Introduction

It is commonplace for people to use mobile devices (MNs) and expect connectivity. In a typical situation, mobile devices (nodes) associate and communicate with wireless access points (APs) and use them to communicate with other hosts on the network, with the access points themselves communicating via wired connections. WiFi protocols such as the IEEE 802.11 family of standards allow nodes to communicate wirelessly in the presence of other transmitters, whereas wired protocols such as 802.3 Ethernet standards allow wired devices to communicate. We have implemented protocols similar to IEEE 802.11 and IEEE 802.3 to allow mobile nodes to communicate and tested them using the cnet network simulator to evaluate their performance.

Description of Protocols

The data-link layer (DLL) protocols we have designed are similar to the IEEE 802.11 and IEEE 802.3 standards in functionality. The implementation of these protocols was using a simplified model of the UWA CSSE network. This means our protocols have been tested under the assumptions that the distance between mobile hosts and access points is on the order of meters (this can be seen in the included topology files).

In order to overcome the hidden and exposed node problems in Wi-Fi communication systems, CSMA/CA has been implemented. The Wi-Fi protocol has implemented an RTS/CTS handshake to avoid collisions. Before a Wi-Fi frame is transmitted, an RTS frame is sent to the destination, which responds with a CTS frame. Each node which 'overhears' either of these will then cease transmission for a time period specified in the RTS or CTS.

For the purposes of association, the Wi-Fi data link layer on each MN broadcasts a probe frame nearby APs, to determine round trip time and signal strength. A table of APs is thereby maintained by each MN. Nodes associate with Access points based upon the best signal strength; if they are not associated with the Access point with the best signal strength, the MN will associate with it and send a disassociation notice to the Access point it was previously associated with (if any).

Mobile nodes and wired connections will attempt to associate with an Access point upon simulation startup, with a small randomized delay to ensure that upon starting the simulation, the system is not deterministically flooded with simultaneous association attempts. Mobile nodes probe regularly for access points in order to maintain their respective AP information tables (containing signal strength etc.).

The DLL Ethernet protocol implements CSMA/CD with binary exponential back-off, in a manner similar to that employed by the IEEE Ethernet standard. For each data-link layer protocol, APs maintain a FIFO queue of frames to be sent, so the interior of the network has some buffering capability. Both data-link layer protocols employ CRC-32 checksum calculation, and will drop frames which fail a checksum. Furthermore, there is no DLL acknowledgement, a decision which made in accordance with conventional 'intelligence at the edge' network design philosophy.

At the network layer, each MN maintains a sliding-window queue for each destination to which it sends packets, and a corresponding queue for each destination from which it receives them. When a mobile node retrieves a packet from the Application layer it is appended to the relevant queue. Each node pair then employs selective-repeat ARQ to ensure in-order delivery.

Description of Experiments

Our experiments were carried out with the use of a python script (generate_data.py) to automatically generate and execute topology files, in order to obtain data given varying rates of frame corruption, frame loss, message send rates and number of nodes.

However, the cnet simulator is designed in such a way that simulations run very slowly if individual nodes have nontrivial quantities of data associated with them as variables. In order to overcome this limitation it would have been necessary to design the protocol code to keep most data on the heap while maintaining only pointers to the actual data in the node variables themselves.

At the outset of design, we did not expect the effect of this limitation of the simulator to be as large as it proved to be, and as such we chose to prioritize code maintainability and ease of debugging by not adopting this method of data-handling

Due to this, result our simulations have been run for a relatively short amount of simulated time. To retrieve a large as possible data set for accurate results, a 12 core 3.2 Ghz Intel Core i7 Linux server with 48 GB of RAM was used to process the data, running 10 concurrent cnet processes using the Python multiprocessing module. This did not decrease simulation time to run a single simulation, but allowed for multiple simulations to run concurrently.

Initial simulations were run with a total of 5 mobile nodes, 5 access points and a duration of 12 seconds. Further simulations were run with a total of 2 mobile nodes, 2 access points and a duration of 12 seconds. These further simulations were intended to have 5 mobile nodes and 5 access points; this error is described in the section "Identification of Problems". Finally, a third round of simulations was run with 5 mobile nodes and 5 access points.

Figure 1.2 demonstrates the layout used for the third round of simulations. The parameters that were varied include message generation rates, frame corruption rate and frame loss rate listed in Figure 1.1. Simulations were run for each permutation of the listed parameters, resulting in 54 total simulations in each round.

It is important to note that due to the small simulation time, frames generated in the initial seconds of the simulation while mobile nodes are still associating with access points would drastically alter averages, as a result the application layer was turned off for the first 2 seconds (for 12 seconds simulations) and 20 seconds (for 60 second simulations).

Ideally, simulations would occur for a longer amount of time to better understand long term performance, but this would take too long for cnet to simulate given the current implementation of the protocols. The number of mobile nodes, access points and duration were chosen as a balance between the time necessary to run the simulations and measuring the performance of the protocols within the simulation as accurately and reliably as possible.

