香港城市大學
City University
of Hong Kong

Department of Electronic Engineering

FINAL YEAR PROJECT REPORT

BENGU2-CDE-2018/19-Project Code

LD-01

Project Title

Optimal Tuning of Backoff Parameter for Massive Access of Machine-type Devices in LTE Systems

Student Name: lee tsz lok

Student ID: 54409124

Supervisor: Dai Lin

Assessor: HCS - Prof SO, H C

Bachelor of Engineering (Honours) in

Computer and Data Engineering

Student Final Year Project Declaration

I have read the student handbook and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I declare that the work submitted for the final year project does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Student Handbook.

Project Title: Optimal Backoff Tuning For Massive Access of Machine-type Devices in LTE System

Student Name : LEE TSZ LOK	Student ID: 54409124
Signature	Date :

Turnitin Originality Report:

Abstract:

Machine-to-Machine (M2M) communications involve many machine-type devices such as smart watch. In a M2M network, Machine type device (MTD) send its data to a remote server according its event triggered. The progress is without human interface that means MTDs will send its data by itself program.

To support M2M communication, the Long Term Evolution (LTE) system is usually applied because it has a world-wide network coverage. In LTE system, MTDs need to access a LTE base station and then send data. Congestion occur inevitably if many machines send access requests, requiring a connection at the same moment. To relieve the congestion, The Access Class Barring (ACB) scheme can be used, which the ACB factor (i.e acess request transmission probability) is a crucial system parameter that affects the access performance. In this project, a computer-based simulation program is set to simulate the random access procedure of the LTE network. Based on this program, we will study 1) how the ACB factor affects the access performance under different traffic scenarios, and 2) how to optimally tune the ACB factor to maximize the network throughput and minimize the access delay.

Contents

Section I Introduction

M2M communications are becoming more popular since they are expected to apply in ubiquitous domains like e-health. Machine can execute the instruction defined by human. It can be run without human interface. Due its characteristic, it has been applied to a wide range of different fields. For example, smartwatch like iWatch is a machine. It can detect any data wanted and send to remote server. These operations are in autonomous manner. The trend of machine developing is rising sharply.

1.1 Machine type device (MTD)

The characteristic of Machine is vital to its network. It does not rely on the presence of a human interface. Composed by a sensor part, network part and application part. It involves two objects: MTD and server. MTDs have sensor inside. The program in MTDs will be executed forever. Fetch the sensor data and send to remote server. And the data is very small. For example, autonomous cars set many sensors like speed sensor or temperature sensor. There is a server to receive the data from car sensor. Meanwhile the server connects other autonomous cars, it locates the car location and know it situation around the road. Then calculates and sends data to car to control the car speed. In this progress, human intervention does not exist. All thing is finished by server and machine type devices. The sensor or device will execute forever unless it is out of power.

A Human type device is an electronic device with human intervention. Device would not execute program forever. It must be triggered by human-event. Pressing keyboard is an input. Displaying on monitor is the corresponding output. The important difference between MTD and HTD is that MTD will only send small data information when triggered by program or sensor. MTD does not run unessential action for saving power. Therefore, MTDs always give us more convenient than HTD.

1.2 M2M communication in Long-Term Evolution (LTE)system

To support M2M communication, LTE system is used to be the network. Machine requires a reliable network that can transmit data at high data rate and low interference environment. LTE is a good choice because it had already developed for few years. Machine which produced at present day is always equipped latest network technology. It has the newest protocol that can connect LTE network. Machine can be benefited from LTE since LTE can provide a reliable and fast network. Machine can send small data as fast as possible and not suffer crash. M2M always communicate in wireless network. The LTE system protocol is designed for wireless network. This is the reason to use LTE system. In section II, LTE architecture is presented.

1.3 LTE Architecture

LTE has a new Radio Access Network and core network architecture which is presented as Evolved Packet Core.

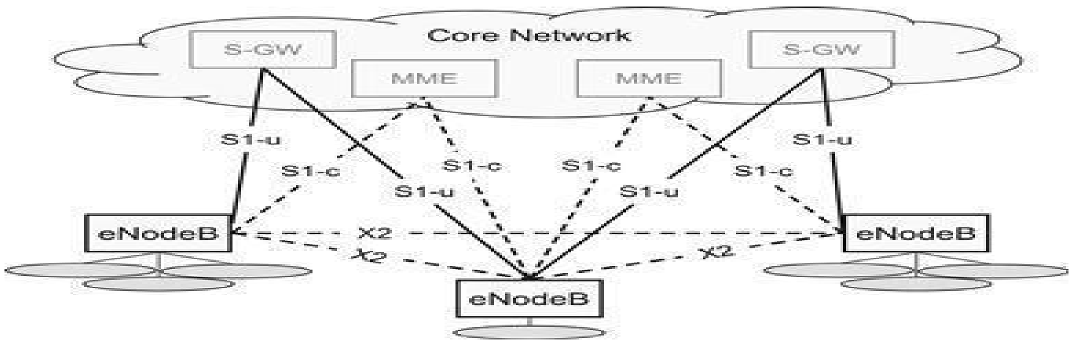


Figure 1 - Radio-Access Network (RAN)

The network involved three parts: S-GW, MME and eNodeB which used in LTE network. Mobility Management Entity (MME) is the manage to control network. It needs to initiate paging and authenticate mobile device which connect its network. MME saves its location information to machines and allocates the correct gateway to machine during the process of initial registration. It has responsibility to transmit signal between three generation mobile network and LTE network.

Serving Gateway (S-GW) is a user-plane node which link to core network to LTE RAN. It is treated as a mobility anchor if terminals shift between eNodeB and for earlier 3G or previous technologies. It will charge by collecting information and statistics essential.

