Quality of Service Implementation within IEEE 802.11 DCF Interframe Space

David Ehley
Computer Science
University of Wisconsin-Parkside
Kenosha, WI 53144
ehley001@rangers.uwp.edu

Joseph Stewart
Computer Science
University of Wisconsin-Parkside
Kenosha, WI 53144
Stewa045@rangers.uwp.edu

Miguel Estrada
Computer Science
University of Wisconsin-Parkside
Kenosha, WI 53144
estra012@rangers.uwp.edu

Dr. Kamil Samara
Computer Science
Illinois Wesleyan University
Bloomington, IL 61701
ksamara@iwu.edu

Dr. Zaid Altahat
Computer Science
University of Wisconsin-Parkside
Kenosha, WI 53144
altahat@uwp.edu

Abstract

The IEEE 802.11 (WiFi) was designed as a simple and efficient protocol to achieve maximum effort. However, Quality of Service (QoS) was not on the list of major design goals for IEEE 802.11. This design trend combined with multimedia dominating networks traffic, made the IEEE 802.11 face serious challenges. In this research project, we create and test a Differentiated Service Quality of Service (QoS) within the DCF Interframe Space (DIFS) of IEEE 802.11 (WiFi) that can interface with the application layer. Network traffic is divided into three different classes, normal, urgent, and critical. Normal traffic will continue to use the standard DIFS while traffic deemed as critical and urgent could be transmitted faster by using shorter DIFS. Out of the three classes, critical traffic would be transmitted the quickest by using the shortest DIFS. A cross-layer approach was implemented in this research, where the application layer will decide packets classes and the Medium Access Control layer (MAC) will read that information and assign the corresponding DIFS accordingly. This research uses the network simulation framework NS3 to modify the network stack, and simulate traffic in order to compare performance.

1. Introduction

As traffic on a given network can grow exponentially and slow things down, the need to implement a more suitable way as data is being transferred between hosts has never been more important. The quality of service is nothing new in the world of networking, however the need to implement better Quality of Service (QoS) has led to further research on this field. The goal of this research project is to create a quality of service (QoS) with the DCF Interframe Space (DIFS) of IEEE 802.11, being able to modify the mac layer and the application layer as packets are being transferred in a network where a better quality of service can be implemented. To help conduct this research the NS3 network simulator will be used. While the scope of this research is limited to the IEEE 802.11 (WiFi) standard, many other standards have some methods managing services at a low level.

This quality of service would allow an application running on a host system to reserve the wireless medium for a longer period of time, in turn prioritizing traffic from that host. In this particular project, the following three priorities would be implemented: normal, urgent, and critical. Intuitively, normal traffic would be unmodified while traffic deemed as critical and urgent could be transmitted faster. Out of the three priorities, critical traffic would be transmitted the quickest.

In general, networks can be modeled in a couple different ways. For the extent of this project we'll be following the OSI model to model a network.

2. Related Works

QoS approaches for IEEE 802.11 can be categorized into three main groups:

• Differentiated services

• Bandwidth reservation

• Link adaptation

Here, we will be focusing on differentiated services (DS) because it is closely related to our work. In DS, no service guarantee is provided. QoS in DS is achieved by classifying traffic into different classes where higher priority classes are provided advantage over lower ones by giving them preference when competing for wireless access.

The main differentiated services approaches could be summarized as follows:

In *[1]* the authors present a Persistent Factor DCF protocol, in this protocol each traffic class is assigned a persistent factor P where low-priority classes have a greater P and high-priority classes have smaller P. In a back-off stage, a uniformly distributed random

number r is generated in every slot time. Transmission re assumes only if (r > P) in the current slot time.

In *[2]* a Distributed Weighted Fair Queue (DWFQ) algorithm is presented. In DWFQ, the back-off window size of any traffic class is increased to decreased the class's priority and decrease the window size to increase the class's priority.

Distributed Fair Scheduling (DFS): The main idea of DFS is to assign high-priority class a smaller back-off interval which will result is traffic flow with lower back-off interval to transmit first *[3]*.