Message Rate (microseconds)	**Frame Corruption (as entered into cnet)**	**Frame Loss (as entered into cnet)**
10 000	0	0
100 000	3	3
500 000	12	12

Figure 1.1: Parameters that were varied during simulation

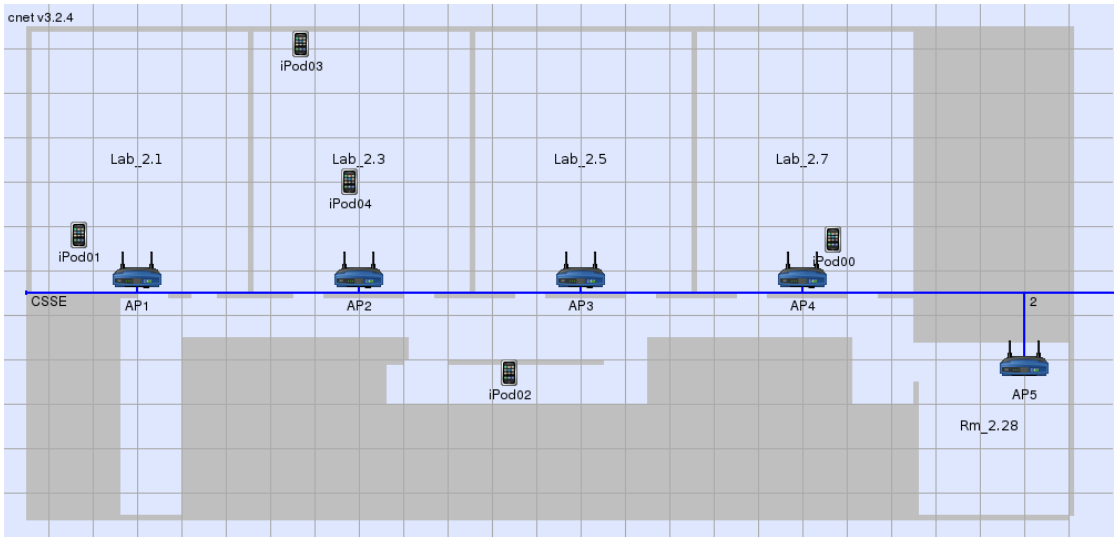


Figure 1.2: Seed used for simulation with 5 nodes and 5 access points

Initial Results

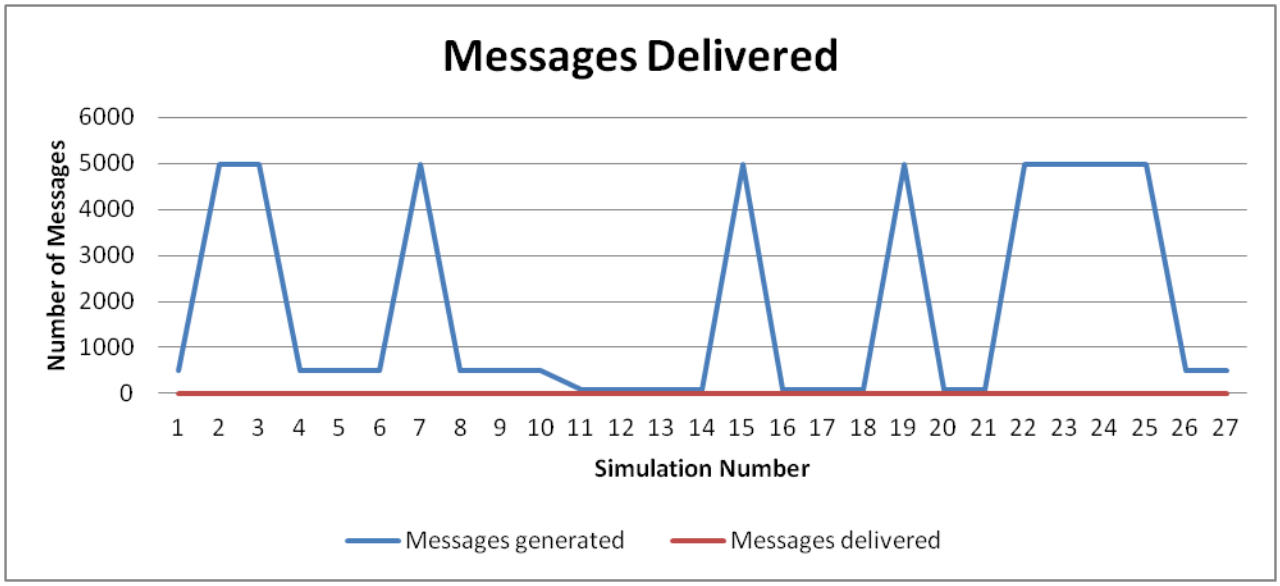


Figure 2.1: Number of messages delivered for 5 nodes and 5 access points

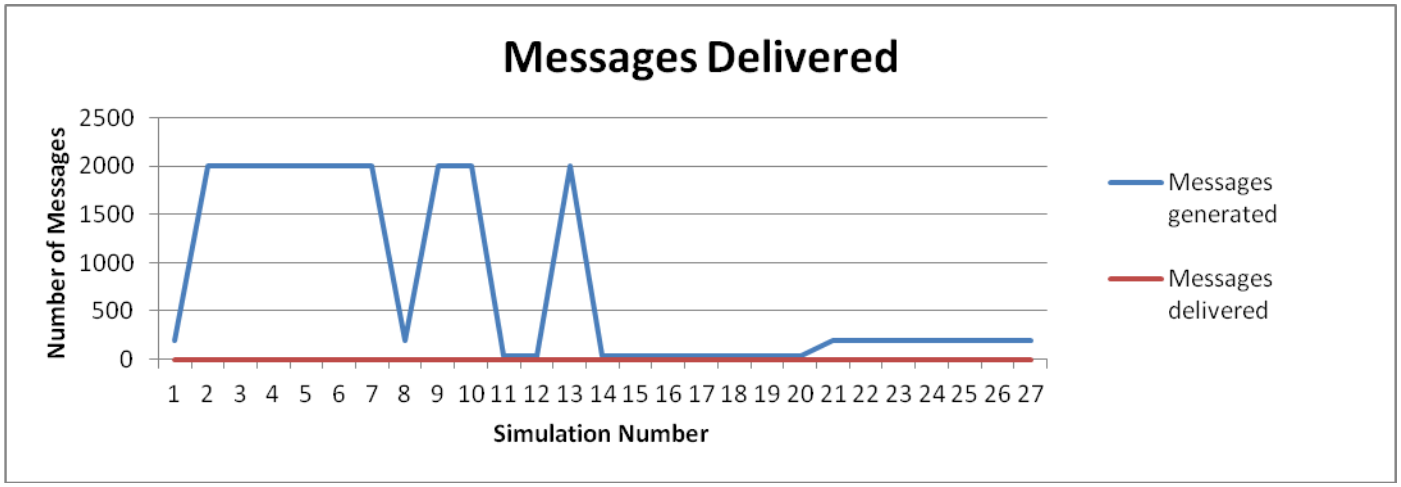


Figure 2.2: Number of messages delivered for 2 nodes and 2 access points

Figure 2.1 and 2.2 demonstrate that the protocols do not reliably transmit messages within a reasonable time frame. This could be due to multiple factors. Firstly, that the implementation of our protocols were not developed to be as compatible with cnet as possible. The short simulation duration time means that although our protocols were developed to send message between nodes, there may have not been enough time allowed to observe the messages being delivered. Another reason may be that the protocols may still have unforeseen bugs or errors that are preventing messages from being delivered.

It is important to note that there is one instance, simulation 2, of a message being delivered correctly. In the one instance of a message being delivered, it was during a simulation with 5 nodes and 5 access points, with a 12 second simulation time. This occurred under the parameters of message generation rate being 10 000 microseconds, frame corruption rate of 3 and frame loss rate of 3. In no other instance of running a simulation with 5 nodes and 5 access points, was a message delivered. Based on this one instance of a message being delivered, the average delivery time for a message is 3 622 546 microseconds.