Evolved Based Station (eNodeB) is an evolved telecommunications node in mobile network. The unevolved node B was used in 3G mobile network. It provides connection between mobile phones and telephone network. To associate LTE network, node B was improved its element to be eNodeB. In M2M communication network, eNodeB provides connection between machine and core network. Figure 1 shows that the connection. eNodeB is linked to S-GW by the S1 interface user-plane part, S1-u, and to MME by the S1 control-plane part, S1-c. And the connection to other eNodeBs will be done by X2 interface which often used to support active-mode mobility. The three components compose a Radio Access Network.

1.4 Random access procedure in LTE

Random access is an important process to build a connection between machine and eNodeB. As mentioned in 1.3, eNodeB provides connection between machine and network. Therefore, machine sends data to network through eNodeB. Before machine connects to core network and send data, machine is required to process a random access to eNodeB. Machine sends data only if it had set connection to base station by Random Access. A four-handshake way is implemented since it secures a connection setup. Figure 2 demonstrates the Random Access procedure.

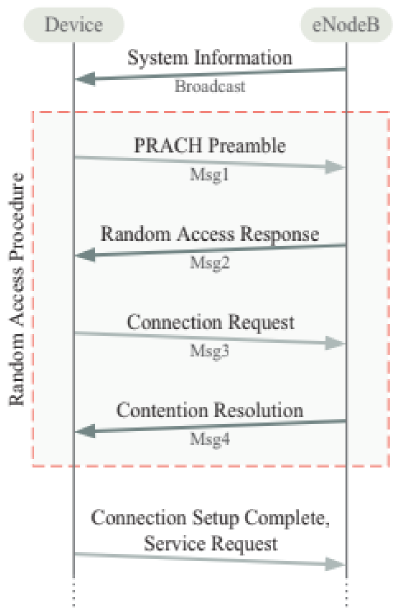


Figure 2 - Random Access procedure in LTE

The four-handshake way procedure step in LTE is follow as:

1. MTDs send a preamble signal which include the information like device’s ID.
2. After eNodeB receive the preamble signal. Then compute and response by RAR which has the ID of device to inform eNodeB is ready.
3. MTD then send connection request to eNodeB
4. eNodeB make the connection and response to MTD
5. Connection complete successfully then MTD could send data if need.

PRACH: a digital signature transmits in the RA. Only 64 timeslots in a RA slot. And 54 preamble signals are used in contention-based which for general. The remaining 10 preambles signals are reserved for special device like emergency.

RAR: After successfully received preamble signals. RAR including PDSCH(Physical Downlink Shared Channel) which have ID of preamble indicated what device it from will be sent.

Collision: In Random Access Procedure, the collision will occur if there are two or more preamble signals sent at the same timeslot. The preamble signals will be declared as invalid. They need to do retransmission next time. It is called as collision.

1.5 Key challenge for M2M communication

M2M communication is facing a problem that the congestion will be serious if number of machines in the network is increasing. The congestion will be caused by many collisions. More collision means that many packets need to be retransmitted until successfully. It is unsatisfactory because it only has very small data and cost time to transmit. It is not effective. The time is wasted in the retransmitting.

1.6 Access Class Barring

To mitigate the collision. ABC scheme has been proposed. In this scheme, a parameter back off factor *q* is selected. First, MTD will draw a number between 0 to 1, if the number is smaller than *q*, the preamble signal will be sent. Otherwise the MTD will stop sending. At next time, MTD can try to retransmit the preamble signal. This parameter is a probability of transmission.

1.7 Objectives

Intuitively, by choosing a smaller q, lesser number of MTDs will send preambles in each timeslot. Therefore, we can expect that the Access Class Barring factor *q* affects the LTE network performance. In this report, three main steps are done to study ACB and finds the performance. And then use the result to find out an optimized performance.

(1) *Establish a simulation program based on MATLAB*
(2) *Evaluate the effect of q with different number of MTDs and the packet arrival rate of each MTD*
(3) *Get optimal factor q in ACB scheme which conducts an optimized network.*

Section III Methodology

In methodology section, it demonstrates how to design a simulation program and variables. First, it is necessary to know the model of Random Access Procedure. It helps us to design the program. Secondly, the flowchart is needed to plan which divide program into each piece of code. Each piece of code could be written one by one then integrate them all. Finally, the program is built and executed to find out result.

3.1 Simulation Random access procedure

A network simulation is required. In this section, it explains the simulation model and its characteristics. In Random Access procedure, preamble signal is sent by machine first. After receiving response from base station, Response will be sent to base station. The collision will occur if there are multiple preamble signals sent at same timeslot. The preamble signals will be invalid. Machines will know their invalid preamble signal if the response does not back. Therefore, Slotted Aloha Network can be used to simulate the M2M network because slotted aloha network characteristics is similar.

