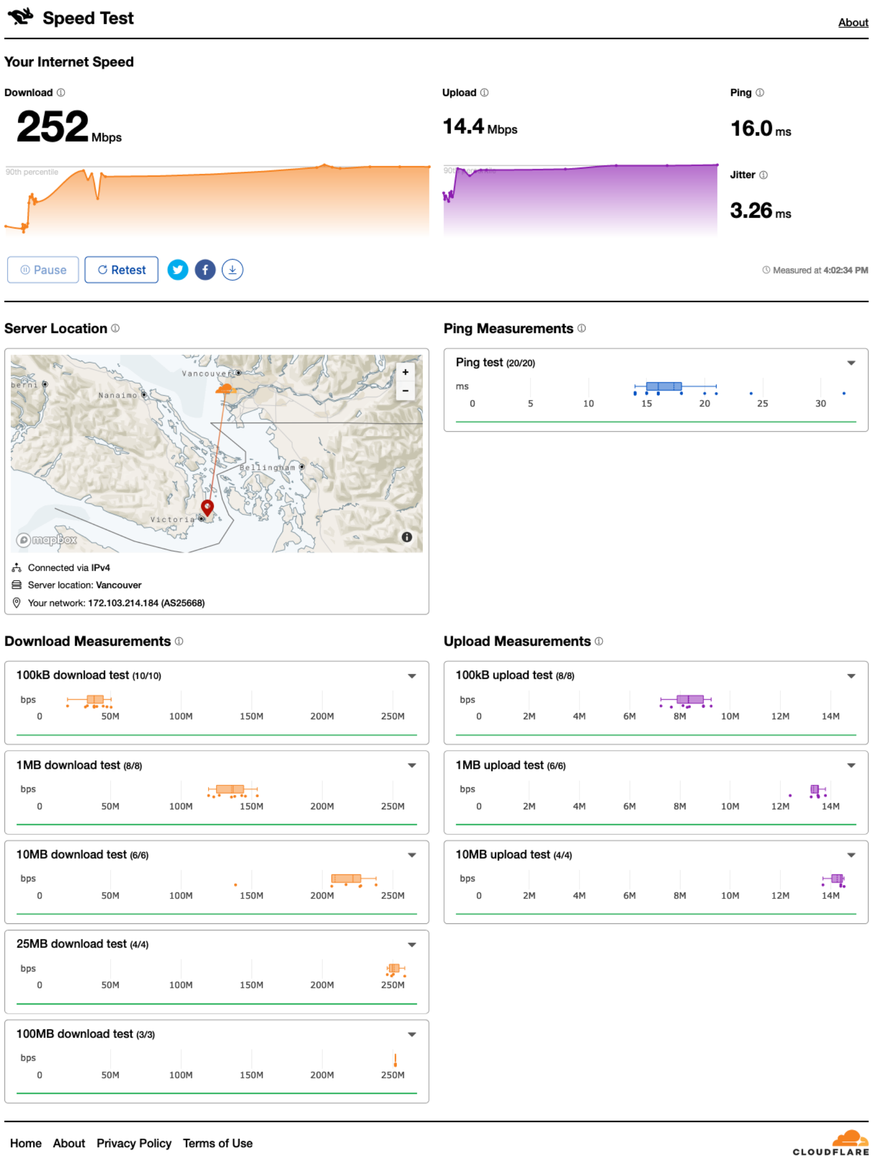
10MB upload test (4/4)

Home About Privacy Policy Terms of Use

Figure 9: A Speed Test

10.2.4 TP-Link Router to Internet

This measurement covers the route from the TP-Link Router to the Cloudflare server in Vancouver. As seen in Figure 10, the download speed was 252 Mbps and upload was 14.4 Mbps. Also, there is a very small amount of jitter, 3.26 ms, and a small ping at 16.0 ms.