This figure is not very reliable due to the limited number of messages that were sent and the small sample size used to calculate the average. The important result from this observation is that the implemented WiFi and Ethernet protocols are functional to some degree. The main limiting factor to the accuracy of this data is that we were only able to accommodate short simulation times. It is possible that in longer simulations, more messages would be successfully delivered.

If the protocol was working as expected, the data should show that messages are being delivered at a steady rate. For simulations with low rates of messages being generated, the number of messages being delivered should be expected to be relatively constant; due to the low amount of traffic throughout the network there will be less collisions and back off periods allowing the messages to be sent relatively quickly. In simulations where rates of message generation are high, the number of messages being delivered should rise and then plateau if the network reaches capacity. This is due to the large number of collisions occurring and the resulting back off periods due to CSMA/CA and CSMA/CD being implemented.

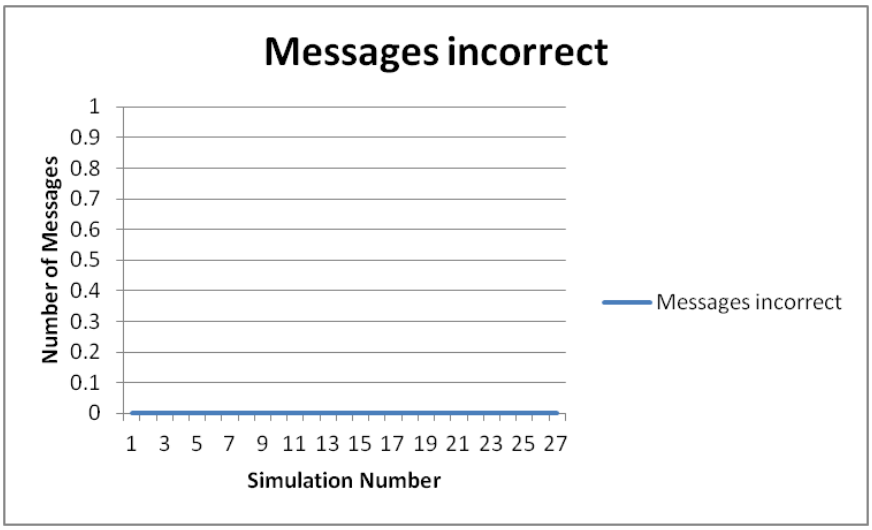


Figure 3: Number of messages incorrectly delivered for 5 nodes and 5 access points and also for 2 nodes and 2 access points

Figure number 3 indicates that no messages were delivery incorrectly. This could have many implications such as the protocols not functioning correctly, nodes not associating with access points or it may be a limitation of the short simulation time. This would initially suggest that messages are not being delivered at all, either incorrectly or correctly. There is one instance of a message being correctly delivered, which means that the implemented WiFi and Ethernet protocols can function, but they do not do so consistently.