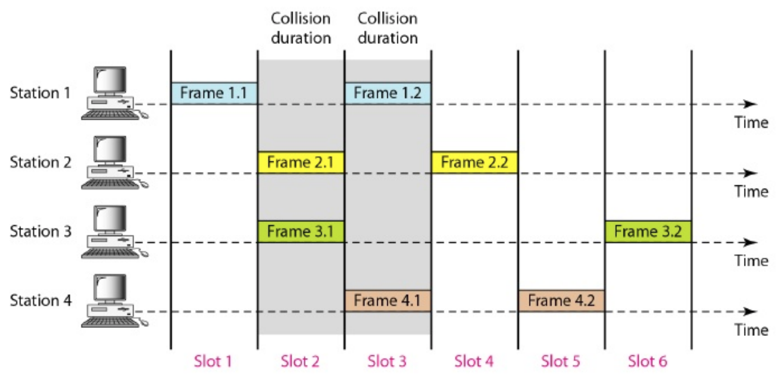
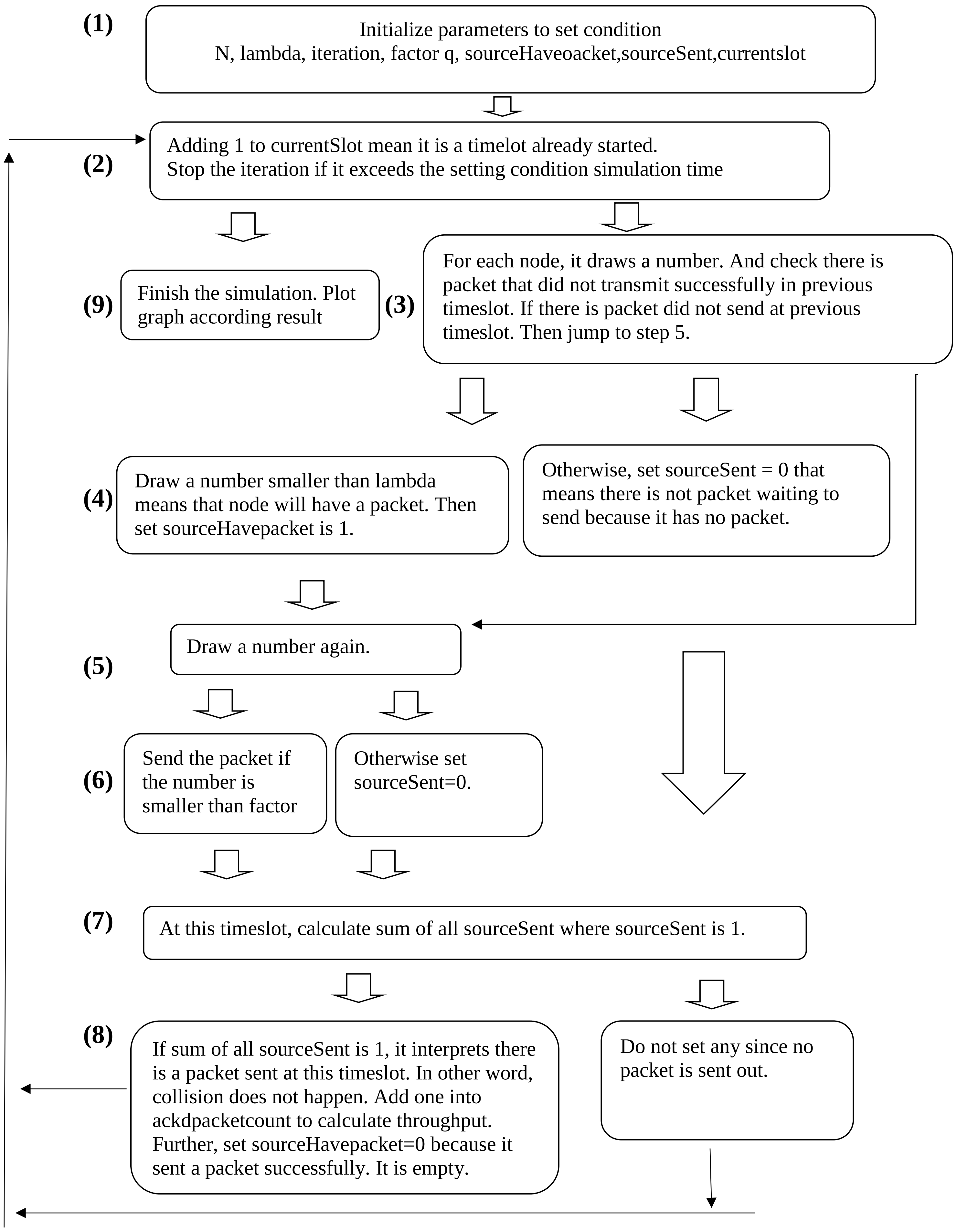


Figure 3 - Slotted aloha network. The length of time is fixed. In slot2 and 3 , collision occur.

In slotted aloha network, node sends packet at each timeslot. The station can be treated as machine in the network and it sends preamble signal to eNodeB. ACB factor *q* will decide the transmission of preamble signal. This project concentrates on the method-Access Class Barring (ACB). In this project, the result is expected to give more detail about relation between parameters and its output.

3.2 Flowchart of the simulation program

In Methodology, a program is designed to simulate the Random Access process. The program plots graph by the data. First, a flowchart needs to be designed which is useful to describe the work of program. Then uses the result to plot a graph and elaborates relation between parameters. Simulation network program is built for collecting data such as optimal parameter and throughput. And MATLAB code for simulation is attached in appendix I-IV. This section shows the flowchart and elaborates its each step. The flowchart is presented in next page.


(1) Initialize parameters to set condition
N, lambda, iteration, factor q, sourceHaveoacket,sourceSent,currentslot

(2) Adding 1 to currentSlot mean it is a timelot already started.
Stop the iteration if it exceeds the setting condition simulation time

(9) Finish the simulation. Plot graph according result

(3) For each node, it draws a number. And check there is packet that did not transmit successfully in previous timeslot. If there is packet did not send at previous timeslot. Then jump to step 5.

(4) Draw a number smaller than lambda means that node will have a packet. Then set sourceHavepacket is 1.

Otherwise, set sourceSent = 0 that means there is not packet waiting to send because it has no packet.

(5) Draw a number again.

(6) Send the packet if the number is smaller than factor

Otherwise set sourceSent=0.

(7) At this timeslot, calculate sum of all sourceSent where sourceSent is 1.

(8) If sum of all sourceSent is 1, it interprets there is a packet sent at this timeslot. In other word, collision does not happen. Add one into ackdpacketcount to calculate throughput. Further, set sourceHavepacket=0 because it sent a packet successfully. It is empty.

Do not set any since no packet is sent out.

There are nine steps to generate result about throughput and optimal factor *q*.

Step (1): Set 7 parameters.

Variable name	Definition
N	The number of Machine Type Devices. Set from 50 to 250.
λ	Packet arrival rate. Set from 0.002 to 0.006.
q	ACB factor *q* of each MTD. Set from 0.025 to 1.
sourceHavepacket	A vector stores each node status that having packet or not. It is determined by λ .
sourceSent	A vector stores each node status that transmit packet or not. It is determined by *q*.
SimulationTime	Timeslot in the network. For enough data, 10^6 is chosen.
Currentslot	The simulation time which already simulated. It is used to check the stop of simulation by using While Loop function. It is initialized as zero.

