A NEW DESIGN OF Q-CHAIN ON WIMAC: A THROUGHPUT-OPTIMAL RANDOM ACCESS MAC PROTOCOL WITH PIGGYBACK TRANSMISSIONS

A Thesis

by

YUE YANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	I-Hong Hou
Committee Members,	Jiang Hu
	Anxiao (Andrew) Jiang
Head of Department,	Miroslav Begovic

December 2018

Major Subject: Computer Engineering

ABSTRACT

While IEEE 802.11 DCF has been the dominant protocol used in the existing WLANs, it suffers the low efficiency from the increasing devices connected to the network. The more nodes under one AP, the more collision it will happen in the network.

To reduce the overhead contention issue, we propose a new design of Q-CHAIN. The original Q-CHAIN achieves a MAC protocol using piggyback transmissions, which can update the piggyback relation on the client side independently. However, it is not robust under unreliable data transmission and still needs an intelligent AP to decode special packets and transmit special ACKs. The new design of Q-CHAIN can handle unreliable data transmission. It can be cooperated with the legacy DCF and inherit all advantages of the original Q-CHAIN. Moreover, it does not need an intelligent AP and can be integrated into the existing network in a friendly way. We implement the new design of Q-CHAIN on WiMAC, and the experiment results show the performance of Q-CHAIN with the legacy DCF and the legacy CHAIN.

DEDICATION

This is dedicated to my mother, my father, who love me, believe in me, inspire me and have supported me every step of the way.

ACKNOWLEDGMENTS

First of all, I wish to thank Dr. I-Hong Hou, my supervisor, for supporting me to finish the research. It was a wonderful learning experience under him. He helped me a lot on the research and also taught me a serious attitude beyond the study. I would also like to thank Dr. Ping-chun Hsieh for helping me to understand the algorithm and finish the experiment. The solutions presented in the thesis were a direct benefit from my discussions with him. He was very enthusiastic and knowledgeable.

Moreover, I want to thank Dr. Jiang Hu and Dr. Anxiao (Andrew) Jiang for serving as my committee members and being always reachable for help.

CONTRIBUTORS

Contributors

This work was supported by a thesis committee consisting of Dr. I-Hong Hou and Dr. Jiang Hu of the Department of Electrical and Computer Engineering and Dr. Anxiao (Andrew) Jiang of the Department of Computer Science and Engineering. All work for the thesis was completed by the student, under the advisement of Dr. I-Hong Hou of the Department of Electrical and Computer Engineering.

Funding Sources

Graduate study was supported by a scholarship from the Department of Electrical and Computer Engineering.

NOMENCLATURE

ACK	Acknowledgement
DCF	Distributed Coordination Function
CSMA/CA	Carrier-sense multiple access with collision avoidance
AP	Access Point
WLAN	wireless local area network
CHAIN	Coordinated Heavy-traffic efficient Access scheme
MAC	Media Access Control Layer
DIFS	DCF Interframe Space
SIFS	Short Interframe Space
CW	Contention Window
TX	Transmit
RX	Receive
CCA	Clear Channel Assessment
PHY	Physical Layer
MCS	Modulation and Coding Scheme
QAM	Quadrature Amplitude Modulation
PN	Pseudo Noise
IEEE	Institute of Electrical and Electronics Engineers

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

In WLANs, IEEE 802.11 standard is the most popular technology nowadays. Distributed Coordination Function (DCF), which implements Carrier-sense multiple access with collision avoidance (CSMA/CA) with binary exponential backoff algorithm, is one of the fundamental MAC techniques of this standard. DCF has lots of advantages as it is simple, robust, lightweight and has been widely used in various amendments of 802.11 standards. However, DCF needs to consider significant waste. One is from the random access backoff time, and the other is from the unavoidable collisions.

The low efficiency of DCF will happen in two conditions[1][2]. First, if a network system handles a large number of access points (AP), it will increase the collisions in the system. It's very common in existing wireless local area networks (WLANs). Think about in one room, each person has a cell phone, a laptop, an electronic watch, all of them will become the nodes which are connected under one AP. WLANs will become more and denser during the rapid development of technology. Second, if the packet size of the nodes is always small, it will use little time to transmit. Then the backoff time will cover most of the time in the channel, which is always idle. Therefore, the overhead issue of DCF cannot be ignored. Improving the efficiency of DCF is necessary.

To improve the efficiency of DCF, lots of researchers focus on two goals: (1) Reduce collisions. (2) Get a reasonable backoff time and reduce the idle time of the channel as much as possible. Most of the existing studies to improve the efficiency of DCF focus on two research directions:

(1) **Improve the backoff algorithm.**

Kim and Hou propose a scheduling schema MFS[3]. In MFS, each client could calculate the utilization of current work using an estimation from counting the number of collisions between two consecutive frame transmissions. In [4], CalÃň et al. consider to find an optimal p-persistent model according to check the status of the network system, then get the appropriate contention window size. The results of these studies show a better performance than the original DCF. However, they cannot avoid the severe collisions in a dense WLANs.

(2) **Coordinate the transmission for the nodes in MAC**

Muir and Garcia-Luna-Aceves propose GAMA-PS in [5], which allow all nodes in the network organizing into cycles. In each cycle, all nodes can transmit the data packet one by one. It reduces the collision in the network. However, the nodes in GAMA-PS cannot work with the nodes that run the existing DCF. Xiao [6] applies for a concatenation mechanism, to connect multiple frames and transmit the multiple frames at one time. This way sacrifices the processing time to concatenate frames. Also if the transmitted package is big, it will cause a severe package delay.

This research is based on the Coordinated Heavy-traffic efficient Access scheme (CHAIN) protocol in [7], which is to implement a piggyback mechanism according to overhear ACKs. In CHAIN protocol, each client is assigned a predecessor, and it could transfer one packet after overhearing the ACK from its predecessor without waiting for the backoff time. The clients can be connected like a chain because each of them has a unique predecessor, and they could transmit one by one. Moreover, CHAIN could adjust a modified exponential backoff scheme based on whether the client faces a massive collision or not. This mechanism could significantly improve the efficiency of DCF, and the simulation result is in [7]. However, the disadvantages of CHAIN cannot be ignored, as it needs an intelligent AP to broadcast the piggyback relation.

In the lab, Yau has implemented parts of the CHAIN on WiMAC [8], a wireless testbed to implement real-time MAC protocols for WLANs. Each client is assigned a unique predecessor. It can transmit under two conditions: (1) Overhearing the ACK from its predecessor, it will transmit one data packet immediately after SIFS, in other words, each client does not need to wait for any backoff time in this condition. (2) When the client does not overhear the ACK contains its predecessor's MAC Address, it will run the basic DCF. In Yau's work, if all the clients in the network system are connected as a chain loop, without considering the loss of packages, the system will get an optimal throughput. However, the optimal throughput only happens in an ideal condition. The legacy CHAIN cannot support the dynamic network environment and it cannot be incorporated with the legacy DCF.

Hsieh et al. [9] present a new MAC protocol with piggyback transmissions based on the existing DCF. It achieves optimal throughput for the clients under Q-CHAIN, and it does not need AP to broadcast the piggyback relation. However, it also meets a vital problem under unreliable data transmission (Packet Loss), and it needs an intelligent AP to decode the dummy packet and transmit the ACK with special meaning.

This thesis proposes a new design of Q-CHAIN handles the unreliable data and keeps all advantages of Q-CHAIN. Moreover, the new idea of Q-CHAIN does not need any dummy packets and intelligent AP, so our protocol can be updated into the existing DCF system in a friendly way. Moreover, we implement the new Q-CHAIN on WiMAC.

The rest of the thesis is organized as follows. Section 2 provides the network model. Section 3 introduces the background of the research. The new design of Q-CHAIN is described in Section 4. Section 4 discusses the implementation of Q-CHAIN on WiMAC. Section 5 shows the experimental results. Finally, Section 6 concludes the thesis.

2. NETWORK MODEL

The wireless ad hoc network system contains one Access Point running DCF and multiple clients. Each client in the system either runs DCF or Q-CHAIN.

Consider each client has one packet stream with a transmitted packet queue. If two or more clients transmit the packets to AP at the time, the collision will happen and this transmission will fail. If not, the transmission is successful, and the AP will send an ACK that contains the mac address of the transmitted client immediately after SIFS.

The ACK in the network model is reliable, and the data cannot be seen as reliable. It means if the AP transmits an ACK, all the clients (either DCF or Q-CHAIN) can overhear it. However, sometimes the AP will not receive the packet although no collision happened in the network. This condition is called Packet Loss.

Q-CHAIN is built on the top of DCF. The contention process of the client running Q-CHAIN is the same as the one running DCF. Both of them contain DIFS period and backoff period before they transmit. Q-CHAIN could skip the backoff period when it does piggyback transmission, overhearing the ACK with its piggyback address.

Under Q-CHAIN, each client will follow at most one predecessor client, so it is a single, one-sided chain. The status of the clients under Q-CHAIN can be divided into in-chain and out-of-chain. Before joining into the chain, the clients run DCF to contend the channel. Once it has been transmitted successful, it will join into the chain.

3. BACKGROUND

3.1 Distributed coordination function (DCF)

3.1.1 Overview of DCF

This section provides an overview of the three significant related works: DCF, CHAIN, Q-CHAIN. It also refers to the disadvantages of them.

IEEE 802.11 DCF has been employed in the existing WLANs widely. It utilizes CSMA/CA with binary exponential backoff algorithm for random access. Figure 3.1 describes the mechanism of DCF.

1. When Client A and Client B have the packets to transmit, they will be required to sense the channel for the DCF Interframe Space(DIFS) time. If the channel continues idle during DIFS, the client should go into the backoff period.

2. Then Client A and Client B will go into the backoff period. First, they need to get a random integer number in the CW (Contention Window). In the IEEE 802.11a protocol, the default CW is 15, so the range of the backoff value is from 0 to 15. In Figure 3.2, the backoff value of Client A is 5, the backoff value of Client B is 9.

3. During the backoff period, the Client will sense the channel after each slot time. If the channel is idle, the counter of the backoff value will counts down. In Figure 3.1, after five slot times, the backoff value of Client A counts down to 0 and the backoff value of Client B counts down to 4.