If the protocol was working as expected, the number of messages being incorrectly delivered should be low due to proper allocation of mobile nodes to access points, proper allocation of MAC addresses for devices and correct routing throughout the network by the protocols accompanied by rigorous error checking(using CRC-32 checksum calculation) in each DLL link and in each end-to-end NL connection.

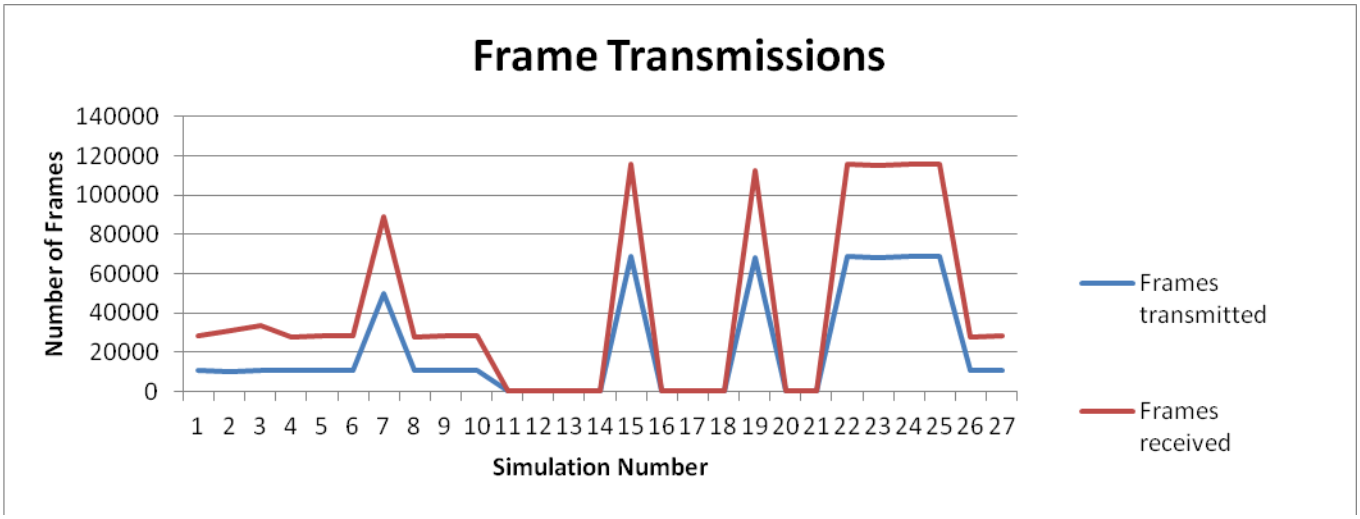


Figure 4.1: Number of frames transmitted for 5 nodes and 5 access points

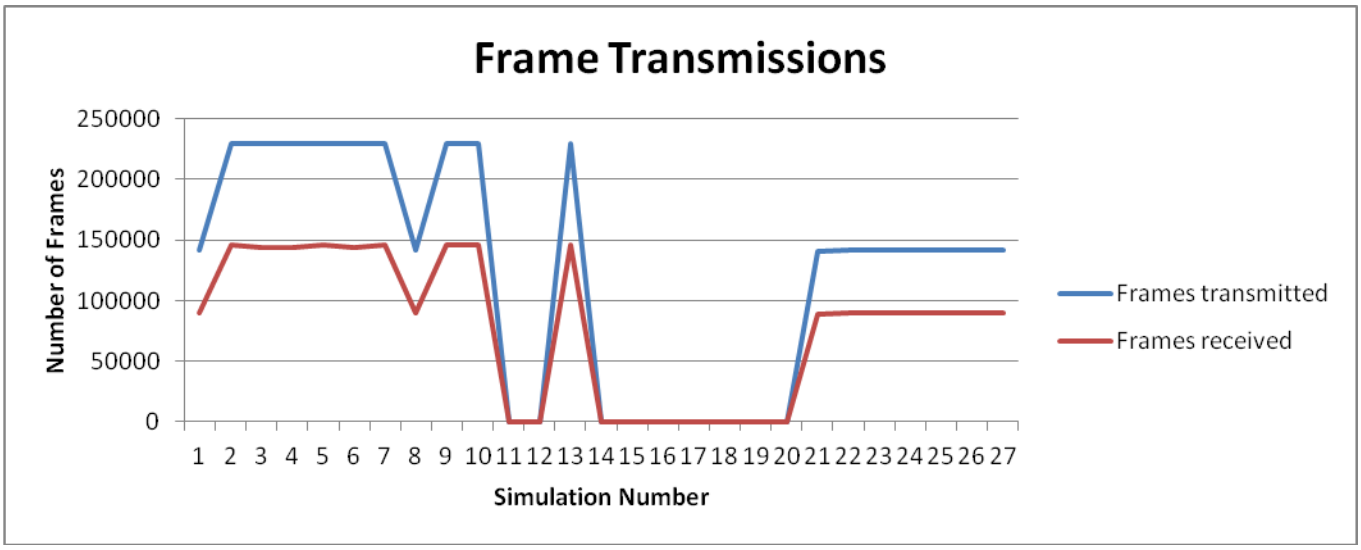


Figure 4.2: : Number of frames transmitted for 2 nodes and 2 access points

Figure 4.1 and 4.2 demonstrates that there is a large number of frames being transmitted with spikes corresponding to simulations with increased message generation rates. This demonstrates that nodes are transmitting frames when necessary. This means that the RTS/CTS is most likely occurring, which allows each node to connect to an access point successfully. The average number of frames being