Step (2): Check the iteration is being stop or continue. In this program, it is expected to simulate 10^6 timeslot. In step 2, use While function to check the loop is going to stop or continue.

Step (3): Draws a number which used to compare with λ and check there is packet did not transmit successfully at previous timeslot. Jump to step 5 if check true.

Step (4): Uses the number drawn in step 3 compare with λ to decide the transmission. Using if-statement to decide the sourceHavepacket vector is true or false.

Step (5): Draws a number which used to compared with factor *q*.

Step (6): After determining sourceHavepacket vector, compares to drawn number in step 5 and factor *q* to decide the transmission. Then sets the sourceSent vector. Otherwise nothing changes until next timeslot.

Step (7): As the definition of collision two or more preamble signals are sent at same timeslot, this step calculates the sum of sourceSent which set true in step 6 to determine there is or not collision.

Step (8): After determining sourceHavepacket vector, then compares to drawn number in step 5 and factor *q* to decide the transmission. Otherwise nothing changes until next timeslot. In step 8, it calculates the throughput, access delay and probability of packet.

Step (9): The step 8 will go back to step 2 and repeat step until the iteration over the timeslot 10^6. The last step 9 stops the iteration and use the result to plot graph about throughput, access delay and, success probability.

3.3 Calculation of performance metrics

In this report, we are interested in the following three performance metrics.

Output	definition
Success probability	The ratio of the number of successful packets to the total number of transmitted packets
Throughput	The ratio of the number of successful packets to the total number of time slot 10^6
Access delay	The time spent from the generation of packet until its successful transmission

According to their definition, three trends are expected: High success probability, High Throughput and Low Access delay.

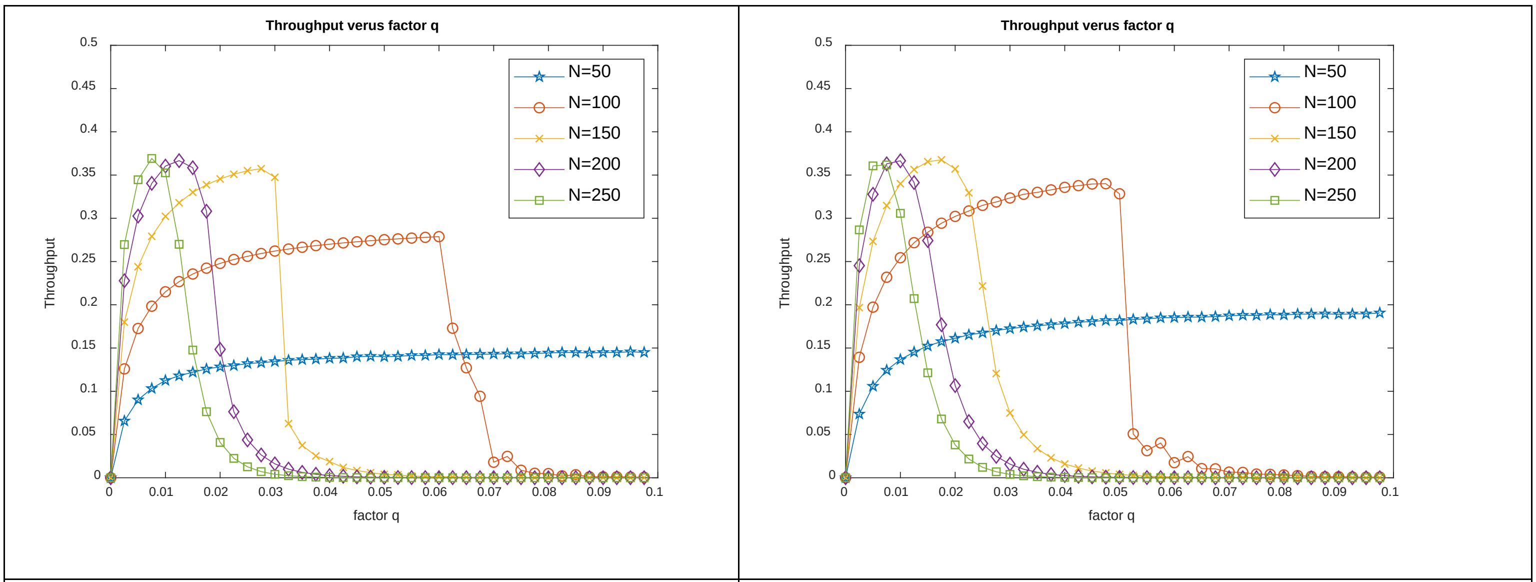
Section IV Simulation Results

In Result section, throughput versus factor *q* is studied first and success probability, finally the access delay is showed. Optimal factor will be conducted by the results.

4.1 Throughput

Throughput is an index to measure the efficiency of the channel. It is expected channel always send packet successfully as possible. The higher throughput imply that more packet successfully transmits at each time slot. Throughput is defined as ratio of number of acknowledged packets to total timeslot. In this sub section, the effect of packet arrival rate λ and number of MTDs N will be studied and look how they affect the optimal factor to achieve the maximum throughput.

4.1.1 Effect of number of MTDs N



(a) (b)

Figure 4 – Throughput versus ABC factor *q* where 4a) λ= 0.003 and 4b) λ= 0.004.

Figure 4 shows throughput with different device number. We can see that the throughput increases when the number of devices increases when *q* is small. Throughput will drop when q is large. For example, at q=0.01, the throughput with N=100 is 0.2, and the throughput with N=150, N=200, N=250 are 0.3,0.35.0.37 respectively. The large *q* which cause congestion obvious, throughput drops sharply. At q=0.01, the throughput with *N*=250 is 0.37, then its throughput drops after q=0.018.