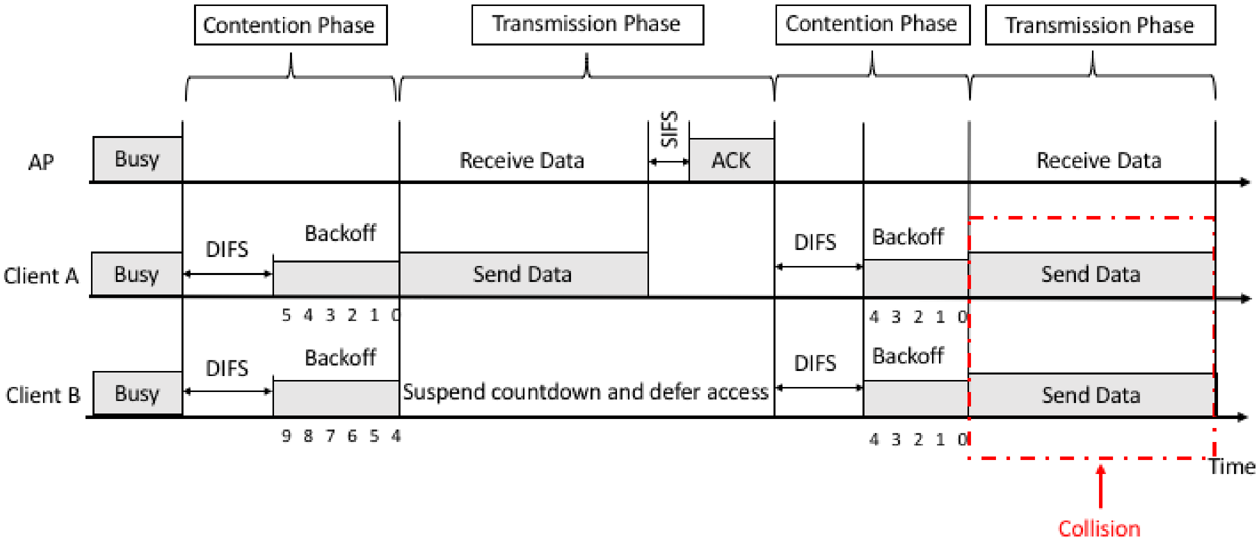


Figure 3.1: IEEE 802.11 Distributed Coordination Function (DCF)

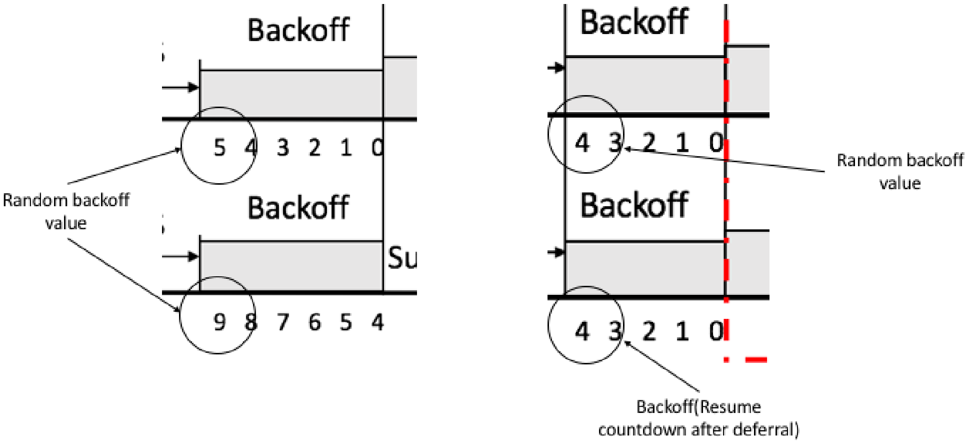


Figure 3.2: Backoff value in DCF

4. When the backoff value of the client counts down to 0, this client contends for the channel, then transmits the packets. In Figure 3.1, Client A transmits the packet to Access Point (AP), when AP receives the packets, it will send an ACK to all clients after Short Interframe Space (SIFS).

5. If Client A receives the ACK from AP, it would finish one round-trip transmit.

6. After one transmission, all clients need to sense an idle DIFS period again, then go into another backoff period. If the client has transmitted last time, it needs to get a new random backoff value from CW. If not, it should resume the backoff value after the last deferral. In Figure 3.2, the Client A chooses 4 as the backoff value, and Client B resumes the backoff value 4 in the last contention phase.

7. If two or more clients finish the backoff period at the same time, they would transmit together, and the collision will happen. The collision is happening in Figure 3.1. In the next backoff period, both of the clients who meet the collision should use the binary exponential backoff algorithm to enlarge CW and get a new backoff value.

3.1.2 Drawbacks of DCF

IEEE 802.11 standard has been the most dominant technology globally, the wireless channels have been much denser and denser. During a crowded event, the measurement in [10] shows the developing networked devices and new applications leading to a massive amount of uplink traffic. The more nodes in the networks, the probability of the nodes have the same backoff value is higher and the more collision it will happen. Therefore, the increasing contention overhead issue of DCF cannot be ignored. Moreover, although DCF uses the binary exponential backoff algorithm to enlarge CW to reduce the collisions, the larger CW will lead to a waste of the channel during the backoff period, which will reduce the efficiency of MAC.

3.2 CHAIN

3.2.1 Overview of CHAIN

CHAIN[7] is based on DCF, and it reduces the backoff period by using piggyback transmissions. It means each CHAIN client follows a predecessor client, and it will transmit immediately after SIFS when overhearing the ACK with the MAC Address of its predecessor client. The client under CHAIN skips the contention phase so that it can reduce collisions in the channel. Under CHAIN, the AP should periodically broadcast the relationship of piggyback table based on the average throughput of each client under CHAIN so that the piggyback table can be updated and kept consistently in all clients. Moreover, the system needs an intelligent AP to achieve CHAIN protocol.

3.2.2 Piggyback transmission

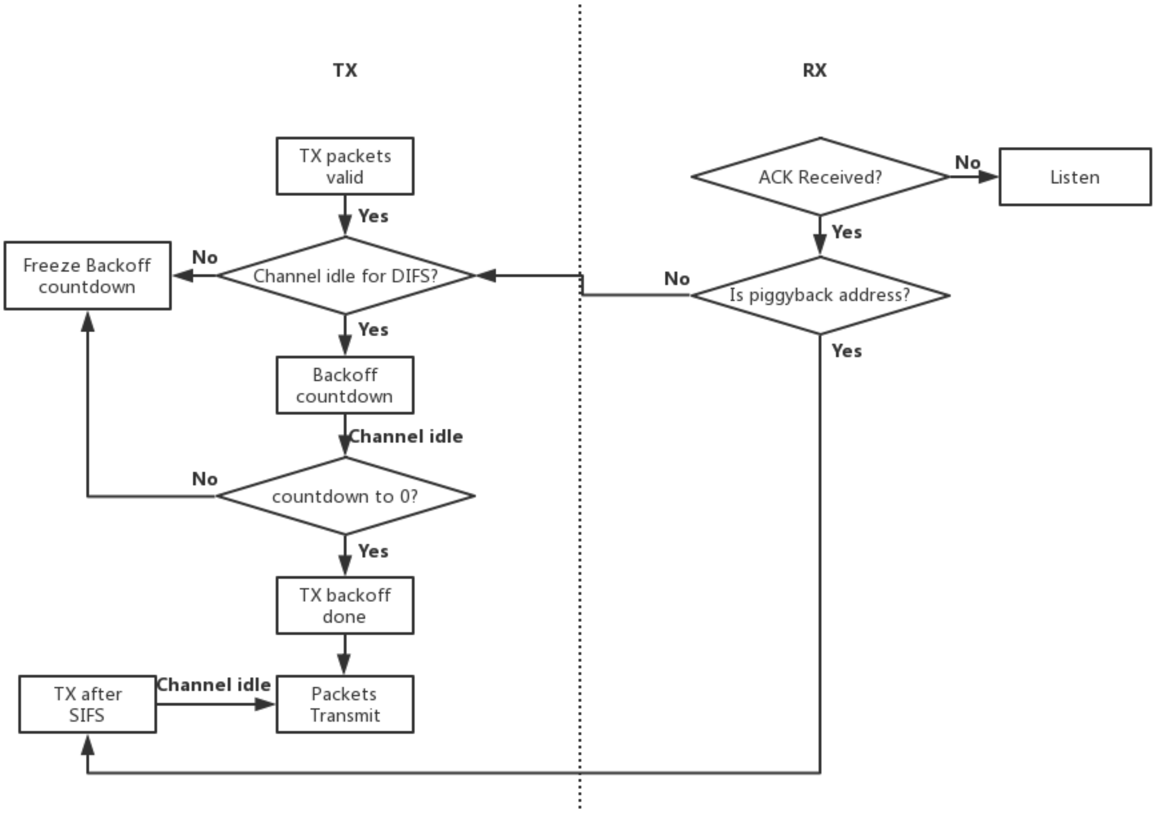


Figure 3.3: The flow of piggyback transmission

Each client contends the channel independently under DCF. While the network system becomes more and more congested, the collision will increase apparently. Could we build a relationship between a group of clients, to let them share the opportunity of contending the channel with other clients? It is the basic idea of piggyback transmission.

The client under CHAIN can either transmit via DCF or transmit via the piggyback transmission. The process of DCF has been mentioned. About the piggyback transmission, if the client overhears the ACK contains its predecessor client's MAC Address (its predecessor client has been transmitted successfully), this client will skip the contention phase and transmit immediately after sensing an idle SIFS. Figure 3.3 describes the flow of piggyback transmission. Each client at most has one unique predecessor client under CHAIN. It cannot exist two or more clients to do the piggyback transmission at the same time. Hence, it will decrease most of the collisions in the network system. Table 3.1 provides the piggyback relation of Figure 3.4 shows the scheme of piggyback transmission. Under piggyback transmission, the timeline contains one contention phase with lots of continuous transmission phase. Under DCF, the timeline of each client should be the one contention phase with one transmission phase. Therefore, another advantage of piggyback transmission is to reduce lots of idle contention phases in the channel.

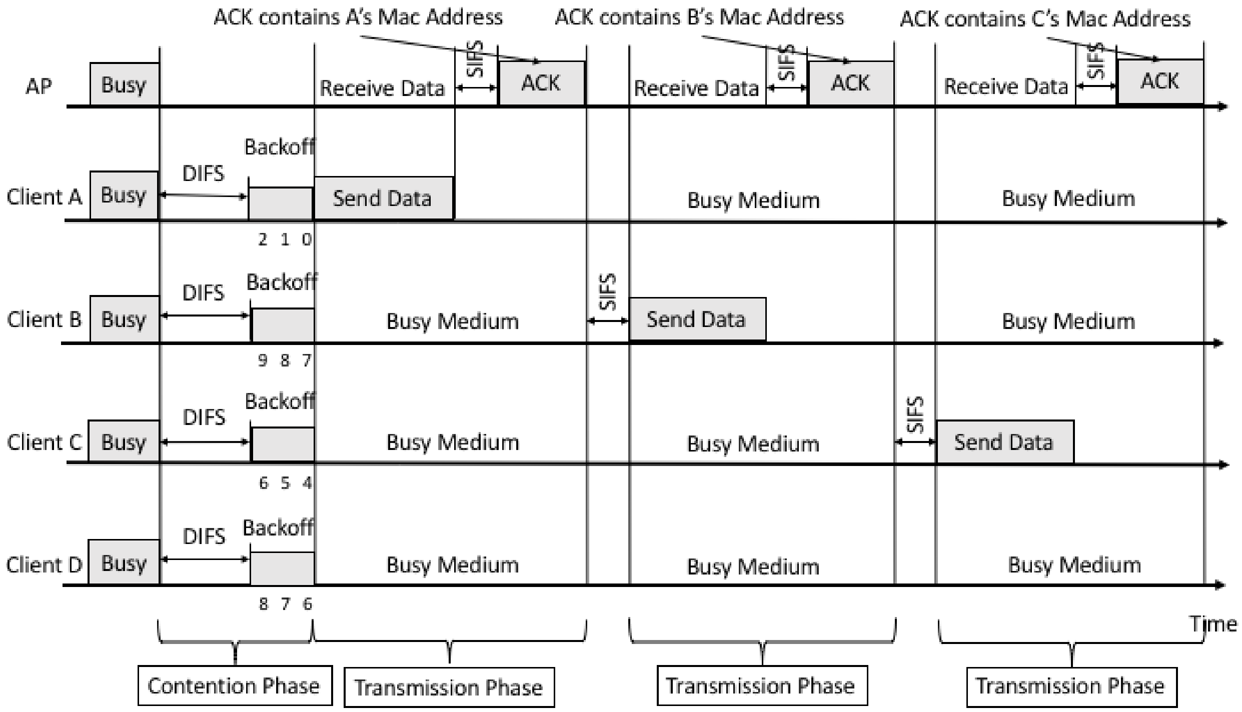


Figure 3.4: The diagram of piggyback transmission

Client Address	Piggyback Address
A	D
B	A
C	B
D	C

Table 3.1: Piggyback Table of CHAIN

3.2.3 Drawbacks of CHAIN

There are two major drawbacks of CHAIN:

(1) The performance of CHAIN is wholly based on the piggyback relation, which is periodically updated by calculating the average throughput of each client. If the throughput of the client changes quickly, the piggyback relation will break frequently. Therefore, the design of CHAIN only achieves sub-optimal throughput.

(2) The piggyback relation is maintained on the AP side, then via broadcasting to let all clients know. It requires considerable implementation on the AP side. It is not a good idea to promote a new protocol to the existing networks.

3.3 Q-CHAIN

3.3.1 Overview of Q-Chain

Hsieh et al. propose Q-CHAIN, a MAC protocol built on the top of DCF for random access with piggyback transmissions. Under Q-CHAIN, each client can join and withdraw from the piggyback relation independently by overhearing the ACKs from the AP. It does not need to synchronize the clients in the network system. Also, most of the implementation of Q-CHAIN is built on the client side, and it does not need the AP to broadcast the piggyback relation. Moreover, Q-CHAIN accomplishes throughput-optimality without the prerequisites of contention time or traffic situations.

3.3.2 Join Mechanism of Q-CHAIN

Candidate Table
A
B
C
D
E
F
...

Table 3.2: The candidate table in Q-CHAIN

Under Q-CHAIN, each client should know which MAC Address(Table 3.2) could be able to join into the chain.

Figure 3.5 shows the Join Mechanism of Q-CHAIN, A, B, C three clients are the candidates who can join into the chain. Table 3.3 describes the chain table of A, B, C three clients.

(1) At the beginning of the network, no client has transmitted. All clients' chain tables are empty.

(2) When A joins into the system, A's chain table will update to A. The piggyback address of Client A is also A.

(3) When B joins into the system, A will overhear the ACK contains B's MAC Address. After that, a will check if the address is in the candidate table, if so, execute piggyback transmission after SIFS, then update the chain table to BA. If not, use legacy DCF to contend the channel after the next idle DIFS period. For client B, after it has transmitted successfully, it will overhear

the ACK with A's MAC Address before the next idle DIFS. So B considers A is the client in the chain behind itself.

(4) When C joins into the system, the process is same with step (3). The chain will transmit once between two idle DIFS periods. All clients will know the chain has been transmitted once when the channel is idle during DIFS. At this time, the first one (we call head) in the chain will use the MAC Address of the last one in the chain table as the piggyback address. So the chain is seen as a circle.

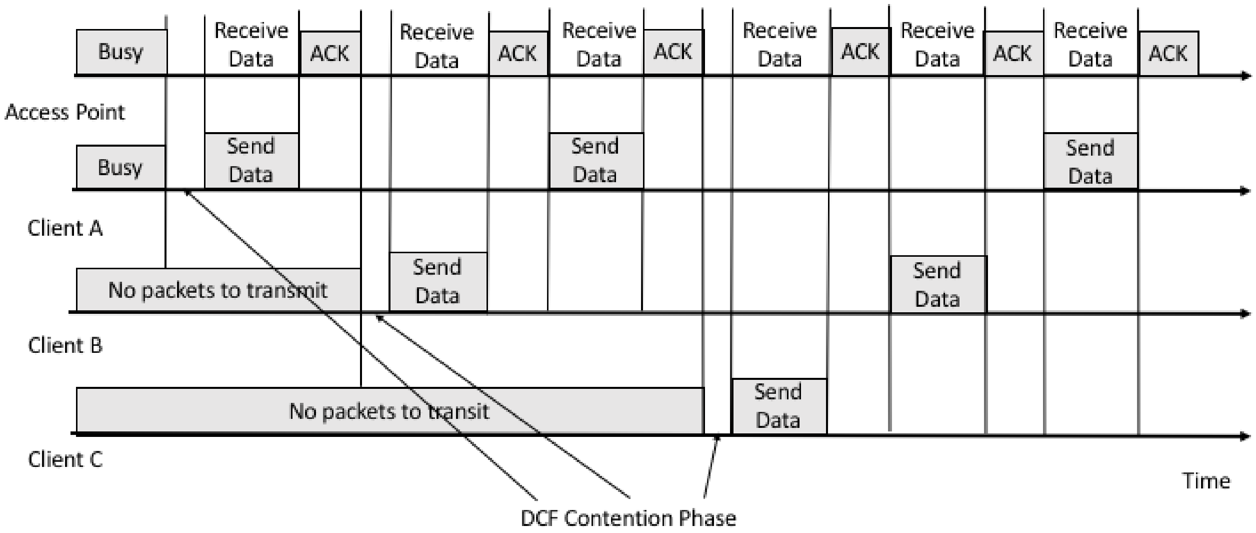


Figure 3.5: Join mechanism of Q-CHAIN

Client Condition	A	B	C
Initial	∅	∅	∅
A joins into the chain	A	∅	∅
B joins into the chain	BA	BA	∅
C joins into the chain	CBA	CBA	CBA

Table 3.3: The Chain Table under Join Mechanism

3.3.3 Withdraw Mechanism of Q-CHAIN

Figure 3.6 shows the original Withdraw Mechanism of Q-CHAIN, the client will transmit a one-time dummy packet if it does not have packets to transmit (the size of transmitted queue does not reach the threshold). Then the AP will decode the dummy packet and transmit an ACK contains the info that this client has withdrawn from the chain. When other clients receive this ACK, they will check the order of the withdrawing client in the chain table, reset the order of it to zero and advance the clients behind it. Table 3.4 describes how the chain table will change when the client has withdrawn from the chain.

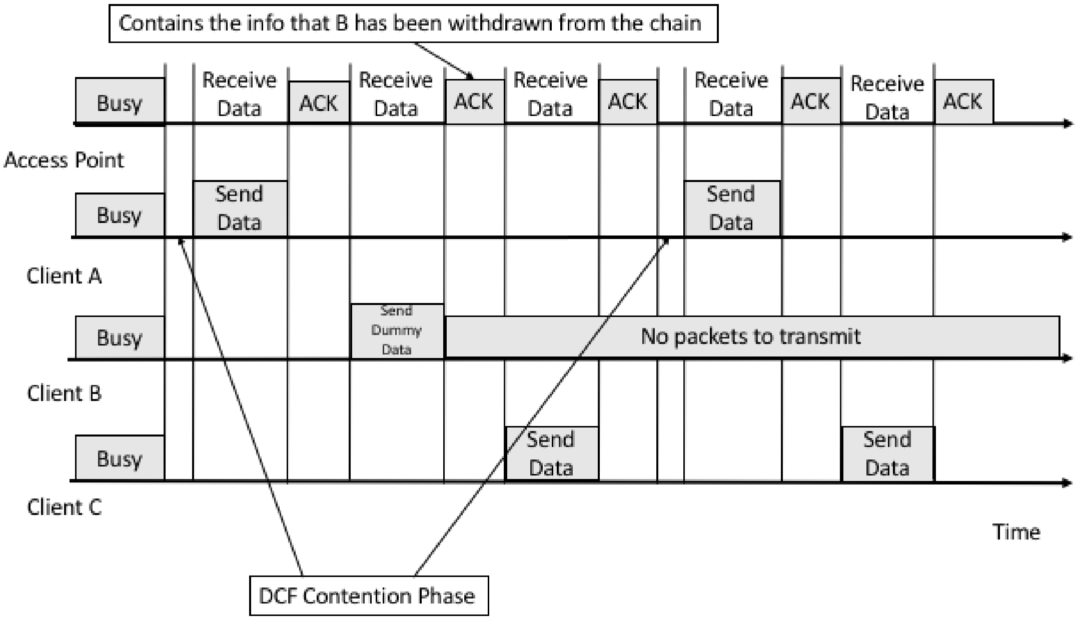


Figure 3.6: Withdraw mechanism of Q-CHAIN

Client / Condition	A	B	C
ABC are in the chain	ABC	ABC	ABC
B withdraws from the chain	AC	∅	AC

Table 3.4: The Chain Table under Withdraw Mechanism

3.3.4 Drawbacks of Q-Chain

3.3.4.1 Q-CHAIN is not robust under unreliable data transmission

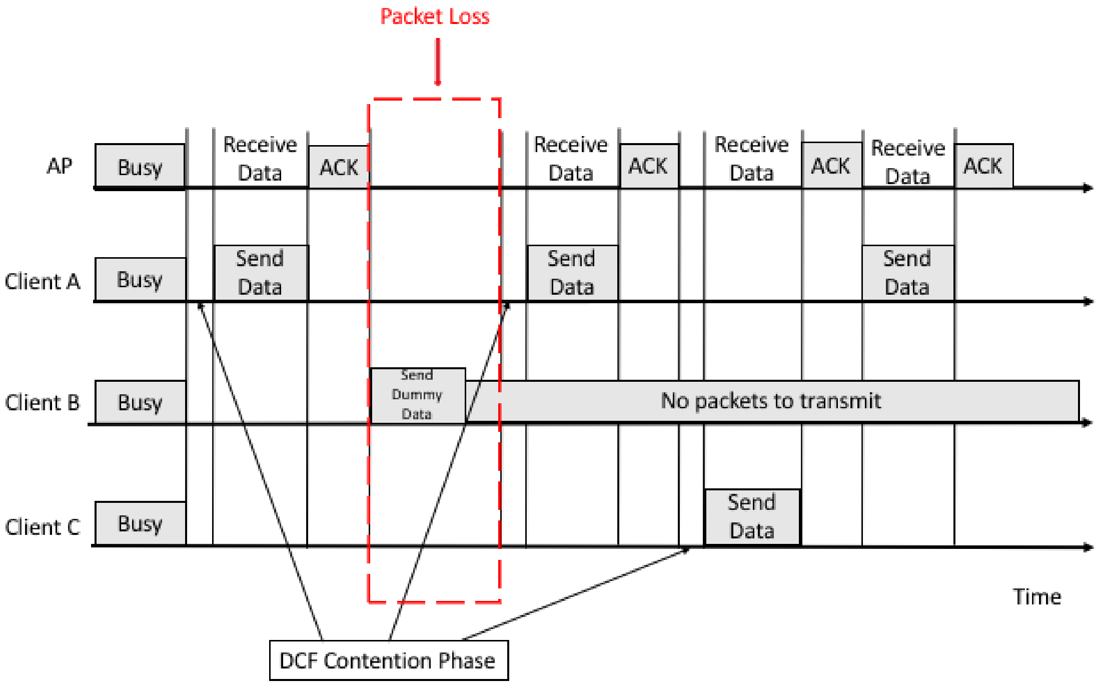


Figure 3.7: Q-CHAIN is not robust under unreliable data transmission

Not consider the Packet Loss condition, the Join and Withdraw Mechanism achieve throughput-optimality which has been proved in [9]. However, when we consider the Packet Loss condition, Q-CHAIN will not have a good performance.