Speed Test

Your Internet Speed

Download 252 Mbps

Upload 14.4 Mbps

Ping 16.0 ms

Jitter 3.26 ms

Measured at 4:02:34 PM

Server Location

Connected via IPv4

Server location: Vancouver

Your network: 172.103.214.184 (AS25668)

Ping Measurements

Ping test (20/20)

Download Measurements

100kB download test (10/10)

1MB download test (8/8)

10MB download test (6/6)

25MB download test (4/4)

100MB download test (3/3)

Upload Measurements

100kB upload test (8/8)

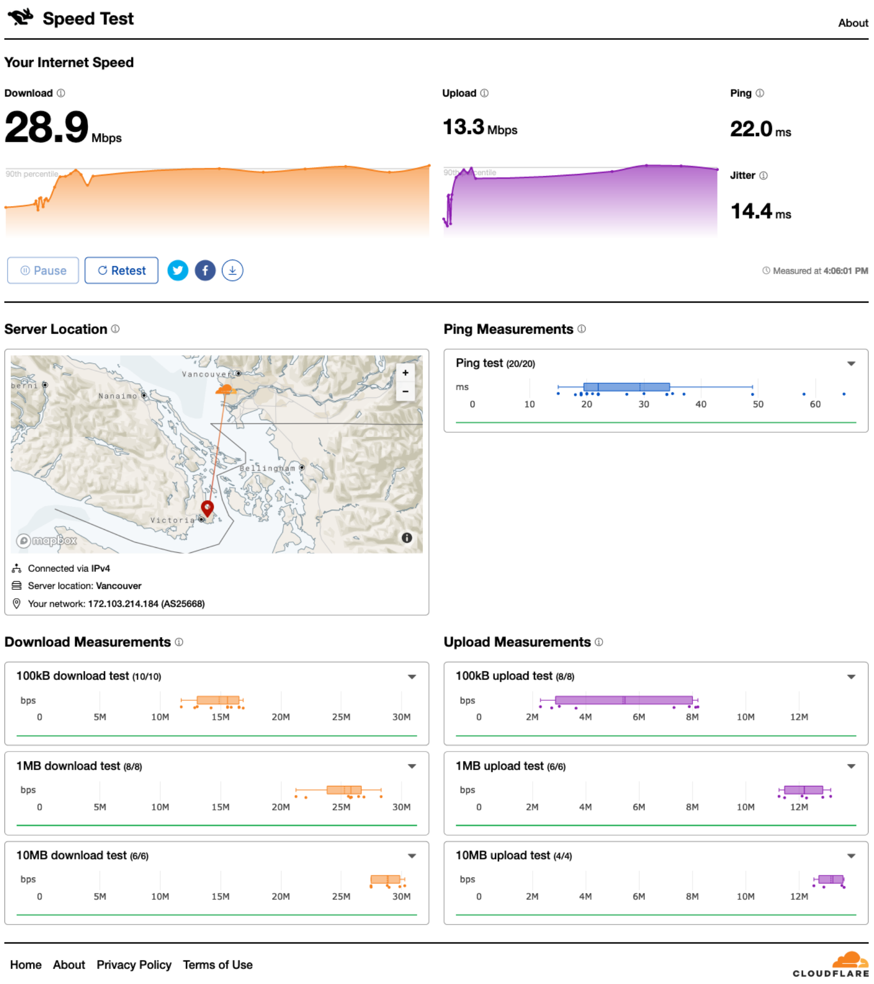
1MB upload test (6/6)

10MB upload test (4/4)

Figure 10: B Speed Test

10.2.5 Raspberry Pi to Internet

This measurement covers the route from the TP-Link Router to the Cloudflare server in Vancouver. As seen in Figure 11, the download speed was 28.9 Mbps and upload was 13.3 Mbps. Also, the jitter is at a decent level, 14.4 ms, and there is a good ping, measuring at 22.0 ms.



Speed Test

Your Internet Speed

Download 28.9 Mbps

Upload 13.3 Mbps

Ping 22.0 ms

Jitter 14.4 ms

Measured at 4:06:01 PM

Server Location

Connected via IPv4

Server location: Vancouver

Your network: 172.103.214.184 (AS25668)

Ping Measurements

Ping test (20/20)

Download Measurements

100kB download test (10/10)

1MB download test (8/8)

10MB download test (6/6)

Upload Measurements

100kB upload test (8/8)

1MB upload test (6/6)

10MB upload test (4/4)

Figure 11: C Speed Test

11 Discussion

In this section, we will discuss the process and analysis of our network in the problem space of our project's stated goals.