Also, the maximum throughput increases when number of MTDs increases. For instance, in 9a, the highest throughput arrived at q=0.06 with *N*=100 is 0.27. To add 50 device number into network that *N*=150, throughput increases to 0.35 at q=0.03. The throughput increases to 0.37 at q=0.01 when add more 50 device number that *N*=200. It can be observed that however the number of MTDs increases, the highest throughput increases very slowly. The upper-bounded throughput is 0.37 with *N*=200 at q=0.01. It implies that the throughput will not exceed 0.37 if the number of MTDs increases any more in the network. It is because that larger number of devices will enlarge the total number of transmission of packets. The throughput is restricted to 0.37 however the number of devices is larger.

Furthermore, the throughput drops to closed to zero when *q* is large. In figure 4a, the throughput increases stably. In *N*=100, from q=0 to q=0.06, throughput increases. It looks that throughput grows stably and slowly at small *q*. The Maximum throughput is achieved at q=0.06. The throughput drops sharply and closed to zero when the *q* is larger than 0.06. It is because that the collision happens more frequently when the *q* is large. The more packets cannot be transmitted successfully.

In addition, to reach maximum throughput which is optimal throughput, the optimal *q* is reduced if the number of MTDs increases in network. In figure 4a, the throughput with *N*=150 goes its highest throughput 0.35 at factor q=0.03. If more 50 device numbers are added that *N*=200, the throughput *N*=200 goes its highest throughput 0.37 at factor q=0.01. It means that the maximum throughput is reached at reduced *q* when the number of devices is larger. Therefore, smaller optimal factor *q* is required to achieve optimal throughput to avoid more frequent collision in M2M communication.

4.1.2 Effect of packet arrival rate λ of each MTD

It is observed that lines are compressed if λ increases. Their throughput is higher and factor q is smaller. This proves that factor q will be lesser if λ is higher. Meanwhile throughput will be taller too.

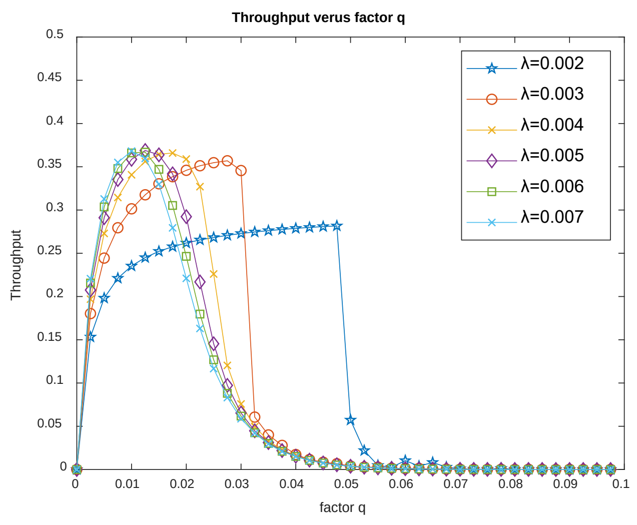
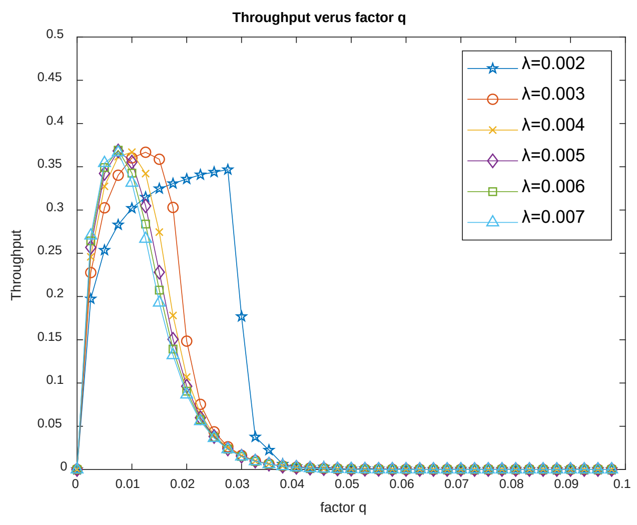
	
(a)	(b)

Figure 5 – Throughput versus factor q at different λ from 0.002 to 0.007 where 5a) N=150, 5b) N=200

Figure 5 illustrated that the increased λ leads higher maximum throughput. In 5a, the maximum throughput with λ=0.002 is 0.27 at q=0.06. If the λ increases to 0.003, its maximum throughput increases to 0.36 at q=0.03. The maximum throughput increases to 0.37 while λ increases to 0.007 at q=0.018. We can say that maximum throughput will be higher if λ increases. But the highest throughput is limit to 0.37 if λ is higher more. It indicates the maximum throughput is 0.37 however the λ increases. In figure 5a,5b, the high λ which are 0.005, 0.006 and 0.007. The maximum throughput is still limit to 0.37. It implies that the throughput only can be maximized as 0.37 which defined as upper throughput.

Moreover, throughput drops to very low closed to zero when the q is larger. The throughput raises slowly until its optimal factor. In figure 5a, the line λ=0.002 goes to maximum throughput at q=0.05 stably. The sharp fall occurs after the optimal factor. Throughput drops to very low when the q is larger than q=0.05. It is because that the collision becomes very serious that most packets cannot be transmitted successfully.

Meanwhile, optimal *q* is reduced when *λ* increases. For instance, in 5a, the optimal factor is 0.06 with *λ*=0.002 where the throughput is the highest. The factor *q* is reduced to 0.03 at *λ*=0.003. The line *λ*=0.004 goes its maximum throughput at q=0.018 which is smaller than previous. It is because that more *λ* will generate more packets. Hence the optimal factor *q* is required to be smaller to avoid the frequent collision when *λ* increases.

4.1.3 Summary

We can see that the throughput will be affected by number of MTDs *N* and packet arrival rate *λ* of each MTD. First, their throughput will increase at small *q* range such as from q=0 to 0.04. It is because the small *q* from 0-0.04 will not cause collision very often. The number of collisions is acceptable. Hence the throughput increases. But the collision occurs very often when *q* is quite large. The preamble congestion becomes obvious. The throughput will decrease. Therefore, it can be observed that both graphs drop rapidly when *q* is large.