Quality of service is not a new concept, however creating an interlayer implementation of it is a relatively new idea. Researches from Beijing Jiao Tong University have also studied this approach by aiming to create a multimedia quality of service within the Medium Access Control of 802.11. Similarly to our's, their research was aimed at improving the DCF to be aware of the type of packets being transmitted through the network. By shortening the Interval of Frames, they were able to create different priorities. *[4]*

Another research team from Shanghai Jiao Tong University that was able to improve the 802.11 network stack by integrating TCP acknowledgements into the MAC layer. While their research wasn't concerned with creating a quality of service, they were able to refine interlayer network communications by making the MAC self-aware of Network/Internet Layer information. This was done by incorporating ACK information within the CTS packet at the MAC layer. *[5]*

3. Method

3.1 – Setting up the Simulator

To implement this quality of service, a network simulation tool must be used. For this project, the network simulator NS3 was chosen because of its vast library of documentation and wide range of use. NS3 also supports virtualization, making it easier to use within a virtual environment. While NS3 is a very feature rich simulator and can be used for many different applications, it has many moving pieces that can make it difficult to configure and use. In order to setup and use NS3 it is imperative to have a good working knowledge of Linux/Unix, and a familiarity with C++ and Python. For this research project, Ubuntu 16.04 LTS was used as the underlying operating system. This can be installed natively on a system, or in on a virtual machine with NS3. Prior to installing NS3, it was important to ensure all the dependencies are installed on the system. Most of these dependencies are outlined within the NS3 documentation.

Once NS3 was properly configured and installed, the next step was to look at the source code to determine what modifications needed to be made. While we initially attempted to simultaneously research and modify both the application and MAC layer, we ran into difficulties with NS3's implementation of the application layer. At this point we decided to primarily focus on the MAC layer to build the QoS. This decision allowed us to completely build the quality of service before having an application to interface with it. This process required us to reiteratively sift through documentation, and experiment with code. *[3]*

3.2 – Working with the MAC

Through this process we found the class we needed to modify was the wifi-mac.cc. Within this class we found the source code that set the time delays for slot time, SIFS, RIFS, EIFS, and more. It was interesting to find that each WIFI standard (a,b,g,n,ac) had a different fixed value for these time delays. For example, 802.11a has a slot time of 9 microseconds, while 802.11b has a slot time 20 microseconds. *[4]*

By modifying the slot time when the WIFI standard was being configured, we were able to reduce the time delay between packets being sent. We chose the Slot Time as the figure to modify, because it is directly related when calculating DIFS.

$$DIFS = SIFS + (SlotTime * 2)$$

To improve on this, we came up with a few different algorithms to determine how much we would reduce the slot time. Our first idea was to reduce the time by a percentage, described as follows.

$$Normal\ traffic = (Slot\ Time - 0\%) = No\ Difference$$

$$Urgent\ Traffic = (Slot\ Time - 15\%)$$

$$Critical\ Traffic = (Slot\ Time - 30\%)$$

Our second algorithm aimed to reduce the slot time by a factor of the priority. For example:

$$Normal\ traffic\ was\ set\ to\ 1 :\ default\ priority = (Slot\ Time/1) = No\ Difference$$

$$Urgent\ traffic\ was\ set\ to\ 2 :\ higher\ priority = (Slot\ Time/2)$$

$$Critical\ traffic\ was\ set\ to\ 3 :\ highest\ priority = (Slot\ Time/3)$$

To test these different algorithms, we wrote 2 simulations that establishes a connection between two nodes. This connection is established with each priority to test the

performance difference. The performance is measured by testing throughput, and time for a fixed amount & size of packets.

4. Results

Between these two priority algorithms, we obtained unstable results with the second algorithm. Performance seemed to decrease at every level. However, a different picture came to light when testing the first algorithm. When testing the first algorithm which reduces slot time by percentage, we see a small but noticeable increase between the priorities. We can see this in the figures below.

As we can see in figure one below, modifying the Slot Time does in fact increase the performance when using the first algorithm explained. While this performance increase does net a larger change from priority 1 (normal) to priority 2 (urgent), we still see a positive performance increase from priority 2 (urgent) to priority 3 (critical).