Most of the pain points in the set up of the mesh network were caused by a lack of documentation. It was not clear whether the on-board WiFi chip for the Raspberry Pi supported Ad-hoc mode so discovering that was mostly trial-and-error [15]. Also, the Babel documentation is mostly concerned with the theory and design of the routing protocol and there was no central repository for implementation details. The resources mainly used were the OpenWrt wiki [16], Babel GitHub README [10], and Babel daemon man page [17].

Since we were using consumer devices, there was no default management port. That meant that we had to manage and configure the network through the network itself. If there was a misconfiguration somewhere, it would render the network inaccessible, meaning configuration would have to start from the start again. Wireless configuration output was minimal, if it showed at all. This made the whole process a lot more difficult.

11.1 Hardware

While commodity hardware is plentiful, affordable and abundant compared to enterprise and professional network gear, we encountered numerous issues as a result of our selection of hardware.

Management Interface

We immediately encountered a trivial but fundamental inconvenience with all the hardware we were working with. None of the devices we were working with had a dedicated management interface. By default, all these devices when running OpenWrt could only be accessed via. its hardware Ethernet ports and a static IP address. However if a bad configuration is committed to the device, this could result in losing access to the device. This was mitigated slightly by running a VLAN out of the CPU-connected Ethernet interface to the built-in network switch of each of these devices (except for the Raspberry Pi). A management interface generally does not require a full gigabit Ethernet port so it is a bit of a waste of resources. Needless to say, we were locked out of our devices during development and setup more than we would like to admit.

WiFi Drivers

A useful feature of Babel was its awareness and subsequent handling of wireless links in the mesh network. Unfortunately we ran into many issues with the WiFi adapters in all the devices present in our mesh network. The only configuration we were able to get working was on 802.11 N. This severely bottle-necked our wireless links. Furthermore, the antennas on all Raspberry Pi were not designed for access point duties.

Routing Speed Limitations

The low-cost and low-power nature of these devices resulted in observing limitations to how many packets a node could process per-second. Despite being attached into a full-gigabit Ethernet network, we were unable to fully saturate links within the network. These CPUs are unable to process packets within the mesh network at line speed. These SoCs usually have the ability to perform hardware offload of packet routing. Once again, we encountered issues with drivers within OpenWrt to successfully utilize any hardware offload capabilities.

11.2 Software

In this section we will review our experiences with the software side of our mesh network setup. First, we discuss bootstrapping each node, then we move on to discuss the configuration process of Babel.

11.2.1 Bootstrapping

Similar to the issues described in regard to the lack of management interface on these devices, a lot of the setup on each node initially requires an internet connection on the node being setup. As such, Bootstrapping a new node on the network required connecting it to the internet out-of-band (i.e. not through the mesh network) before it could participate in the mesh network. Specifically, the internet connection was required to install several packages notably the Babel package to get routing configured and thus an internet connection.

11.2.2 Babel

Our experience of getting Babel running and configured within our mesh network was not trivial. There was no step-by-step tutorial available on the Internet in our search. The documentation for the actual Babel protocol can be found easily but our search for documentation to get it running proved lackluster. We believe this gap in documentation between the core technical details currently available and resources such as tutorials to get mesh networks running hinder adoption and general interest in the technology. Once we were able get our mesh network running, we were able to run experiments and learn how it worked. Understanding how Babel worked involved looking at packet dumps as well.

The diagnostic tools for Babel were either severely lacking or not documented. When attempting to configure the mesh network it was hard to tell what the Babel routing daemon was doing at a given moment. We primarily relied on the interface that exists between OpenWrt ubus and the Babel daemon running on the node [16]. This was combined with the log output of the Babel daemon.

12 Future Work

Within our mesh network, we provided internet access directly using the coax-based internet connection we had available in our lab setting. As mentioned in our motivations sections, we believe that combining mesh networks with internet connections like Starlink would have great potential. Therefore, for future mesh network experiment work, we could potentially simulate the Internet connection characteristics of Starlink using tc-netem [18]. This would allow our experimentation and demos to more accurately reflect the general user experience one might experience on a real Starlink connection.