And, maximum throughput will be higher while the number of MTDs and *λ* increase. The upper throughput would be 0.37 however number of MTDs and *λ* increase. From figure 4 and 5, the increased number of MTDs *N* and packet arrival rate *λ* cause higher throughput. It is because that larger *N* and *λ* will generate more packets on average. The total successful transmission will increase. But the throughput does not increase more if the number of devices *N* or *λ* is larger.

The throughput drops and becomes very small as the factor *q* increases at effect of *N* and *λ*. We can see that the throughput rises to maximum where the *q* is optimal. Then the throughput drops sharply. It is because that the number of devices *N* and packet arrival rate *λ* will increase the packets on average. In large *q*, the collision happens very frequent. The network cannot endure much collision. Hence the throughput drops and becomes very small after the large *q*.

Finally, the optimal *q* is reduced as the effect of number of MTDs and packet arrival rate *λ*. To reach highest throughput that optimal throughput, it needs to go a specific *q* point which is restricted by *λ* and number of MTDs. From result, optimal *q* decreases while the *λ* and number of MTDs increase. It is because that the smaller optimal *q* can mitigate the number of collisions when *N* and *λ* increase.

4.2 Success probability

Success probability is a probability of success transmission of packets. It measures the probability of successful transmission in each timeslot. The higher probability means that the packet more chance to transmit successful. The number of retransmissions could be lower. In this section, the effect of λ and N will be found and look how they affect the optimal factor to achieve the maximum success probability.

4.2.1 Effect of number of MTDs N

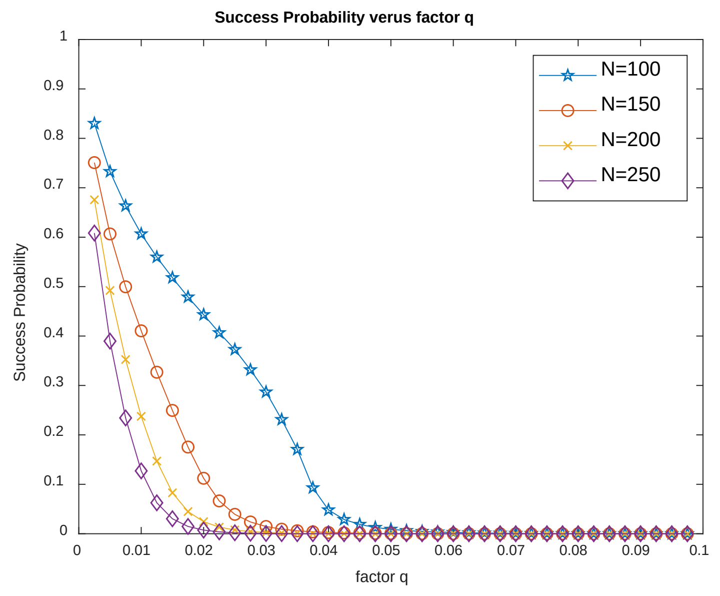


Figure 6 – Success probability versus factor q at λ=0.006

It can be observed that every line success probability will decrease when the factor q increases in figure 6. For N=100, the success probability is the highest at start. Then drops when factor q increases. The success probability will go to very low to zero if q still is increasing. It is because that the number of collisions will increase when factor q increases. The failed transmission increases inevitably. Hence the success probability decreases at all factor q.

Also, the success probability will be lower at same factor q when device number increases. For example, the probability is 0.6 at factor q=0.01 when N=100. When increases more 50 device number that N=150, the success probability is reduced to 0.55 at the same factor q=0.01.

4.2.2 Effect of packet arrival rate λ of each MTD

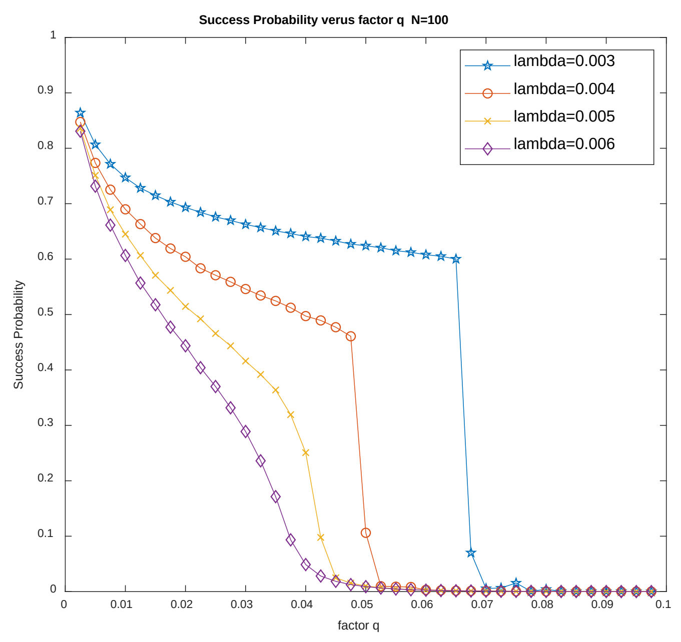


Figure 7 – Success Probability versus factor *q* with different λ and fixed *N*=100

If λ increases, it indicates more packet will be generated. The success probability will decrease as λ increases. In figure 7, it proves this hypothesis. For λ=0.003, its success probability drops slowly from q=0.0 to 0.07 and drops sharply when q=0.07. After q=0.07, the success probability is close to 0. For λ=0.004, the success probability drops from q=0 to q=0.05. At this graph 7, the success probability always is lower when λ increases. Thus, the increased λ leads the success probability lower.

4.2.3 Summary

For success probability, it is found that the success probability always decreases because the failed transmission increases inevitably.

Since number of collisions always increases when number of devices *N* and λ increase, the success probability always be lower. We can see that the success probability will not keep the optimal performance or go to optimal performance when changes the factor *q*. The optimal *q* always is 0.001 if we consider the highest success probability however the number of MTDs and number of λ.