transmitted per simulation with 10 nodes and 10 access points is 21 784. The difference between the number of frames transmitted and frames received appears to indicate that many collisions are occurring. Figure 4.2 shows that more frames are being transmitted and received when the number of access points and mobile nodes is lowered. This means our network stack implementation is not able maintain high throughput in a larger, more congested network. A notable difference between Figure 4.1 and Figure 4.2 is that in Figure 4.1 the number of frames received is much larger than that of the number of frames transmitted and in Figure 4.2 the number of frames transmitted is greater than the number of frames received, overall (the number of frames received can exceed the number of frames transmitted due to the use of broadcast addresses in some frames (such as Wi-Fi probe frames sent by mobile nodes to identify nearby APs).

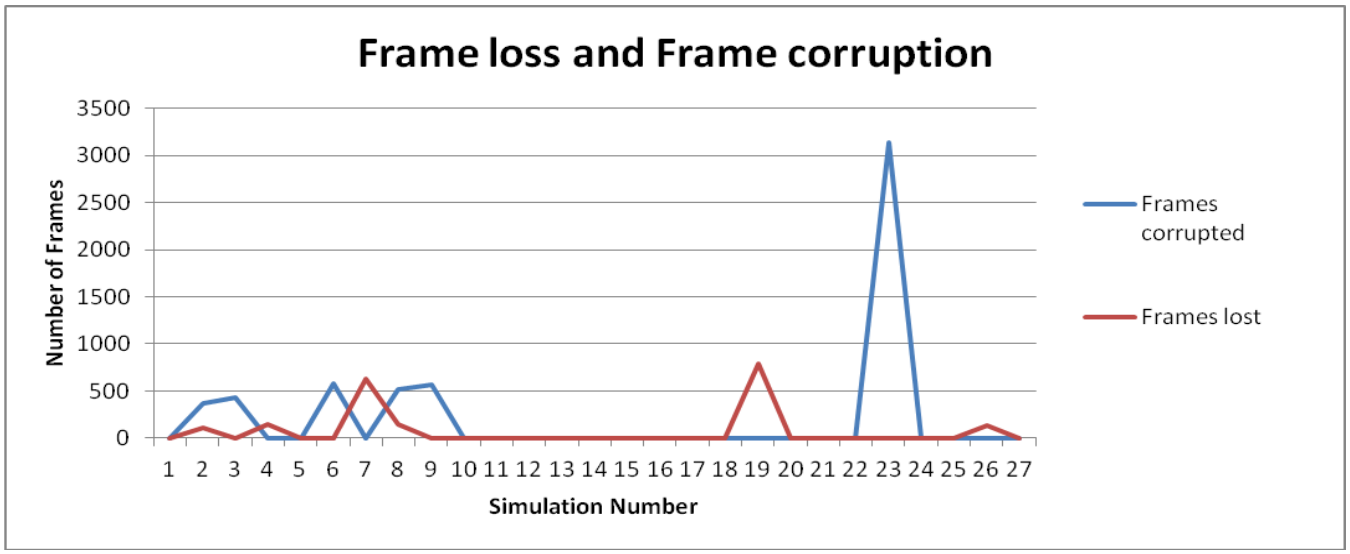


Figure 5: Frame loss and Frame corruption for 5 nodes and 5 Access points

It is also important to note that the number of frames lost, is relatively low (with one unusually high amount for simulation 23). This does not seem to impact message delivery rate since through varying levels of frame corruption and frame loss, message delivery was still largely not occurring. This means there is a problem with the implementation of the protocol stack preventing the network from reliably sending and receiving messages. If the protocols were working as expected, high levels of frame loss and frame corruption would impede performance of the protocols and result in slower delivery, but would not cause non-delivery except in cases in which the network layer sliding window queues were overwhelmed completely (as the network layer should theoretically guarantee eventual delivery with the use of selective-repeat ARQ).