The usage of commodity network gear was a great challenge but for future work we would likely standardize which commodity hardware we use. This could allow the device to be better understood and configured in the best possible configuration. At the very least removing some of the issues surrounding WiFi drivers would allow us to better focus on the mesh networking aspects.

Within our configuration of Babel, it was mostly kept to default settings besides indicating the type of link of each interface and which routes to distribute. Babel has a lot more configuration which were beyond the scope of this project however tweaking these configuration parameters could change network characteristics like adding more latency in exchange for more nodes on the network. Extensions also exist for Babel like actively probing the round-trip time of links (BabelRTT) which takes this metric into account for routing decisions.

While we were able to get a functioning hybrid mesh network working in our project, developing this network further to mimic something closer to what one might see in the real world would further develop this project. Adding more internet gateways in the network and in general more nodes participating would take this project to the next level.

Our configuration of the internet was extremely simple. We simply distributed the default route (0.0.0.0/0) throughout the network. If the mesh network were to become multi-homed, we could experiment with a smarter routing policy through the network. The mesh network could choose the best gateway for a given route. An example could be to configure the prefix of Google's public DNS servers to route over the link with the lower latency. This would open the door to optimizing the usage of links to maximize QoS.

As mentioned in our discussion section about the lack of diagnostic and observability tools for Babel, this could potentially be future work on creating these tools and contributing to the community and make Babel and mesh networking technologies more accessible. This could include anything from command-line interface utilities to graphical applications that visualize the network in real-time.

OpenWrt provides many utilities that are well documented. The image builder could be used potentially in future work to reduce the amount of manual configuration and internet bootstrapping required by customizing firmware images and pre-installing the Babel package and any other required packages. The image could also include basic configurations for Babel to help speed up the process of bootstrapping a node.

13 Conclusion

The goal of this paper was to assess the usability and performance of low-cost, accessible Ad-hoc Mobile Mesh Networks using consumer-grade hardware and open-source firmware. We were successful in setting up a mesh network with a TP-Link router, Raspberry Pi 3B, EdgeRouterX, all with OpenWrt and using Babel as the routing protocol. Babel was chosen for the full mesh network setup because it was the most usable for our hardware compared to B.A.T.M.A.N. and OLSR mesh routing protocols. Even though it was the most usable, the set up with Babel presented its own challenges. The set up was difficult and delicate but possible for people with our level of expertise. This mesh network was able to support video calls with screen sharing and video streaming at the same time. The quantitative results reflect what we witnessed during the demonstration and qualitative results, with an entire-mesh down bitrate of about 12.3 Mbps at the worst case. The next steps are to expand the network with more devices, varying ranges, different routing protocols, and maybe even use an actual Starlink connection to connect neighbours or a small community. Also, during this future work, we will record additional documentation for the community.

References

[1] "Starlink," Starlink. (), [Online]. Available: https://www.starlink.com (visited on 02/06/2022).

[2] Mykhailo Fedorov [@FedorovMykhailo]. "@elonmusk, while you try to colonize mars — russia try to occupy ukraine! while your rockets successfully land from space — russian rockets attack ukrainian civil people! we ask you to provide ukraine with starlink stations and to address sane russians to stand.," Twitter. (Feb. 26, 2022), [Online]. Available: https://twitter.com/FedorovMykhailo/status/1497543633293266944 (visited on 04/11/2022).

[3] OpenMedia. "2021 report shows canada's cell phone prices STILL among most expensive globally." (), [Online]. Available: https://openmedia.org/article/item/2021-rewheel-report-shows-canadas-cell-phone-prices-still-among-most-expensive-globally (visited on 04/11/2022).

[4] "Price comparisons of wireline, wireless and internet services in canada and with foreign jurisdictions: 2020 edition - strategic policy sector." (), [Online]. Available: https://www.ic.gc.ca/eic/site/693.nsf/eng/00190.html#s0 (visited on 04/11/2022).