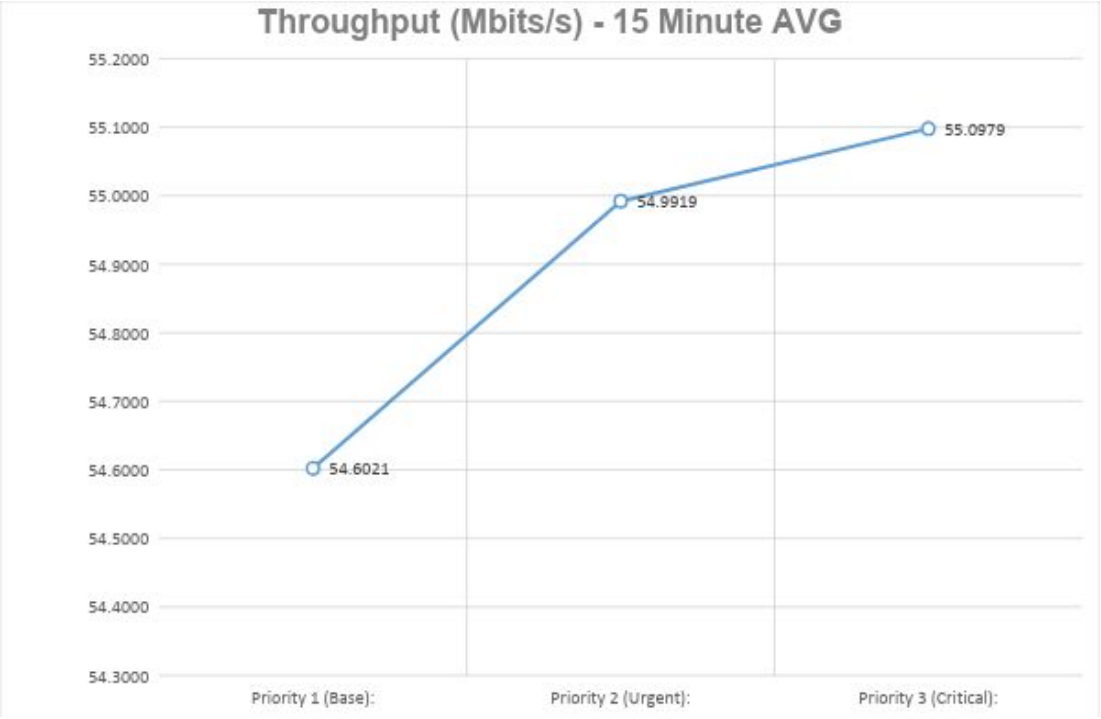

Figure 1: Throughput Test P1->P3

Besides testing throughput, we also tested the amount of time it took to send a fixed number of packets, and saw a similar performance increase. We sent 2048 packets, all of size 4096 bytes. In each advancing priority from priority 1(normal) to priority 3 (critical), we saw a reduction of the time required to send the fixed number of packets. This is demonstrated in figure 2 below.