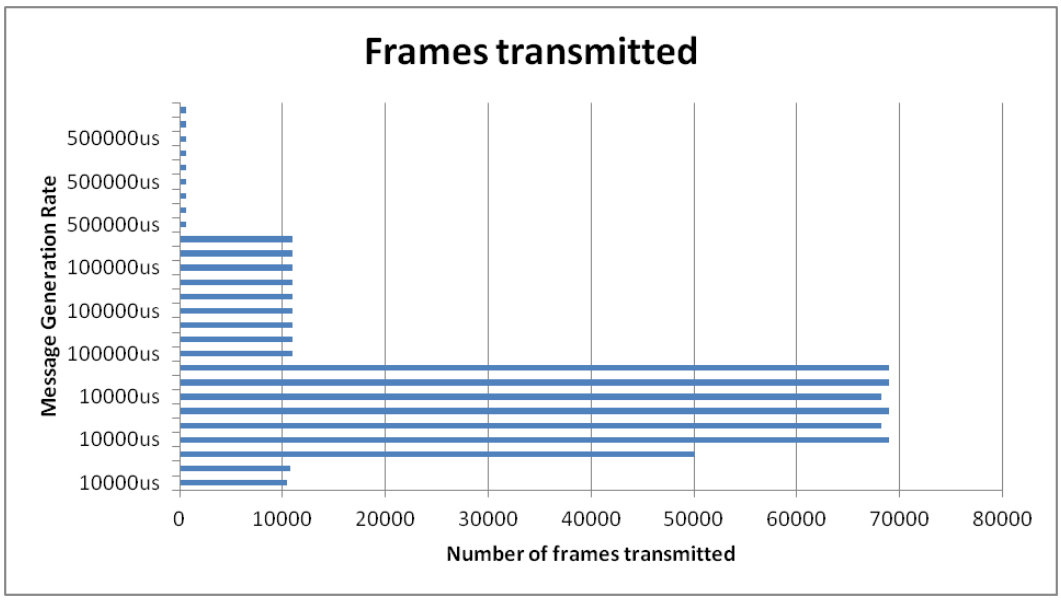


Figure 6.1: Frames transmitted at different message generation rates for 5 nodes and 5 access points

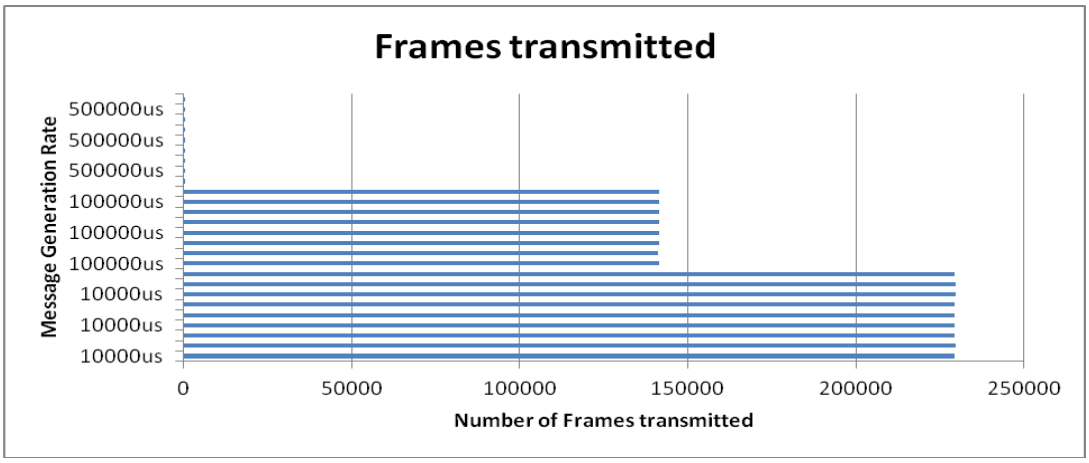


Figure 7.2: Frames transmitted at different message generation rates for 2 nodes and 2 access points

Figures 6.1 and 6.2 demonstrate that as message generation increases, so does the number of frames being transmitted. This occurs especially in regards to the Wi-Fi protocol due to the transmission of RTS and CTS frames for each data frame. As a result, the overall throughput decreases steadily as the number of messages generation rate increases, resulting in a flood of the network. Further results, such as latency and efficiency, cannot be inferred from the number of frames being transmitted due to the lack of delivered messages.

If the protocols were working as expected, during periods of low message generation, we would expect to see few frames being sent. This expectation is confirmed in the Figure 6 when the message generation rate is set to 500 000 microseconds. However the protocols do not handle large amounts of traffic very well. The Ethernet protocol based on CSMA/CD will also have small windows to time of find that channel is clear, causing the buffers in each AP to fill up, and causing congestion throughout the network.

Identification of Problems

After running the above simulations, an attempt to diagnose the problem was made. When the message rate was decreased to 10 000 000 microseconds, a significant proportion of messages were observed being delivered. This would tend to imply that the lack of message delivery in the previous simulations is due to a congestion-related effect. To test this hypothesis, 9 additional simulations were run. The following results will demonstrate that the protocol is able to deliver some messages successfully for reduced message generation rates. The parameters for the simulations that were varied include rate of frame corruption and rate of frame loss: each permutation of these values listed in figure 1.1 was tested. However, for these additional tests, the message generation rate was kept at a constant 10000000 microseconds. The simulations ran for one minute, with 5 mobile nodes and 5 access points.