Table 3.5 shows the original Q-CHAIN is not robust under unreliable data transmission. The Q-CHAIN cannot handle this case, and it will destroy the consistency of the chain table and influence the performance of Q-CHAIN seriously.

In Figure 3.7, we assume client A, B, C have been in the chain. When AP has not received the dummy packet from Client B, the chain table of Client A and C are still ABC. Then when Client

A has transmitted successfully, no client will do the piggyback transmission after it forever. The performance of Q-CHAIN, in this case, is only similar to the legacy DCF.

The performance of the original Q-CHAIN under unreliable data transmission may be worse than the legacy DCF. Table 3.5 shows this worse case. After the dummy packet from Client B has been lost, Client D joins into the chain. Follow the Join Mechanism as mentioned before, Client A will run piggyback transmission after Client D. The chain tables of Client A and C will update to DABC. For Client D, its chain table will only update to DA. Since the client B is not in the chain now, this chain cycle will finish after client A has been transmitted, Client D will not overhear any ACKs contains B's information. Then, Client B comes back to the chain in the next phase. At this time, the chain table of Client A and C will not change, because B has already in their chain table. Also, this is the first time for Client D to overhear the ACK contains B's MAC Address, the chain table of Client D should update to BDA. In this case, the predecessor of Client D is B, the predecessor of Client C is also B. When B has transmitted successfully, Client C and D will execute the piggyback transmission at the same time and the collision will happen.

When Q-CHAIN cannot maintain the consistency of all the chain tables, it means it cannot control the order of transmission in the channel. The collision cannot be avoided, the performance of Q-CHAIN is poor severely.

Client Condition	A	B	C	D
ABC are in the chain	ABC	ABC	ABC	∅
B withdraws from the chain(But the packet is lost)	ABC	∅	ABC	∅
D joins into the chain	DABC	∅	DABC	DA
B joins into the chain	DABC	B	DABC	BDA

Table 3.5: The Chain Table under Packet Loss with the legacy Withdraw Mechanism

3.3.4.2 Q-CHAIN needs an intelligent AP

The AP under Q-CHAIN should have the ability to decode the dummy packet, and the clients should have the ability to distinguish the different messages from different types of ACK. It also needs additional implementation on the AP side and the client side. It is not good enough to promote to the existing networks.

4. DESIGN OF Q-CHAIN

In this section, it presents the principles of the design. Moreover, it provides the new design of the Withdraw Mechanism of Q-CHAIN.

4.1 The principles of design

4.1.1 Handle the Unreliable Data condition

The new design of Q-CHAIN should have the ability to handle the unreliable data condition. It means the new design should be robust under packet loss condition, so the improvement of Q-CHAIN is realistic.

4.1.2 Do not need an intelligent AP

The AP in the original Q-CHAIN should have an additional implementation on decoding dummy packet and the ACK with special meaning. The new design of Q-CHAIN reduces the requirement of the intelligent AP. The new Q-CHAIN can fit on the AP with legacy DCF algorithm.

4.1.3 Inherit all the advantages of the original Q-CHAIN

The join mechanism of the new design of Q-CHAIN is same as the original one. Also, the new design should inherit all the advantages of the original Q-CHAIN.

4.2 New Withdraw Mechanism of Q-CHAIN

To solve the problem we have mention before, we propose a new Withdraw Mechanism of Q-CHAIN. If the client doesn’t have packets to transmit, it will not transmit anything into the channel. The new Withdraw Mechanism only uses overhearing ACK to judge whether the client has been withdrawn from the chain or not. We assume the ACK is reliable, the ACK from AP can be overheard by all the clients.

Figure 4.1 describes the new Withdraw Mechanism, client A, B, C have been transmitted in the chain and the chain table (Table 4.1) is ABC. In the timeline, client A has transmitted in the channel, client B should follow Client A using piggyback transmission. However, B doesn't have packets to transmit, it will not transmit anything. Then Client A and C will not overhear the ACK contains B's MAC Address. So Client A and C know B has withdrawn from the chain, then we force the client whose order is behind the withdrawing client to drop out of the chain. In this case, Client B and C will drop from the chain at the same time, the chain table will update to A. Client C can attend into the chain using the same Join Mechanism under Q-CHAIN.

Under the new Withdraw Mechanism, we don't need the dummy packet to tell AP which client has withdrawn from the channel, therefore we don't need a intelligent AP to decode the special packet and transmit a special ACK.

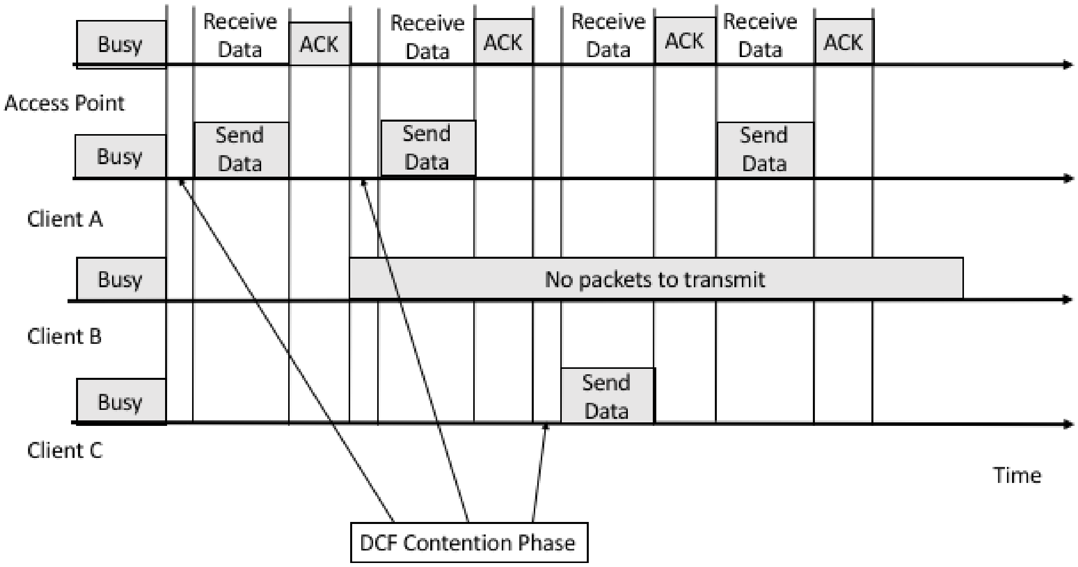


Figure 4.1: The new Withdraw mechanism of Q-CHAIN

Client Condition	A	B	C	D
ABC are in the chain	ABC	ABC	ABC	$\emptyset$
B withdraws from the chain/Packet Loss	A	$\emptyset$	A	$\emptyset$
D joins into the chain	DA	DA	$\emptyset$	DA
C joins into the chain	CDA	CDA	CDA	CDA

Table 4.1: The Chain Table under unreliable data transmission with the new Withdraw Mechanism

4.3 The new Withdraw Mechanism of Q-CHAIN is robust under unreliable data transmission

Figure 4.2 shows that the new Withdraw Mechanism of Q-CHAIN can handle the case under unreliable data transmission. When the packet from the client is lost, AP will not transmit any ACK contains its information. Therefore, this client will be seen as the one has withdrawn from the chain. All chain tables will drop the clients after the withdrawing one include itself. In Figure 4.2, the packet from Client B is lost, then the chain table of Client A and C will update to A. When Client A has transmitted successfully in the next phase, no client will execute the piggyback transmission after it. Then when Client C or B contends the channel, it will join into the chain using the same Join Mechanism.

Table 4.1 shows that the new withdraw mechanism keeps the consistency of the chain table. The new design of Q-CHAIN is robust under unreliable data transmission.

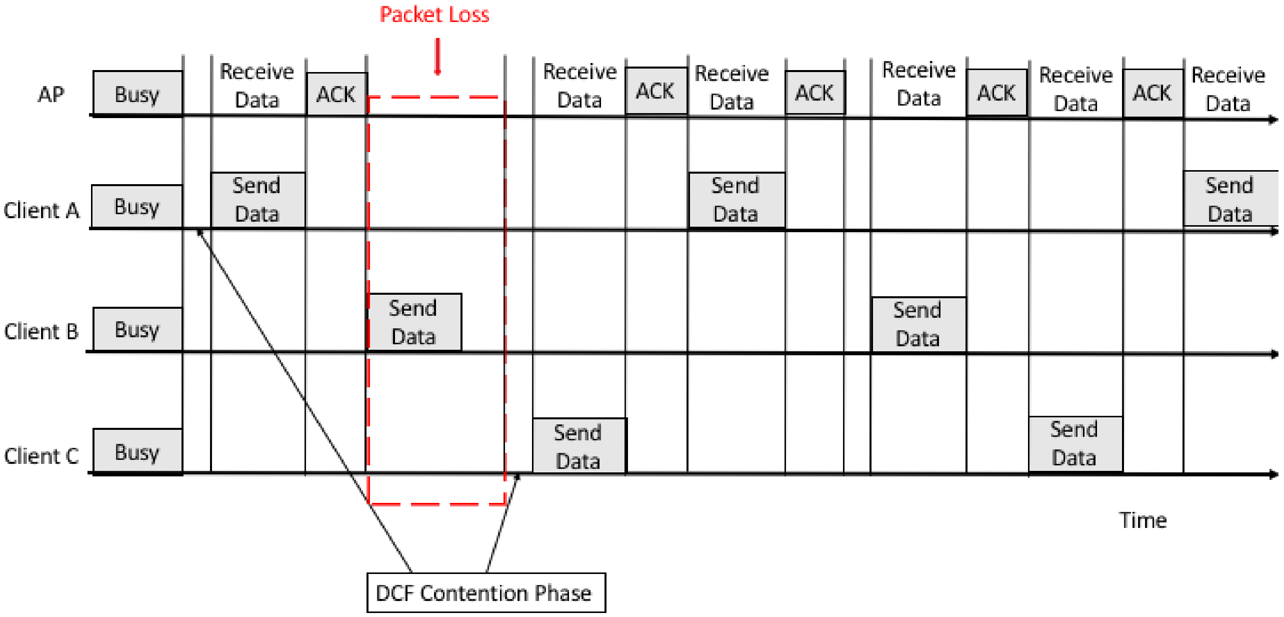


Figure 4.2: The new Withdraw mechanism of Q-CHAIN under unreliable data transmission

5. IMPLEMENTATION OF Q-CHAIN

We will discuss the implementation details of Q-CHAIN in this section.