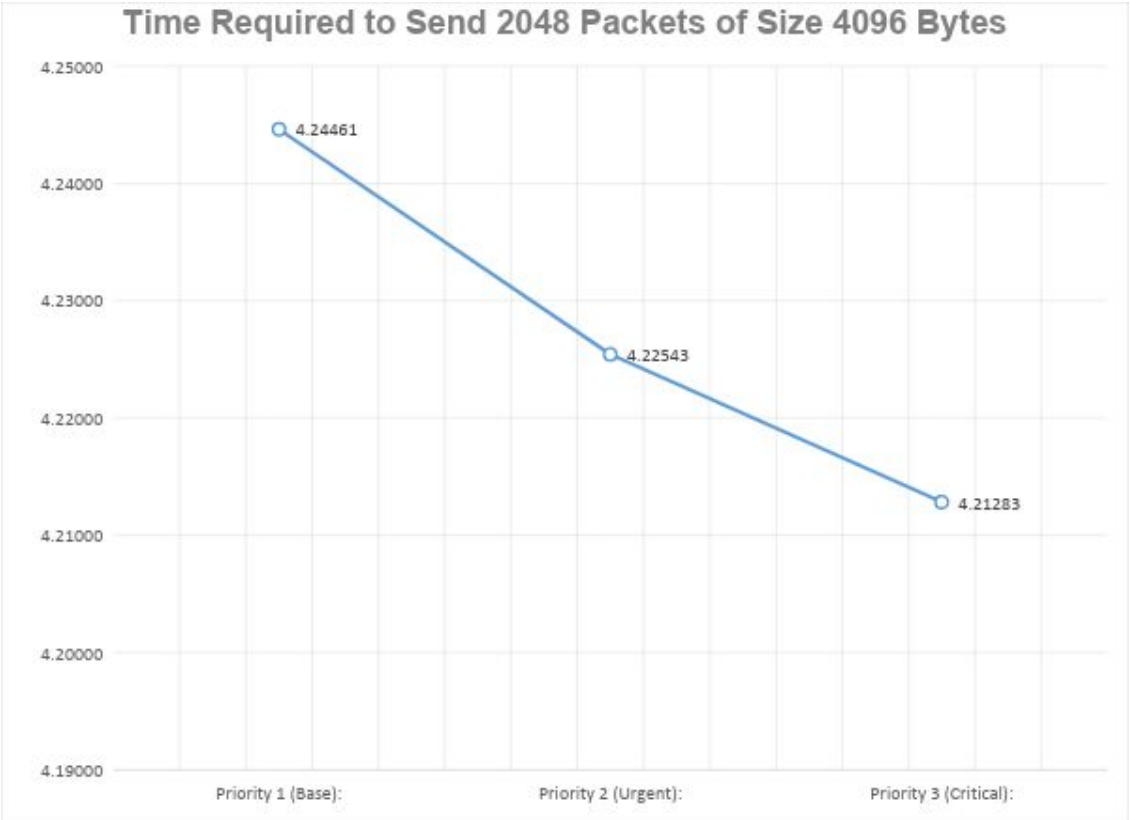

Figure 2: TimeStop test P1->P3

This confirmed that our quality of service within the MAC layer was working as intended. However, for purposes to test the limitations of the quality of service, we decided to create two more pseudo-priorities. These pseudo-priorities were priority 4, and priority 5.

$$Priority\ 4 = (Slot\ Time - 50\%)$$

$$Priority\ 5 = (Slot\ Time - 75\%)$$

When running the same tests with these new additional priorities we saw an interesting result. It seemed that packets were still being transmitted back and forth, but after a short duration packets would start colliding with each other causing a chain reaction of failed transmissions. A packet collision occurs when there is not enough time in between packet transmission. This is very similar to when construction causes a road to become one lane, and traffic from both sides need to travel down the same road. Typically a construction

worker will hold up a stop sign, and let traffic flow in one direction. After a given amount of time, the construction worker will stop the flow of traffic, and start letting traffic flow in the other direction. A packet collision is similar to letting traffic flow from both directions at the same time.

This phenomenon can be demonstrated in the figures below. Not these two tests are the same, with the exception of the additional two priorities added. The values for priorities 1→3 remain the same.