4.3 Access Delay

Access delay measures the time separation between the generation of packet until its successful transmission. Acknowledged packet means the data is received successfully. A good network should equip with low access delay. Packet can be transmitted successfully as soon as possible. They do not cost time to do retransmission. It also presents as LTE network-reliable and fast network. In this section, it can observe how *N* and *λ* affect access delay. At optimal factor, access delay should be as low as possible. If access delay is high, it implies machines have to spend time to retransfer.

4.3.1 Effect of number of MTDs

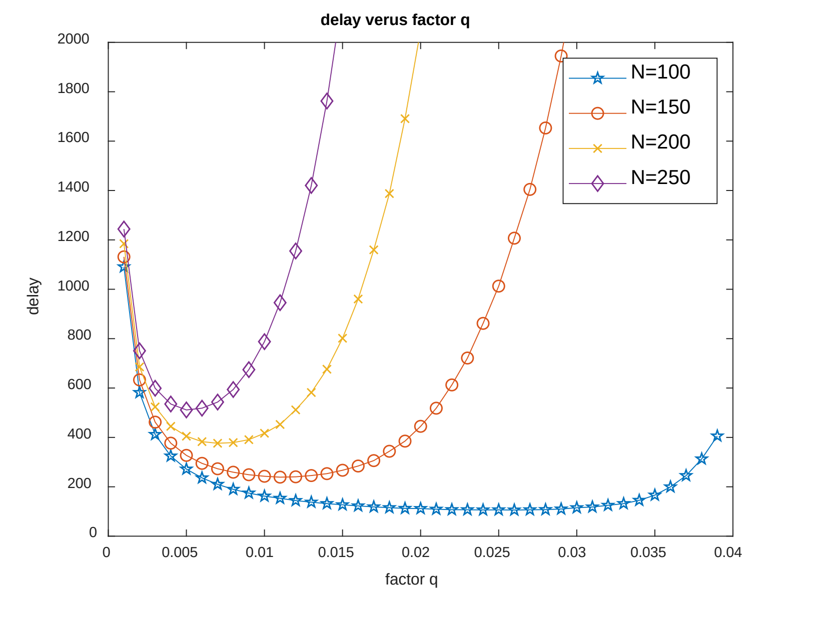


Figure 8 – Access delay versus factor *q* with different *N* and fixed *λ*=0.006

Figure 8 shows different device number and their corresponding access delay. We can see that access delay increases when number of MTDs increases. For example, in figure 8, the access delay is 200 at q=0.006 with *N*=100. With larger number of devices that *N*=250, its access delay is 600 at the same q=0.006. In figure8, at same *q* the access delay is higher if the number of devices is larger.

We can find that access delay is high when *q* is small and large. For example, the access delay of *N*=250 delay is higher after q=0.005 and before q=0.005. It is because that the *q* has different meaning at small *q* and large *q*. In small q, it means that the packet has very low probability to be transmitted. It is hard to send the packet at this very low factor *q*. Thus, the high access delay is caused. The access delay is minimized to 500 at q=0.005. The q=0.005 can let packet be transmitted soon. In large q, the *q* looks very large. Packets can be transmitted at high probability. More packets are sent to server. As a result, collision happens frequently. Packets need to do retransmission until the packets successfully sent. Consequently, the access delay is higher. Hence the access delay increases large q>0.005.

Also, minimum access delay increases when number of MTDs increases. In figure 8, the access delay with *N*=100 reaches its lowest delay 100 at q=0.02. To look *N*=150, its lowest access delay is 300 at q=0.012. Furthermore, *N*=200 access delay goes to 400 which is more than *N*=100 and *N*=150. It is because that the larger number of MTDs increases the total packets on average. The access delay increases inevitably. Hence, the larger number of MTDs will increase the minimum access delay.

Additionally, optimal *q* is reduced for optimal access delay if number of MTDs increases comparing with the original optimal q. To compare with *N*=100 and *N*=150, the optimal *q* is reduced to q=0.012 from q=0.02 for minimum access delay. Their access delay rises after optimal q. In figure 8, the access delay with *N*=200 moves to its lowest delay 400 at q=0.007 which is optimal *q* and minimum access delay. It is expected that optimal *q* will decrease if the number of MTDs grows. Let see *N*=250 in figure 8, the access delay attains minimum at q=0.006 which is smaller than *N*=200 at q=0.007. It is because that the increasing of number of MTDs will generate more packets on average. The collision will happen more frequent. To avoid the numerous collisions. Optimal factor *q* is reduced to achieve optimal access delay if the number of devices increases.