5.1 The principles of implementation

5.1.1 Avoid modifying the architecture or the infrastructure of AP

Q-CHAIN should be implemented on the client side, and it can work under any APs which support the basic DCF. Hence, Q-CHAIN should be adaptable with the simple AP and has more promotional value than CHAIN.

5.1.2 Support a fully-decentralized protocol on the client side

The clients should maintain piggyback table by CSMA/CA and overhearing ACKs to achieve Q-CHAIN. Furthermore, each client does not need to interact with other clients to update the piggyback table.

5.1.3 Avoid adding any frames to the transmitted packets

Because Q-CHAIN is based on DCF, it only contends the channel via CSMA/CA and overhearing ACKs. It does not need to add any control packet or control channel. Also, it does not need to change the structure of any original data formats (i.e. PHY frame, MAC data frame, MAC ACK frame). Q-CHAIN should employ a simple design and low implementation.

5.2 WiMAC: Implementation environment

5.2.1 Overview of WiMAC

The research is based on the WiMAC[8] experiment environment. WiMAC is a testbed to prototype real-time MAC protocols on wireless networks. It separates the functions into Mechanisms in FPGA and Policies in Host. The mechanisms in FPGA implements time-critical functionality, which achieves packet transmissions in the wireless network. The operations on FPGA is simple, low-level, and based on the clock cycle. The policies in Host is more like a high-level programming scheme to handle channel contention and generate the packet, it can support more data structure and complex operations.

5.2.2 The problem of the legacy CHAIN on WiMAC

First, the clients cannot join or withdraw from the chain dynamically under legacy Q-CHAIN. The legacy CHAIN in the lab has been implemented using fixed piggyback relation. Each client has a fixed predecessor client. The piggyback relation is hardcoded into the system as an input.

Moreover, the legacy CHAIN cannot be incorporated with the legacy DCF. The legacy CHAIN still follows the contention process of DCF and use ACK to trigger the piggyback transmission. However, when all CHAIN has been connected as a cycle, CHAIN will not be compatible with DCF any more. For example, Client A, B, C, D are connected as a cycle, not considering the case under unsaturated traffic and unreliable data transmission, once one of the CHAIN nodes transmit successfully in the system, the other CHAIN nodes will transmit using piggyback transmission one by one. So the DCF nodes cannot sense a continuous idle channel in DIFS period, it means the clients under DCF will not transmit in the system anymore.

5.2.3 Q-CHAIN on WiMAC

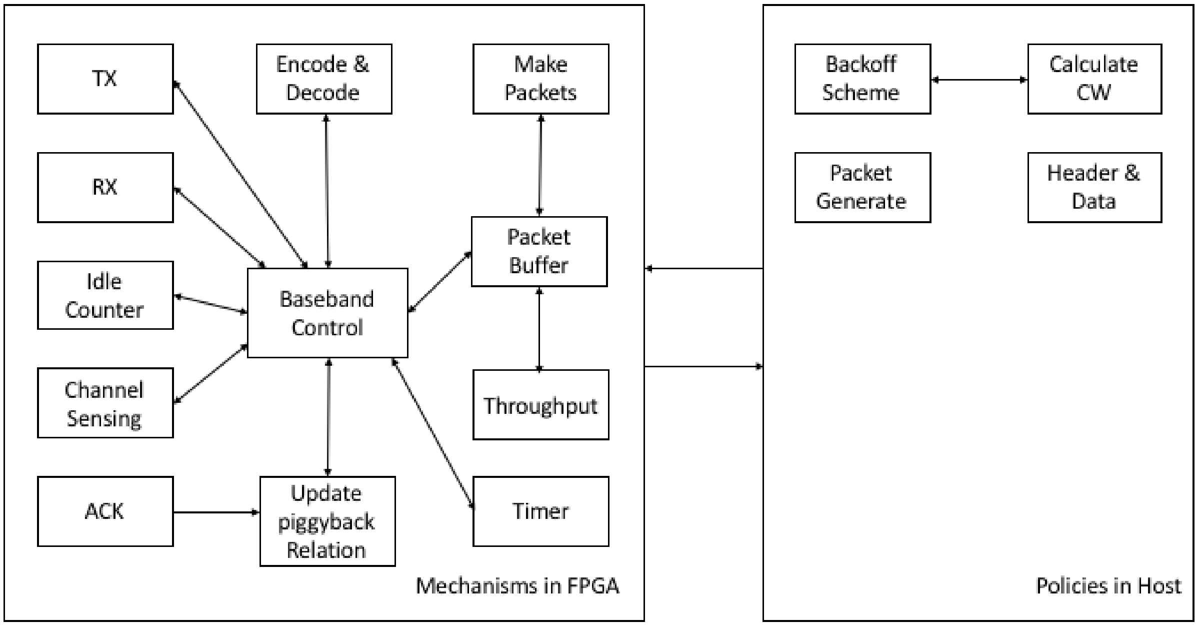


Figure 5.1: Mechanism and Policy of Q-CHAIN on WiMAC

Before the research, WiMAC has already achieved an 802.11-like reference design, including CSMA/CA protocol using random exponential backoff algorithm, then Q-CHAIN is built on the top of the existing design. Figure 5.1 describes the framework structure of Q-CHAIN on WiMAC. Most of the implementation of Q-CHAIN is based on FPGA. The main reason is that the interfacing latency between host and FPGA is around 192 μs [11]. In IEEE 802.11a, the slot time is 9 μs, DIFS is 34 μs. Under Q-CHAIN, we use DIFS as an interframe to judge if the transmission is a piggyback transmission. Hence, this interfacing latency is so obvious that it cannot be ignored. Therefore, although the operations in FPGA should obey a critical time rule and the code is hard to debug, we need to implement Q-CHAIN on FPGA.

5.3 Compatible with the legacy DCF

The previous work of CHAIN on WiMAC is that we can assign fixed precedence to execute the piggyback transmission for each client. The problem of the legacy CHAIN on WiMAC has been mentioned before. The first thing we need to do is to break down the CHAIN Loop and make it be compatible with legacy DCF.

The main idea to support Q-CHAIN to be incorporated with DCF is to achieve each client can transmit using piggyback transmission at most once in one loop cycle. The loop cycle means the clients under Q-CHAIN have been transmitted via piggyback transmission continuously and each client at most transmit once in one cycle. For example, we assume the chain table is ABCD, the most extended loop cycle is from A to D. If Client A, B, C, D has been transmitted using piggyback transmission continuously, Client A will not execute the piggyback transmission after Client D, because we need to begin the next loop cycle.

For our purpose, we need to control each client could only transmit only once between two idle DIFS periods. The piggyback transmission will be forced to stop after each client has been transmitted in one loop cycle, the chain will not loop forever. Moreover, this feature will not influence the existing DCF. Each client under DCF will start the backoff period after sensing the idle channel during DIFS. So at the beginning of loop cycle, all clients either under Q-CHAIN or DCF using the same contention process to contend the channel.

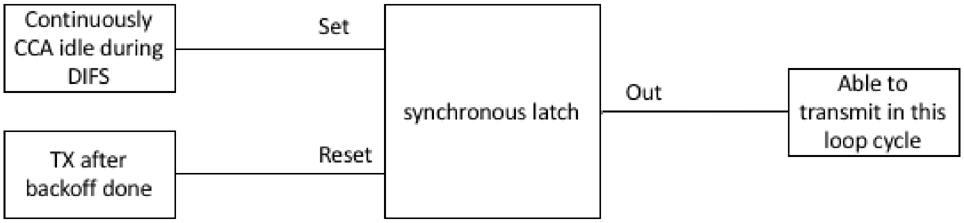

Continuously CCA idle during DIFS
Set
TX after backoff done
Reset
synchronous latch
Out
Able to transmit in this loop cycle

Figure 5.2: The logic of Synchronous Latch

We implemented this feature using Synchronous Latch. Figure 5.2 shows the logic of it. The initial value of $able\ to\ transmit\ in\ this\ loop\ cycle$ is false. Once the channel is idle (CCA is not busy) during DIFS, the Synchronous Latch will lock the true value, and the output $Able\ to\ transmit\ in\ this\ loop\ cycle$ will turn to true until we this client has been transmitted after backoff done. $TX\ after\ backoff\ done$ is a pulse to reset the Synchronous Latch output value.

5.4 The clients can join into Q-CHAIN dynamically

As mentioned before, we need to implement a candidate table as an input.

5.4.1 Maintain the chain table

The most important thing in the implementation is to maintain the chain table for each client. All the clients in the chain should have the same chain table. Before joining into the chain, the chain table of each client is empty. Under Join Mechanism of Q-CHAIN, each client in the candidate table has four cases in the system:

(1) $It's\ not\ in\ the\ chain$

The client has not transmitted in the system.

(2) $It's\ the\ first\ time\ into\ the\ chain$

When the client received the ACK contains itself MAC Address and the chain table is empty at this time, we will define this client is the head of the current chain.

(3) $It's\ the\ first\ time\ to\ update\ the\ chain\ table$

This case follows behind case 2. The client will update the whole chain when it is the first time to itself to be the head. Case 3 should finish in one chain loop, we take the client that transmits after itself as the second order in the chain table, the next one is the third order, and so on, until the channel senses an idle DIFS duration.

(4) $It\ should\ update\ the\ chain\ table\ again$

If the chain table of the client is not empty, it means it has already transmitted in the chain. When the client receives an ACK whose MAC Address is not in the current chain, but it's in the candidate table, this client will be the new head of the chain, and the previous chain table will add behind the new head.

5.4.2 Update the piggyback address

Under Q-CHAIN, each client owns a unique predecessor to do the piggyback transmission. We store the MAC Address of the predecessor as the piggyback address. When the client overhears the ACK contains its piggyback address, this client will execute piggyback transmission after SIFS. Each client should update the piggyback address in two cases:

(1) *It′s the first time to join into the chain*

When the client is the first time to join into the chain, to make the chain like a circle, it should use the last client's MAC Address in the client table as its piggyback address.

(2) *It′s the previous head in the chain*

When the client is the previous head in the chain, which means a new client comes into the chain to be the new head, and it will be the second one in the chain table. In this case, the previous head should use the new head's MAC Address to be the piggyback address.