[5] J. L. E. K. Fendji and S. D. Samo, "Energy and performance evaluation of reactive, proactive, and hybrid routing protocols in wireless mesh network," *International Journal of Wireless & Mobile Networks*, vol. 11, no. 1, pp. 13–31, Feb. 28, 2019, ISSN: 09754679, 09753834. DOI: 10.5121/ijwmn.2019.11102. arXiv: 1903.06875. [Online]. Available: http://arxiv.org/abs/1903.06875 (visited on 04/11/2022).

[6] "What is freifunk about?" freifunk.net. (), [Online]. Available: https://freifunk.net/en/what-is-it-about/ (visited on 04/11/2022).

[7] "NYC mesh," NYC Mesh. (), [Online]. Available: https://www.nycmesh.net/ (visited on 02/06/2022).

[8] "Helium – introducing the people's network." (), [Online]. Available: https://www.helium.com/ (visited on 04/11/2022).

[9] *B.a.t.m.a.n.* In *Wikipedia*, Page Version ID: 1075850169, Mar. 8, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=B.A.T.M.A.N.&oldid=1075850169 (visited on 04/11/2022).

[10] J. Chroboczek and D. Schinazi, "The babel routing protocol," Internet Engineering Task Force, Request for Comments RFC 8966, Jan. 2021, Num Pages: 54. DOI: 10.17487/RFC8966. [Online]. Available: https://datatracker.ietf.org/doc/rfc8966 (visited on 04/11/2022).

[11] "[OpenWrt wiki] raspberry pi." (), [Online]. Available: https://openwrt.org/toh/raspberry_pi_foundation/raspberry_pi (visited on 04/11/2022).

[12] d. digge der. "Ubiquiti EdgeRouter x (ER-x), EdgeRouter x-SFP (ER-x-SFP) and EdgePoint r6 (EP-r6)," OpenWrt Wiki. Section: 2021-11-27T02:25:02-05:00. (Nov. 13, 2015), [Online]. Available: https://openwrt.org/toh/ubiquiti/edgerouter_x_er-x_ka (visited on 03/30/2022).

[13] "[OpenWrt wiki] TP-link archer c5 AC1200 / TP-link archer c7 AC1750 / TP-link TL-WDR7500." (), [Online]. Available: https://openwrt.org/toh/tp-link/archer-c5-c7-wdr7500 (visited on 04/11/2022).

[14] R. Brown. "Welcome to the OpenWrt project," OpenWrt Wiki. Section: 2021-09-04T19:09:24-04:00. (Sep. 27, 2016), [Online]. Available: https://openwrt.org/start (visited on 04/11/2022).

[15] "[PATCH 15/15] brcmsmac: Add support for adhoc mode (linux wireless)." (), [Online]. Available: https://www.spinics.net/lists/linux-wireless/msg105243.html (visited on 03/30/2022).

[16] zorun. “Babel routing protocol (babeld),” OpenWrt Wiki. Section: 2021-07-23T12:51:10-04:00. (Sep. 3, 2014), [Online]. Available: `https://openwrt.org/docs/guide-user/services/babeld` (visited on 03/30/2022).

[17] “Ubuntu manpage: Babeld - ad-hoc network routing daemon.” (), [Online]. Available: `https://manpages.ubuntu.com/manpages/bionic/man8/babeld.8.html` (visited on 04/11/2022).

[18] “NetEm - network emulator at linux.org.” (), [Online]. Available: `https://www.linux.org/docs/man8/tc-netem.html` (visited on 04/11/2022).

Appendix A Contributions

A.1 Aomi Jokoji

- Researched Babel
- Mesh routing protocol research
- Babel configuration
- Hardware configuration
- Recording demo videos
- Wireless driver evaluation
- Project paper discussion and future work
- Mesh network measurement

A.2 Adam Hultman

- Website creation
- Website Updates
- Recording presentation videos
- Presentation slides content
- Presentation editing
- Project Update content
- Researched OLSR
- Discussion replies

- Project paper editing
- Project paper research
- Project paper results & analysis, measurements, conclusion

A.3 Isaac Donaldson

- Researched similar projects (Freifunk, NYC Mesh, Helium Network)
- Researched papers
- Researched B.A.T.M.A.N.
- Project updates writing
- Discussion replies
- Presentation video recording
- Presentation coordinator and slide creator
- Project paper abstract, introduction, background, related work, goals, and topology
- Project paper editing