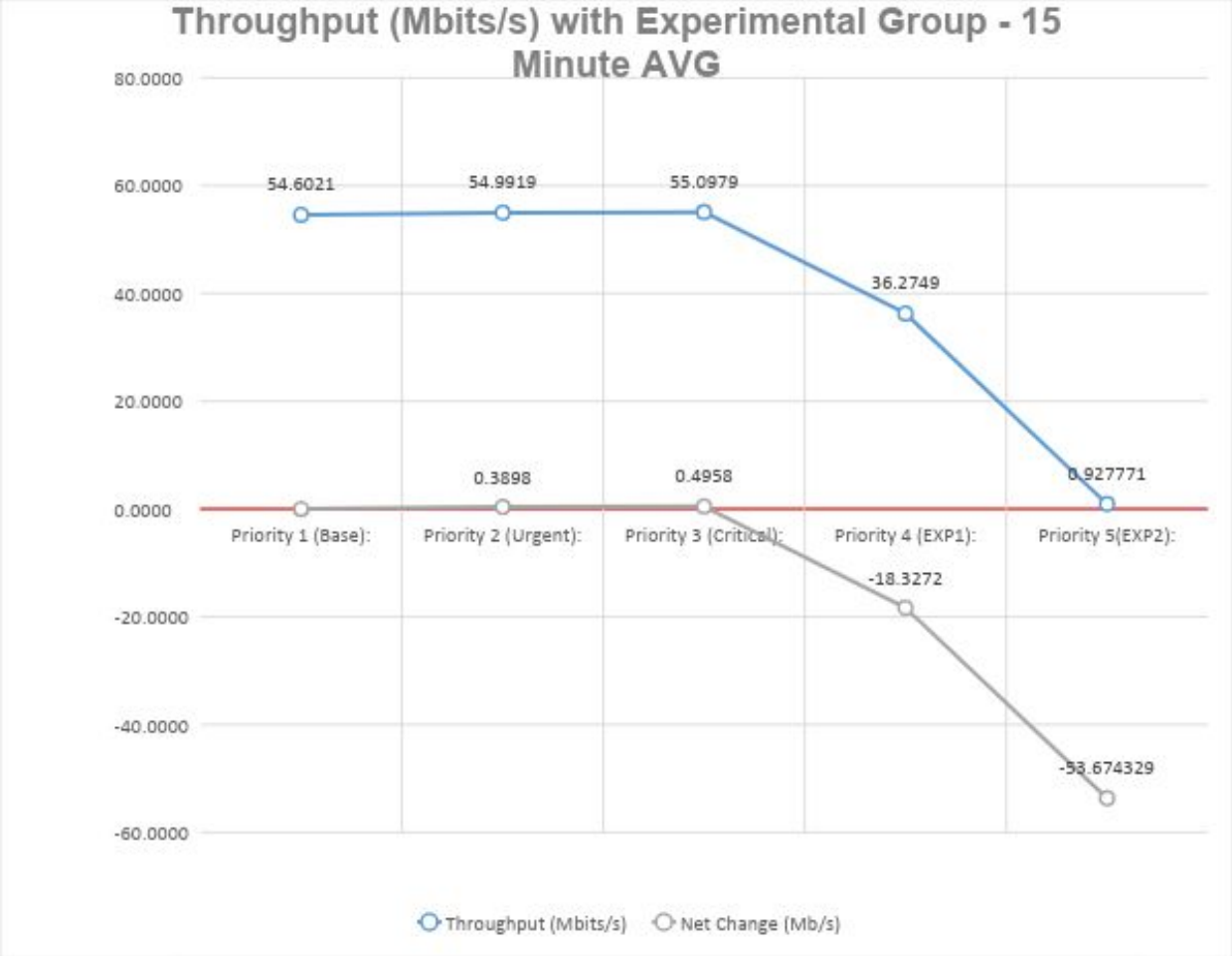

Figure 3: Throughput Test P1->P5

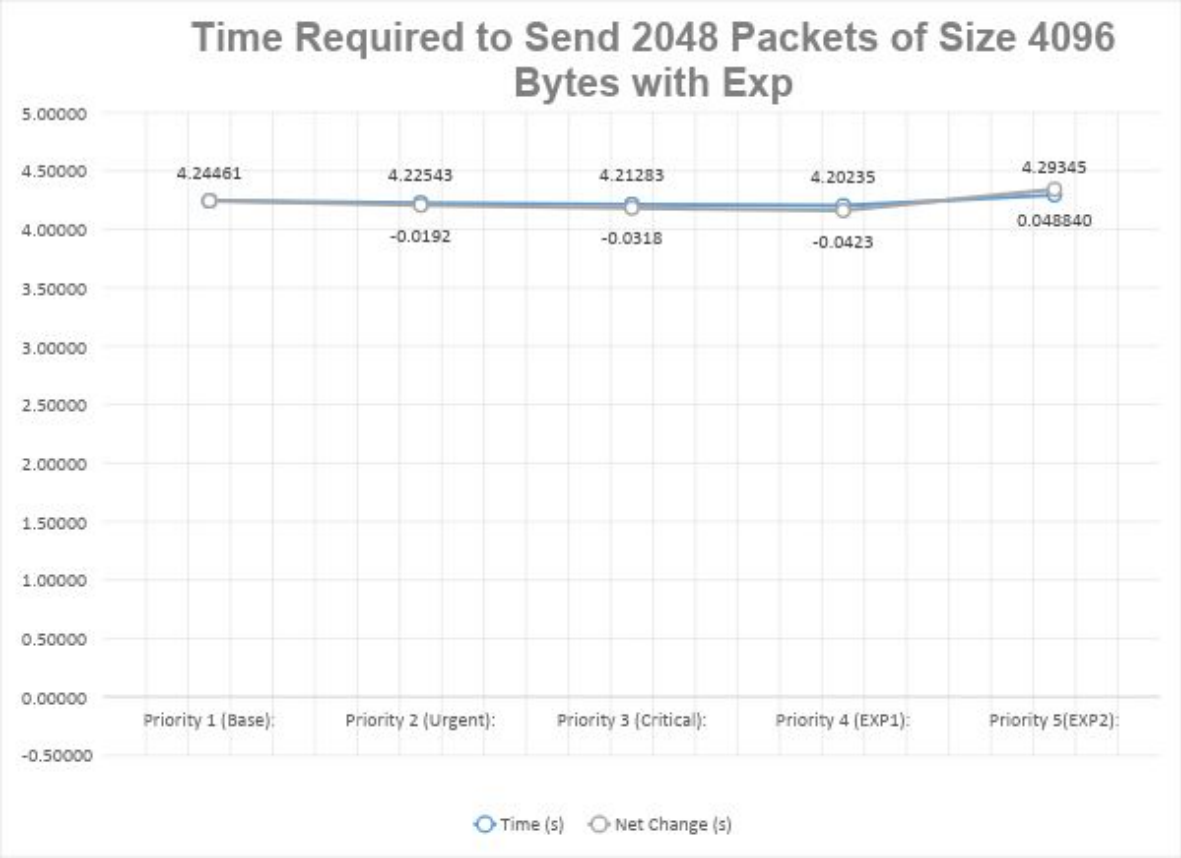

Figure 4: TimeStop Test P1->P5

Looking at these two tests, we can see the immediate performance decrease. In the throughput test, this packet collision occurs roughly after 6 seconds meaning that a reduction of greater than 50% slot time is not sustainable. Also note that in the TimeStop test, all tests are running less than 4.3 seconds. For that reason, Priority 4 still shows as a performance increase in the TimeStop test.

Throughput Test - 15 Minuage AVG - 4096 Bytes/Packet			
Priority Level	**Modifier**	**Throughput (Mbits/s)**	**Net Change (Mb/s)**
Priority 1 (Base):	0% SlotTime reduction	54.6021	0
Priority 2 (Urgent):	15% SlotTime reduction	54.9919	0.3898
Priority 3 (Critical):	30% SlotTime reduction	55.0979	0.4958
Priority 4 (EXP1):	50% SlotTime reduction	36.2749	-18.3272
Priority 5(EXP2):	75% SlotTime reduction	0.927771	-53.674329

Timestop Test - 2048 Packets - 4096 Bytes/Packet			
Priority Level	**Modifier**	**Time (s)**	**Net Change (s)**
Priority 1 (Base):	0% SlotTime reduction	4.24461	0
Priority 2 (Urgent):	15% SlotTime reduction	4.22543	-0.0192
Priority 3 (Critical):	30% SlotTime reduction	4.21283	-0.0318
Priority 4 (EXP1):	50% SlotTime reduction	4.20235	-0.0423
Priority 5(EXP2):	75% SlotTime reduction	4.29345	0.048840

Figure 5: All tests

5. Future Work

To further advance this research project, one could investigate how to implement this quality of service within the network and application layer, allowing an application to set a bit within the packet header designated for this QoS. In turn unraveling the packet at the MAC layer to set the transmission priority. While this research is ongoing, there are many different ways to implement such a quality of service. It is our hope that others can use the discussed approach to design more efficient and agile networks.