5.5 The clients can withdraw from Q-CHAIN dynamically

Be different with the legacy Withdraw Mechanism, and we purpose a new Withdraw Mechanism of Q-CHAIN. We can know the theoretically transmitted order via the chain table. Also in every chain loop, we record the status of each client in the chain table, to observe if it has transmitted. If some clients have not transmitted in the chain loop, we can find which one is the breakpoint. This client should be seen as the one who has withdrawn from the chain. Moreover, the client whose order is behind the withdrawing client's order should drop from the chain. Then, we use the same way to update the piggyback address which has mentioned in Section 5.4.2.

6. EXPERIMENTAL RESULT

In this section, we evaluate the proposed protocol and provide the experimental results based on the WiMAC[8] experiment environment. Figure 6.1 shows the experiment system, each node (AP or Clients) contains one USRP-2153Rs (FPGA) and one Windows laptop (Host Machine). We obtain the experimental result using one AP with multiple clients (at most four clients).

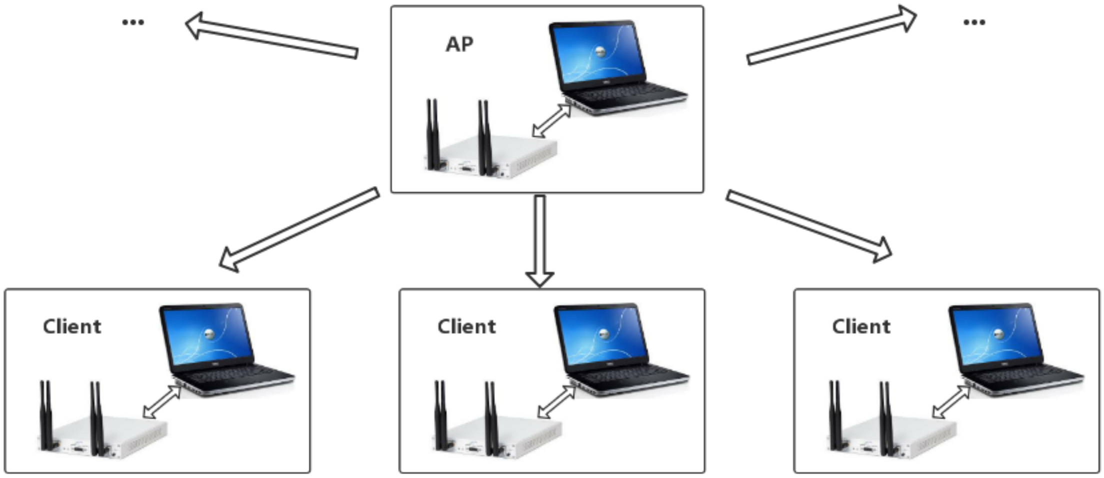


Figure 6.1: Experiment framework: using Host machines and USRP-2153Rs as wireless AP and clients

6.1 Basic parameter of the experiment

Table 6.1 shows the basic parameters in the experiment and Table 6.2 summarizes the Data Source we use in the experiment. We consider three protocols: the legacy DCF, the legacy CHAIN on WiMAC and the Q-CHAIN. The legacy DCF in the experiment does not contain the retransmission part since we use PN data as the transmitted data type and we do not care about the quality of data.

The most advantage of Q-CHAIN is to reduce collisions in the network so that the performance will be better than DCF. Due to the limited number of equipment (at most one node for AP, four nodes for clients) we could use, if we use the default CW (CW_{min} = 15, CW_{max} = 1023) in IEEE 802.11a standard, we cannot see too many collisions under either DCF or Q-CHAIN, It's hard to evaluate the performance of Q-CHAIN. Therefore, we shrink the CW in different experiments, which will be pointed out in each part.

Parameter Name	Value	Comment
Transmit Frequency	2.5 GHz	
Subcarrier Format	20 MHz	Under IEEE 802.11a
MCS	4 16-QAM (1/2) (24 M bit/s)	
Power Level	0 dBm	

Table 6.1: Experiment Parameters

Parameter Condition	Data Type	Packet Size (Byte)	Transmitted Packet (/s)	Data Rate (M bit/s)
Saturated Traffic	PN Data	1500	2000	24
Unsaturated Traffic	PN Data	1500	10	0.12

Table 6.2: Data Source and Data Parameter

6.2 Compatible with legacy DCF

The experiment in this section only considers the saturated traffic without Packet Loss. All clients under either CHAIN or DCF use the same experiment parameters in Table 6.1 and Table 6.2.

The drawbacks of the legacy CHAIN on WiMAC has been mentioned in section 5.2.2. Table 6.3 presents the average throughput for each client under either CHAIN or DCF. From the results, the clients under the legacy CHAIN can get the maximal throughput because the piggyback transmission will never stop. It means the channel will not have any idle DIFS period, the clients under DCF have no chance to contend into the network. The throughput of the client under DCF confirms this vital problem.

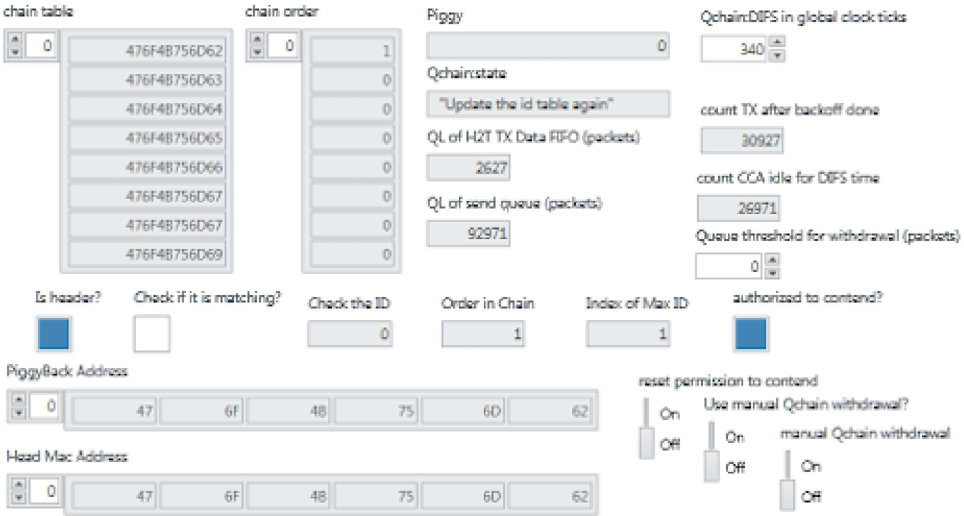
Policy Condition	CHAIN	DCF
1 in CHAIN 1 in DCF	19.1	0.0005
2 in CHAIN 1 in DCF	9.55	0.002
3 in CHAIN 1 in DCF	6.37	0

Table 6.3: The average throughput (M bit/s) of clients in legacy CHAIN, CW_{min} = 8, CW_{max} = 8

Table 6.4 shows the Q-CHAIN could be incorporated with legacy DCF. Due to each client under Q-CHAIN could only transmit at most once using piggyback transmission in one loop cycle. The condition that we use one Q-CHAIN node and one DCF node is equal to use two DCF nodes. So the throughput of DCF node and Q-CHAIN node is the same in the table. When more clients add into the chain, the DCF node will have less chance to transmit in the channel. Because once one of the Q-CHAIN nodes has been transmitted successfully, its successor client will overhear the ACK then execute the piggyback transmission after SIFS. The DCF node should wait after one loop cycle to sense the next idle channel during DIFS.

Policy Condition	CHAIN	DCF
1 in Q-CHAIN 1 in DCF	8.95	8.95
2 in Q-CHAIN 1 in DCF	6.47	4.81
3 in Q-CHAIN 1 in DCF	5.08	2.55

Table 6.4: The average throughput (M bit/s) of clients in Q-CHAIN, CW_{min} = 8, CW_{max} = 8



chain table: 0
476F4B756D62
476F4B756D63
476F4B756D64
476F4B756D65
476F4B756D66
476F4B756D67
476F4B756D67
476F4B756D69

chain order: 0
1
0
0
0
0
0
0
0

Piggy: 0

Qchain:state: "Update the id table again"

QL of H2T TX Data FIFO (packets): 2627

QL of send queue (packets): 92971

Qchain:DIFS in global clock ticks: 340

count TX after backoff done: 30927

count CCA idle for DIFS time: 26971

Queue threshold for withdrawal (packets): 0

Is header? ■

Check if it is matching? □

Check the ID: 0

Order in Chain: 1

Index of Max ID: 1

authorized to contend? ■

PiggyBack Address: 0 | 47 | 6F | 48 | 75 | 6D | 62

Head Mac Address: 0 | 47 | 6F | 48 | 75 | 6D | 62

reset permission to contend: On / Off

Use manual Qchain withdrawal? On / Off

manual Qchain withdrawal: On / Off

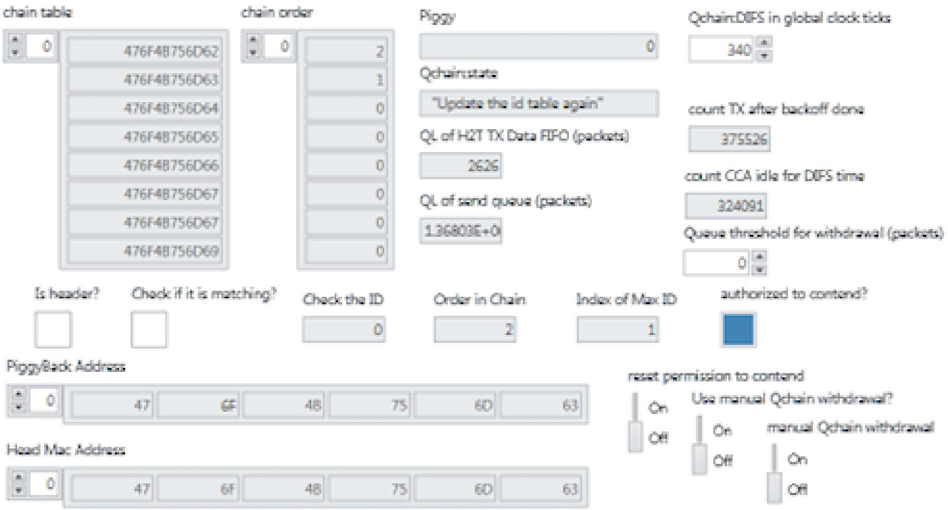
Figure 6.2: The chain table of 47:6F:48:75:6D:62 (One client in the chain)

6.3 A demo of Join Mechanism of Q-CHAIN

This section provides the demo of Join Mechanism.

Figure 6.2 shows the chain table and piggyback address of the client under there is only one client in the chain.