Results of Problem Identification

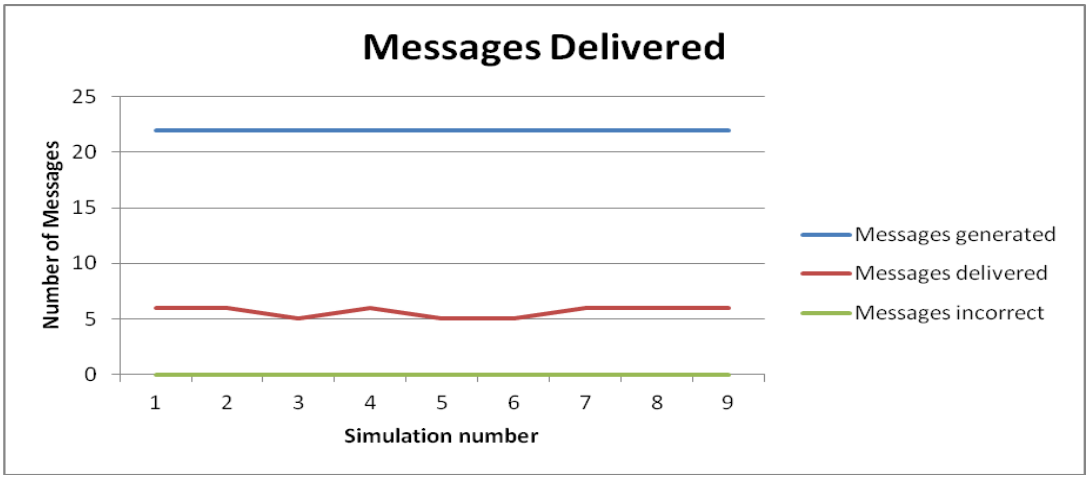


Figure 7: Number of messages delivered for 5 nodes and 5 access points

Figure 7 demonstrates the implemented protocols are at least partly functional. During previous simulation, many issues occurred. Most notably, in the initial with only 2 access points and 2 mobile nodes, mobile nodes were either consistently moving out of range of access points, as simulations were running without enough access points to provide adequate coverage for the simulation map. with an access point, it would never receive an acknowledgement. As a result, mobile nodes would constantly transmit association attempts, slowing simulation execution significantly and resulting in no packet delivery.

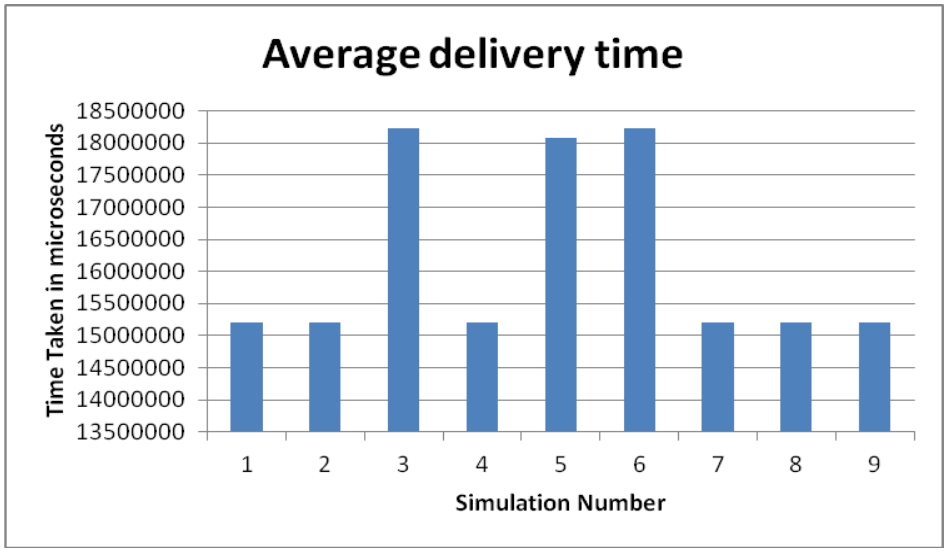


Figure 8: Average delivery time for each simulation under new conditions

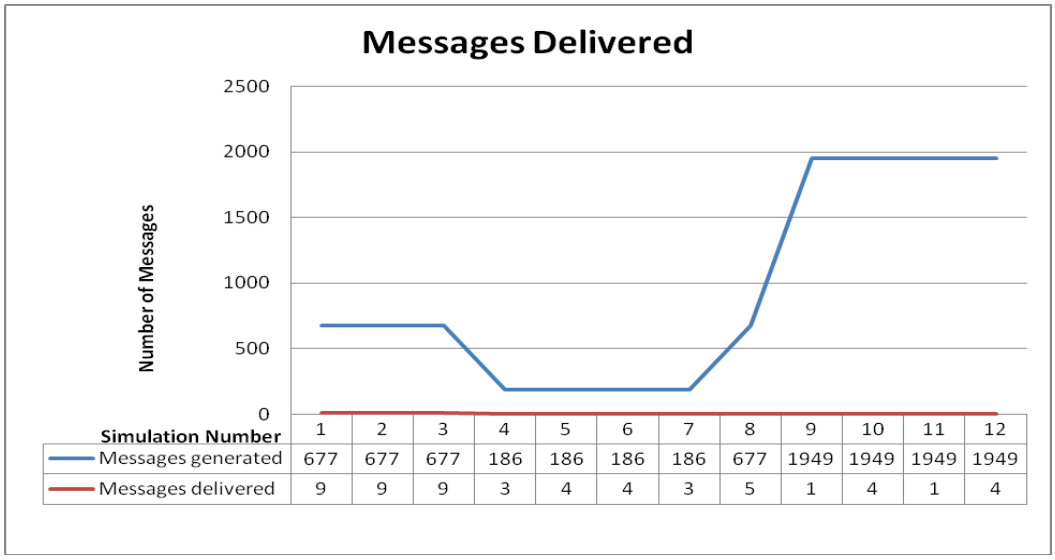
Figure 8 demonstrates the average delivery time within each of the third series of simulations. The sudden peaks in average delivery time correspond to the simulations which delivered 5 messages rather than 6. The median of these averages is 15195524 microseconds. Under a much reduced level of message rate and a greater number of access points, the protocols perform significantly better than earlier simulations. This demonstrates that the implemented protocols very heavily decrease in performance due to network congestion. Frame corruption and frame loss also noticeably affect performance. Simulations 3 and 5 displayed a decrease in message delivery: in both, the number of corrupt or lost frames was 24. In the case of the simulations which were able to deliver 6 messages, the sum of frames corrupted and frames lost was less than 17. Due to a small sample size and relatively small difference between frames, it is not clear whether the number of frames lost or corrupted has a large impact on performance although theoretically, there should be a noticeable decrease in performance where a large amount of frame loss or frame corruption occurs.