6. Conclusion

In conclusion, while we continue to research different ways to integrate this quality of service at the network and application layer, we were able to create the foundation for a quality of service that could later be used by the application layer. This quality of service aims to improve the communication across a network. After analysing the results, we determined that the designed quality of service is sustainable so long that the slot time is not reduced beyond 50%. While doing so may temporarily increase performance, it is unsustainable for any practical use.

7. References

[1] Y. Ge and J. Hou, "An Analytical Model for Service Differentiation in IEEE 802.11," *IEEE ICC '03, vol. 2, May 2003,* pp. 1157–62.

[2] A. Banchs and X. Pérez, "Distributed Weighted Fair Queuing in 802.11 Wireless LAN," *IEEE ICC '02, vol. 5, Apr. 2002,* pp. 3121–27.

[3] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN," *Proc. ACM MOBICOM 2000, Aug. 2000,* pp. 167–78.

[4] H. Tian and W. Yang, "Study on QoS of Multimedia Traffics in MAC Layer Based on 802.11," *2012 International Conference on Information Science and Applications*, 2012.

[5] L. Ding, W. Zhang, H. Yu, X. Wang, and Y. Xu, "Incorporating TCP Acknowledgements in MAC Layer in IEEE 802.11
Multihop Ad Hoc Networks," *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, 2009.

[6] "ns-3 Tutorial," *ns-3 Tutorial - Tutorial*. [Online]. Available: https://www.nsnam.org/docs/tutorial/html/index.html.

[7] ns-3: ns3::WifiMac Class Reference. [Online]. Available: https://www.nsnam.org/doxygen/classns3_1_1_wifi_mac.html