Figure 6.3 and Figure 6.4 show the chain table and piggyback address of each client under there are two clients in the chain.



chain table 0	chain order 0
476F48756D62	2
476F48756D63	1
476F48756D64	0
476F48756D65	0
476F48756D66	0
476F48756D67	0
476F48756D67	0
476F48756D69	0

Piggy: 0

Qchain:state: "Update the id table again"

QL of H2T TX Data FIFO (packets): 2626

QL of send queue (packets): 1.36803E+0

Qchain:DIFS in global clock ticks: 340

count TX after backoff done: 375526

count CCA idle for DIFS time: 324091

Queue threshold for withdrawal (packets): 0

Is header? □

Check if it is matching? □

Check the ID: 0

Order in Chain: 2

Index of Max ID: 1

authorized to contend? ■

PiggyBack Address: 0 | 47 | 6F | 48 | 75 | 6D | 63

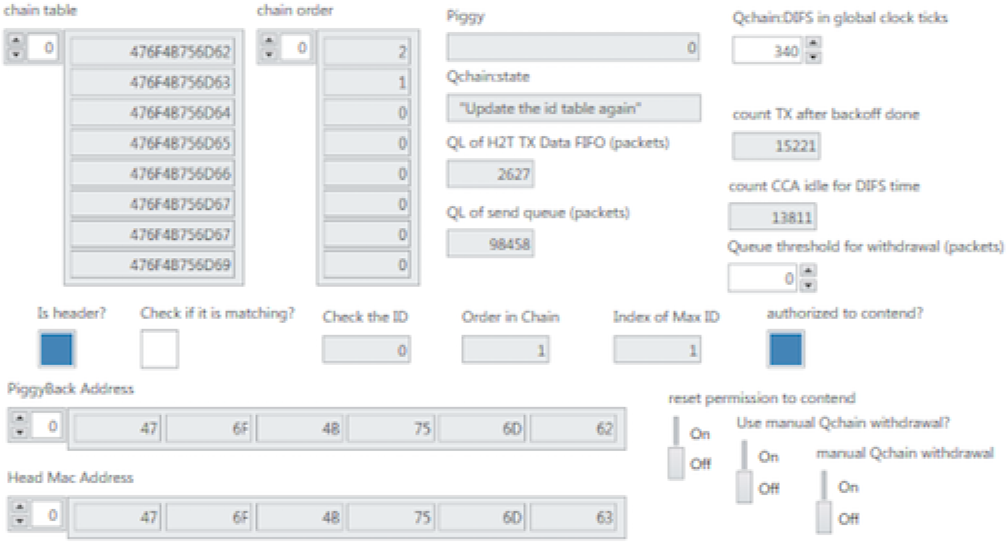
Head Mac Address: 0 | 47 | 6F | 48 | 75 | 6D | 63

reset permission to contend: On / Off

Use manual Qchain withdrawal? On / Off

manual Qchain withdrawal: On / Off

Figure 6.3: The chain table of 47:6F:48:75:6D:62 (Two clients in the chain)



chain table 0	chain order 0
476F48756D62	2
476F48756D63	1
476F48756D64	0
476F48756D65	0
476F48756D66	0
476F48756D67	0
476F48756D67	0
476F48756D69	0

Piggy: 0

Qchain:state: "Update the id table again"

QL of H2T TX Data FIFO (packets): 2627

QL of send queue (packets): 98458

Qchain:DIFS in global clock ticks: 340

count TX after backoff done: 15221

count CCA idle for DIFS time: 13811

Queue threshold for withdrawal (packets): 0

Is header? ■

Check if it is matching? □

Check the ID: 0

Order in Chain: 1

Index of Max ID: 1

authorized to contend? ■

PiggyBack Address: 0 | 47 | 6F | 48 | 75 | 6D | 62

Head Mac Address: 0 | 47 | 6F | 48 | 75 | 6D | 63

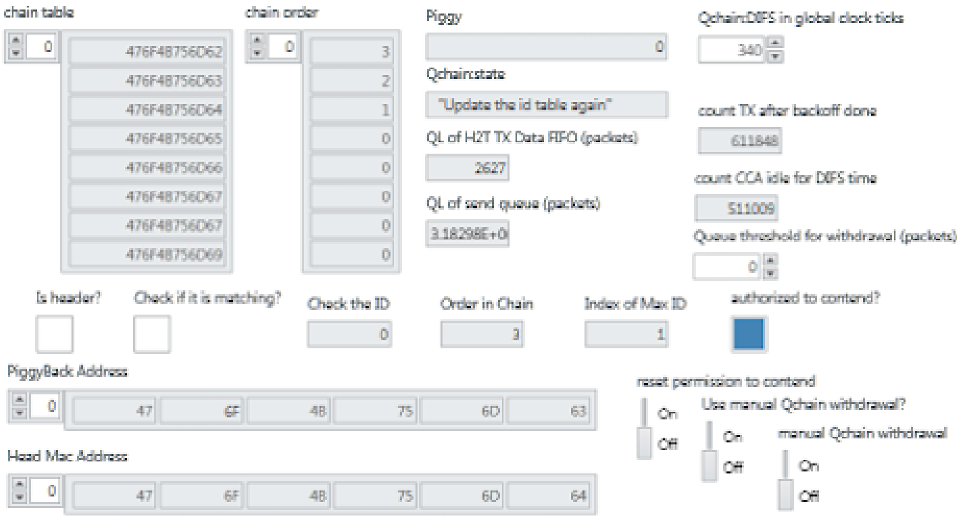
reset permission to contend: On / Off

Use manual Qchain withdrawal? On / Off

manual Qchain withdrawal: On / Off

Figure 6.4: The chain table of 47:6F:48:75:6D:63 (Two clients in the chain)

Figure 6.5, Figure 6.6 and Figure 6.7 show the chain table and piggyback address of each client under there are three clients in the chain.



chain table
0
476F48756D62
476F48756D63
476F48756D64
476F48756D65
476F48756D66
476F48756D67
476F48756D67
476F48756D69

chain order
0
3
2
1
0
0
0
0
0

Piggy
0

Qchain:state
"Update the id table again"

QL of H2T TX Data FIFO (packets)
2627

QL of send queue (packets)
3.18298E+0(

Qchain:DIFS in global clock ticks
340

count TX after backoff done
611848

count CCA idle for DIFS time
511009

Queue threshold for withdrawal (packets)
0

Is header? □

Check if it is matching? □

Check the ID
0

Order in Chain
3

Index of Max ID
1

authorized to contend? ■

PiggyBack Address
0
47 6F 48 75 6D 63

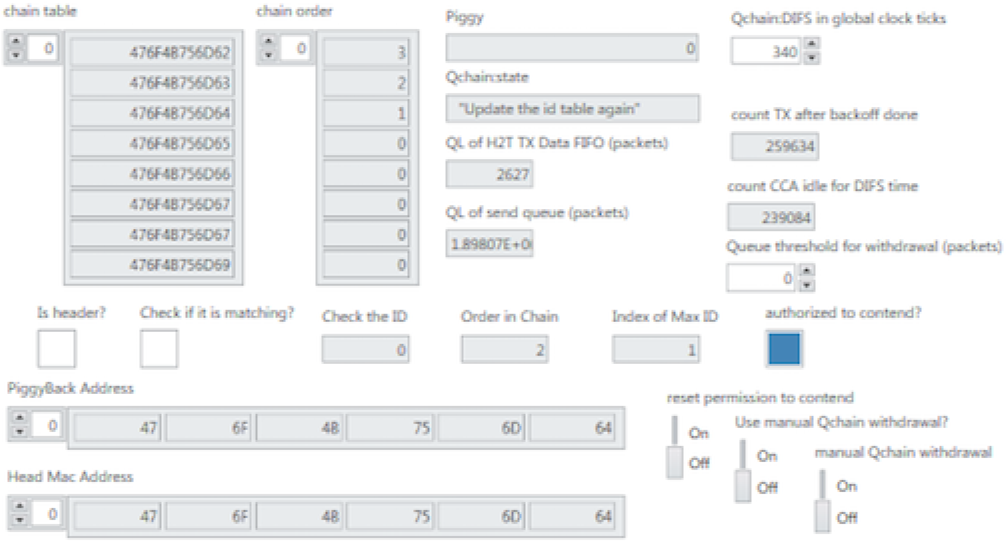
Head Mac Address
0
47 6F 48 75 6D 64

reset permission to contend
On
Off

Use manual Qchain withdrawal?
On
Off

manual Qchain withdrawal
On
Off

Figure 6.5: The chain table of 47:6F:48:75:6D:62 (Three clients in the chain)



chain table
0
476F48756D62
476F48756D63
476F48756D64
476F48756D65
476F48756D66
476F48756D67
476F48756D67
476F48756D69

chain order
0
3
2
1
0
0
0
0
0

Piggy
0

Qchain:state
"Update the id table again"

QL of H2T TX Data FIFO (packets)
2627

QL of send queue (packets)
1.89807E+0(

Qchain:DIFS in global clock ticks
340

count TX after backoff done
259634

count CCA idle for DIFS time
239084

Queue threshold for withdrawal (packets)
0

Is header? □

Check if it is matching? □

Check the ID
0

Order in Chain
2

Index of Max ID
1

authorized to contend? ■

PiggyBack Address
0
47 6F 48 75 6D 64

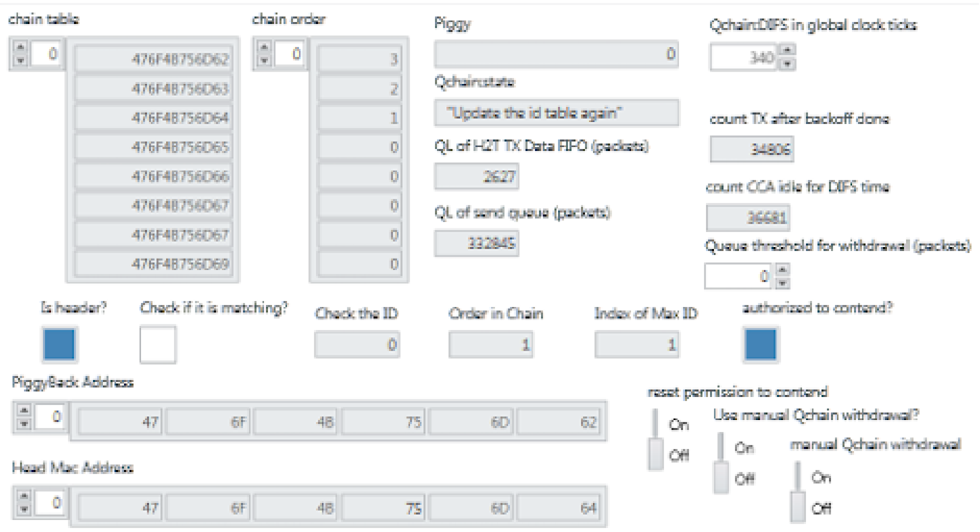
Head Mac Address
0
47 6F 48 75 6D 64

reset permission to contend
On
Off

Use manual Qchain withdrawal?
On
Off

manual Qchain withdrawal
On
Off

Figure 6.6: The chain table of 47:6F:48:75:6D:63 (Three clients in the chain)



chain table: 0

chain table	chain order
476F48756D62	3
476F48756D63	2
476F48756D64	1
476F48756D65	0
476F48756D66	0
476F48756D67	0
476F48756D67	0
476F48756D69	0

chain order: 0

Piggy: 0

Qchainstate: "Update the id table again"

QL of H2T TX Data FIFO (packets): 2627

QL of send queue (packets): 332845

QchainDIFS in global clock ticks: 340

count TX after backoff done: 34806

count CCA idle for DIFS time: 36681

Queue threshold for withdrawal (packets): 0

Is header? ■

Check if it is matching? □

Check the ID: 0

Order in Chain: 1

Index of Max ID: 1

authorized to contend? ■

PiggyBack Address: 0 | 47 | 6F | 48 | 75 | 6D | 62

Head Mac Address: 0 | 47 | 6F | 48 | 75 | 6D | 64

reset permission to contend: On / Off

Use manual Qchain withdrawal?: On / Off

manual Qchain withdrawal: On / Off

Figure 6.7: The chain table of 47:6F:48:75:6D:64 (Three clients in the chain)

The result confirm that the implementation gives us the correct piggyback relation. At first, there is one client in the chain, the piggyback address of the client is itself. If a new client joins into the chain, it should be the head of the chain, and the piggyback address of the new head and previous head should be updated. Also if there are three clients in the chain, the demo shows all chain tables are the same, and each client has a unique predecessor.