Final Simulations

Message Rate (microseconds)	Frame Corruption (as entered into cnet)	Frame Loss (as entered into cnet)
100 000	0	0
300 000	3	3
1 000 000		

Figure 9: Parameters varied during the final round of simulation

When identifying issues with previous simulations message rate was found to be a large limiter on network performance. For this final round of simulations and decreased variation in message rate was considered and due to time constraints, variations of frame corrupt and frame loss were considered. All permutations of variables, shown in figure 9, were used for the last set of simulations, which consisted of testing 5 APs, 5 MNs for a duration of 62 seconds. A layout as in figure 1.2 was used.



Simulation Number	1	2	3	4	5	6	7	8	9	10	11	12
Messages generated	677	677	677	186	186	186	186	677	1949	1949	1949	1949
Messages delivered	9	9	9	3	4	4	3	5	1	4	1	4

Figure 10.1: Parameters varied during the final round of simulation

Figure 10.1 further proves our hypothesis that network congestion is a significant issue for the performance of our protocol. With these decreased levels of message generation, messages are being successfully delivered. Some results from figure 8 are confirmed, that under a much reduced level of message rate and a greater number of access points, the protocols perform significantly better than earlier simulations. This demonstrates that the implemented protocols very heavily decrease in performance due to network congestion. Due to only low levels of frame corruption and frame loss being simulated it the impact of high levels of frame corruption and frame loss cannot be observed.

Theoretically high levels of frame loss and frame corruption should negatively affect performance. An isolated case of this can be observed in regards to simulation 8. Simulation experienced a total loss of 1446 frames and no frame corruption. Simulations 1,2 and 3 all had the same message generation rate as simulation 8, but delivered almost twice the number of messages. The sum of simulation 1 and 3's individual frame corruption and frame loss was approximately 200, with simulation 2's total being 0. This vast increase in simulation 8's frame corruption seems to explain its significantly lower performance. In all other aspects, simulation 8 was practically the same as simulation 1, 2 and 3.

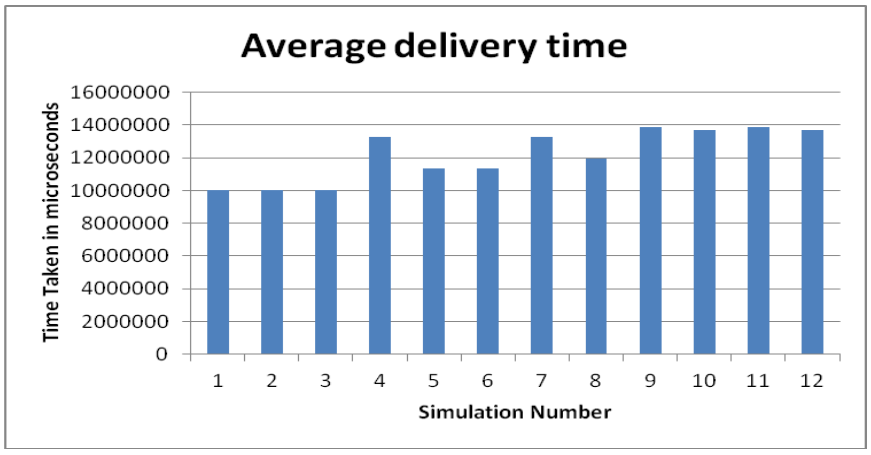


Figure 10.2: Average delivery time for the final round of simulations

Figure 10.2 demonstrates the average delivery time for the final rounds of simulations. There are no especially large peaks noticed in these simulations, suggesting that over time, the messages will be delivered consistently, albeit at a relatively slow rate. As observed in figure 10.1, a significantly higher average delivery time was observed for simulation 8 when compared to simulations 1,2 and 3.

Possible Improvements

Clearly, there are issues with our protocol code which causes the simulated protocols to not function as they were designed to. Furthermore, the code itself is not written in such a way as to minimize the effects of the cnet limitation described on Page 2. Ideally, we would have liked to have rewritten our implementation from the ground up to resolve these issues.

In terms of protocol design, there are some features which could be added to improve the efficiency of the network stack. Frame piggybacking for network-layer acknowledgement could reduce the NL overhead and increase throughput. DLL acknowledgement could be explored, and might possibly yield improvements due to a shorter retransmission path for dropped packets/frames. Currently the RTS/CTS back-off period for the Wi-Fi DLL is constant, but could be intelligently varied with the size of the transmitted frame in order to increase wireless throughput.

Conclusion

Overall, our protocols did not perform as expected: the network was unable to deliver messages successfully under congestion. This is evident in our simulations. Varying the number of nodes and number of access points did not address the issue. Varying rates of frame corruption and frame loss also did not address the issue. In order to improve the protocols, existing bugs and other issues would need to be fixed (ideally, with a complete rewrite). Since the protocol code was not written in such a way as to circumvent the limitations of the cnet simulator, the execution speed was compromised significantly. Simulations were therefore limited to short durations leading to data that was narrow in scope.