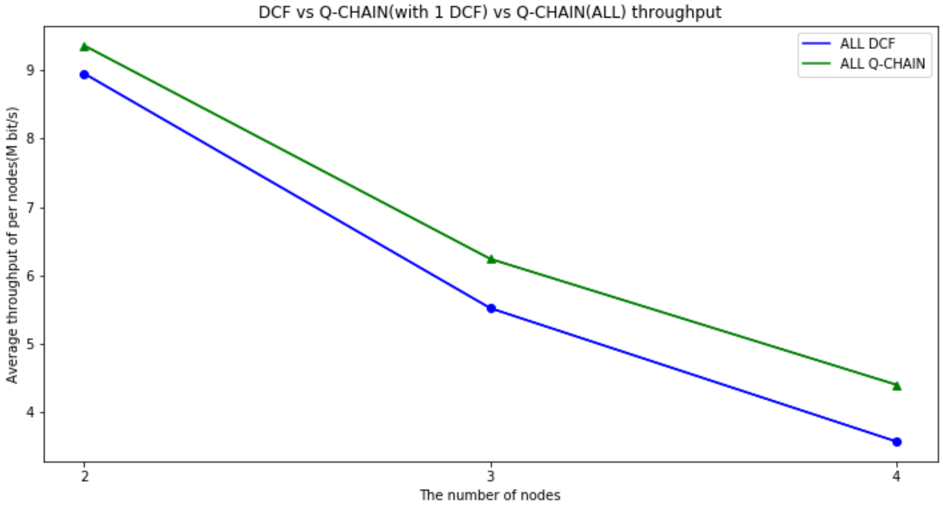


Figure 6.8: The performance of Q-CHAIN and DCF

6.4 Q-CHAIN vs DCF under saturated traffic

This experiment in this section compares the performance of Q-CHAIN and legacy DCF under saturated traffic. Under saturated traffic, the transmitted queue is always full, and we do not consider the Packet Loss condition. Because the most advantage of Q-CHAIN is to reduce the potential collision in the channel, we shrink the contention window size (CW_{min} = 8, CW_{max} = 8) to increase the probability of collision.

Figure 6.8 shows the performance of Q-CHAIN and legacy DCF. We compare these two protocols in two cases. All clients are under DCF and all clients are under Q-CHAIN. The experimental results show that the performance of Q-CHAIN is much better than DCF.

6.5 Q-CHAIN vs DCF under unsaturated traffic and Packet Loss condition

In this section, we have an experiment to measure the performance of Q-CHAIN under unsaturated traffic and Packet Loss condition. In this experiment, we also shrink the contention window to increase the probability of collision, and we do not consider retransmission in our experiment.

6.5.1 Under unsaturated traffic

The experiment compares Q-CHAIN with the fixed-order CHAIN to evaluate the performance of Q-CHAIN under unsaturated traffic. We use A, B, C, D four clients under Q-CHAIN/fixed-order CHAIN, among the four clients, client A and C have the small packets, client B and D are under saturated traffic. In theory, Client A and C do not have enough packets to transmit in most of the time, so for Client B and D, it is hard to execute the piggyback transmission. In most of the time, they only contend the channel using the legacy DCF, so the performance of fixed-order CHAIN should be the same as the legacy DCF. Under Q-CHAIN, all the clients can join and withdraw from the chain dynamically, so Client B and D can execute more piggyback transmission. The performance of Q-CHAIN nodes should be much better than the fixed-order CHAIN.

Throughput (M bit/s) Condition	A/C	B/D
DCF	0.05	4.91
Fixed Order CHAIN	0.07	4.91
Dynamic Q-CHAIN	0.07	6.15

Table 6.5: Four Clients, A and C have the small packets , CW_{min} = 2, CW_{max} = 2

Throughput (M bit/s) Condition	A/C	B/D
DCF	0.07	6.49
Fixed Order CHAIN	0.08	6.49
Dynamic Q-CHAIN	0.08	7.51

Table 6.6: Four Clients, A and C have the small packets , CW_{min} = 3, CW_{max} = 3

Throughput (M bit/s) Condition	A/C	B/D
DCF	0.09	7.51
Fixed Order CHAIN	0.10	7.51
Dynamic Q-CHAIN	0.10	8.15

Table 6.7: Four Clients, A and C have the small packets , CW_{min} = 4, CW_{max} = 4

Table 6.5, Table 6.6 and Table 6.7 show the throughput of each client under different MAC protocol with different CW. Table 6.8 calculates the improvement of Q-CHAIN under unsaturated traffic. It shows that Q-CHAIN is robust under the unsaturated traffic. The more collisions in the system, the performance of Q-CHAIN is much better than the fixed-order CHAIN.

Condition \ CW (CW_{min} = CW_{max})	2	3	4
Fixed Order CHAIN	4.91	6.49	7.51
Dynamic Q-CHAIN	6.15	7.51	8.15
	25.25%	15.72%	8.52%

Table 6.8: The improvement of dynamic Q-CHAIN than fixed-order CHAIN

6.5.2 Consider Packet Loss transmission

To create an environment under Packet Loss on WiMAC, we give AP a probability not to transmit the ACK when it receives the packets from the clients.

Condition \ Throughput (M bit/s)	A/C	B/D
DCF	0.04	4.41
Fixed Order CHAIN	0.06	4.41
Dynamic Q-CHAIN	0.06	5.47

Table 6.9: Four Clients, A and C have the small packets , CW_{min} = 2, CW_{max} = 2, the Packet Loss rate = 0.1

Throughput (M bit/s) Condition	A/C	B/D
DCF	0.06	5.84
Fixed Order CHAIN	0.07	5.84
Dynamic Q-CHAIN	0.07	6.70

Table 6.10: Four Clients, A and C have the small packets , CW_{min} = 3, CW_{max} = 3, the Packet Loss rate = 0.1

Throughput (M bit/s) Condition	A/C	B/D
DCF	0.08	6.75
Fixed Order CHAIN	0.09	6.75
Dynamic Q-CHAIN	0.09	7.25

Table 6.11: Four Clients, A and C have the small packets , CW_{min} = 4, CW_{max} = 4, the Packet Loss rate = 0.1

Table 6.9, Table 6.10 and Table 6.11 show the throughput of each client under different MAC protocol with different CW. Table 6.12 calculates the improvement of Q-CHAIN under Packet Loss condition. The result reflects the similar trend in Section 6.4.1 and Q-CHAIN is robust under unreliable data transmission.

CW (CW_{min} = CW_{max}) Condition	2	3	4
Fixed Order CHAIN	4.41	5.84	6.75
Dynamic Q-CHAIN	5.47	6.70	7.25
	24.03%	14.73%	7.41%

Table 6.12: The improvement of dynamic Q-CHAIN than fixed-order CHAIN, the Packet Loss rate = 0.1

7. CONCLUSIONS

This thesis proposes a new design of Q-CHAIN, which can handle the unreliable data and doesn't need an intelligent AP to decode the dummy packets or transmit an ACK with special meaning. Moreover, this thesis presents the implementation of Q-CHAIN on WiMAC. The experimental results show that Q-CHAIN can be directly compatible with the existing DCF. The performance of Q-CHAIN is better than the legacy DCF under both saturated traffic and unsaturated traffic. It is robust under unsaturated traffic and unreliable data transmission.

REFERENCES

[1] Federico CalÃň, Marco Conti, and Enrico Gregori. Ieee 802.11 protocol: design and performance evaluation of an adaptive backoff mechanism. *IEEE Journal on Selected Areas in Communications*, 18(9), September 2000.

[2] Yang Xiao and Jon Rosdahl. Throughput and delay limits of ieee 802.11. *IEEE Communications Letters*, 6(8), August 2002.

[3] Hwangnam Kim and Jennifer C. Hou. Improving protocol capacity with model-based frame scheduling in ieee 802.11-operated wlans. In *MobiCom*, pages 190–204, 2003.

[4] Federico CalÃň, Marco Conti, and Enrico Gregori. Ieee 802.11 protocol: Design and performance evaluation of an adaptive backoff mechanism. *IEEE Journal on Selected Areas in Communications*, 18(9), September 2000.

[5] Andrew Muir and J. J. Garcia-Luna-Aceves. An efficient packet sensing mac protocol for wireless networks. *Mobile Networks and Applications*, pages 221–234, August 1998.

[6] Yang Xiao. Ieee 802.11 performance enhancement via concatenation and piggyback mechanisms. *IEEE Transactions on Wireless Communcations*, 40(5), September 2005.

[7] Zheng Zeng, Yan Gao, Kun Tan, and P. R. Kumar. Chain: Introducing minimum controlled coordination into random access mac. In *INFOCOM, 2011 Proceedings IEEE*, pages 2669–2677, April 2011.

[8] Simon Yau, Liang Ge, Ping-Chun Hsieh, I-Hong Hou, Shuguang Cui, P. R. Kumar, Amal Ekbal, and Nikhil Kundargi. Wimac: Rapid implementation platform for user definable mac protocols through separation. In *ACM SIGCOMM Computer Communication Review*, pages 109–110, August 2015.

[9] Ping-Chun Hsieh, I-Hong Hou, and Yue Yang. Q-chain: A throughput-optimal mac protocol for random access with piggyback transmissions. *submitted to IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, under review*, 2019. https://drive.google.com/file/d/1XnIfIfatMD9F2PLW7iu0rlNUyvaZQau1/view?usp=sharing.

[10] M. Zubair Shafiq, Lusheng Ji, Alex X. Liu, Jeffrey Pang, Shobha Venkataraman, and Jia Wang. Characterizing and optimizing cellular network performance during crowded events. *IEEE/ACM Transactions On Networking*, 24(3), June 2016.

[11] Simon Yau, Ping-Chun Hsieh, Rajarshi Bhattacharyya, K. R. Kartic Bhargav, Srinivas Shakkottai, I-Hong Hou, and P. R. Kumar. Puls: Processor-supported ultra-low latency scheduling. *Mobihoc '18 Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 261–270, 2018.