4.3.2 Effect of packet arrival rate λ of each MTD

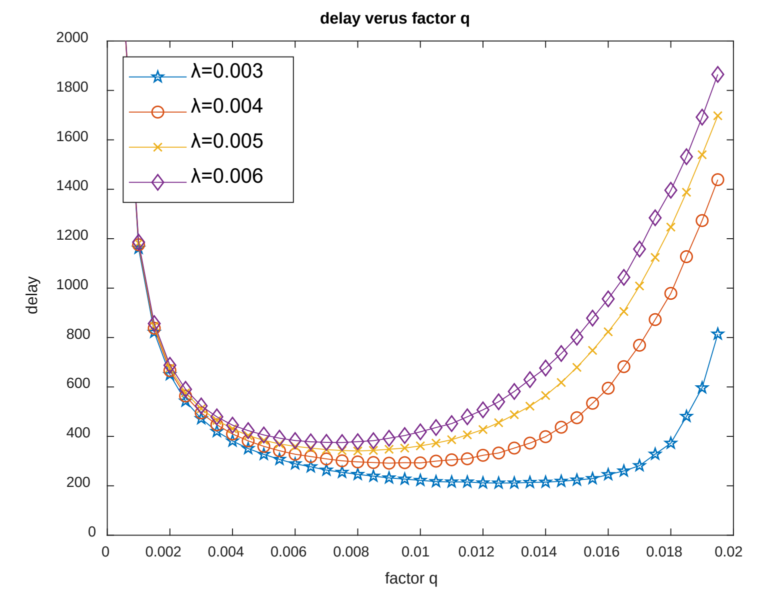


Figure 9 – Access delay versus factor *q* with different *λ* and fixed *N*=200

An assumption that more packet generation will make more collision can predict access delay will be higher if *λ* increases. Because the collision causes many retransmissions of packets. The average transmission time will be longer that access delay will be longer too. To verify this assumption, figure 9 illustrates this phenomenon. In figure 9, the access delay with higher *λ* always be larger. It can be seen the *λ*=0.003 access delay is always lower than other *λ* which are higher than *λ*=0.003. The *λ*=0.006 highest *λ* has the highest access delay at all factor *q*. It is because that the larger packet arrival rate *λ* generates more packets. The collision will increase. Therefore, higher *λ* will lead higher access delay.

We can find that access delay is high when *q* is small and large. For example, in figure 9, the access delay of *λ*=0.004 is higher after q=0.01 and before q=0.01. It is because that the *q* has different implication at small *q* and large *q*. In small q, it means that the packet has very low probability to be transmitted. It is hard to send the packet at this very low factor *q*. Thus, the high access delay is caused. The access delay is minimized to 380 at q=0.01. The q=0.01 can let packet be transmitted soon. In large q, the *q* looks very large. Packets can be transmitted at high probability. More packets are sent to server. As a result, collision happens frequently. Packets need to do retransmission until the packets successfully sent. Consequently, the access delay is higher. Hence the access delay increases large q>0.01.

Minimum access delay increases when λ increases. In figure 9, the access delay with λ=0.003 reaches lowest delay 200 at q=0.015. To look λ=0.004, the lowest access delay is 300 at q=0.01. Furthermore, λ=0.005 access delay goes to 400 which is larger than λ=0.003 and λ=0.004. It is because that the higher packet arrival rate λ increases the total packets on average. The access delay increases inevitably. Hence, the larger packet arrival rate λ will increase the minimum access delay.

On top of that, optimal *q* becomes smaller to achieve lowest access delay when λ increases. In figure 9, the optimal *q* is 0.015 if we want minimum access delay with λ=0.003. λ=0.004 attains its lowest access delay at q=0.009. We can explain that the optimal *q* is reduced because the larger λ causes more collision. It is necessary to adjust transmission probability for congestion control.

4.3.3 Summary

The both effect of number of MTDs and λ cause identical appearance of access delay. First, the access delay is higher if they increase. The bigger number of MTDs and packet arrival rate λ imply that there will be more packet. It could be seen from figure 8 and 9. At same *q*, larger number of devices or λ generate more packets. The packets wait to be sent under same transmission probability. Collision will occur more frequently. Hence access delay will be higher.

And, the minimum access delay will be higher while the number of MTDs and λ increase. It is because that the number collisions will be more. It causes the higher access delay including its minimum access delay.

Access delay is high when *q* is either too small or large in both effects. The access delay is high when its *q* is smaller than or larger than optimal point. Because the effects will increase the average number of packets. At the beginning of q, the packets wait to be sent but very low probability like q=0.001. The successful transmission is finished with low opportunity, the fail transmission should be more than successful case. Then the retransmission often happens. Thus, the access delay is higher at start. Then increases *q* until the minimum delay. After the optimal *q*, the low probability becomes high probability that causes more collision. That is why access delay is high at small and large q.

Finally, optimal *q* is reduced as effect of number of MTDs and *λ*. To reach lowest access delay. It needs to go a specific *q* point which is restricted by *λ* and number of MTDs. From result, smaller *q* is needed while the *λ* and number of MTDs increases because smaller *q* could control the flow of transmission and avoid numerous collisions.

4.1 Optimal q for throughput maximization

In result of throughput versus lambda and device number plot, the explanation is that optimal factor will not stay at identical level. Factor will be changed according the environment condition. In other word, in M2M communication, the number of devices and packet arrival rate *λ* affect the result. Therefore, optimal factor needs to be tuning. It is expected to change factor *q* automatically by any method. For example, Machines receiving a small broadcast which include the number of devices in the LTE network and attempting to access same server. The optimal factor is found in result in section 3. In this section, we would like to find how can perform optimal backoff tuning at two aspects: Optimal *q* for maximizing throughput and minimizing access delay.

In this section, the optimal *q* is chosen in throughput effect section. The optimal *q* will be recorded and updated by using MAX function in program when the throughput achieves higher throughput. Until the end of simulation, the highest throughput and corresponding optimal factor *q* will be stored. Similarly, the lowest access delay will be recorded by using MIN function in program. Finally, the optimal *q* is stored in array and used to simulate the network again.

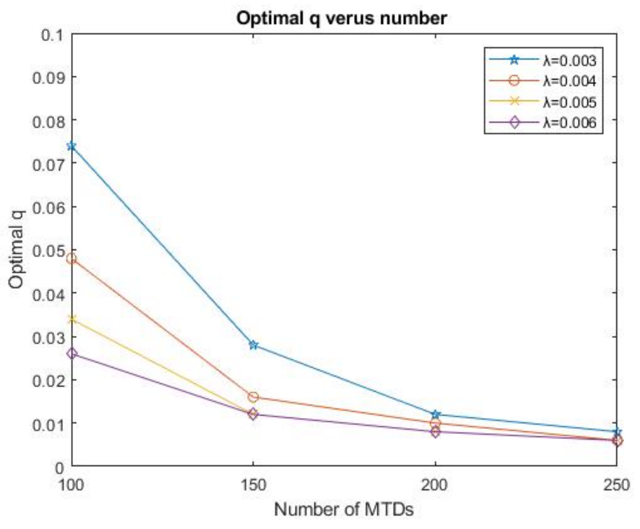


Figure 10 – Optimal factor versus the number of MTDs with λ = 0.003,0.004,0.005 or 0.006.

Figure 10 tells us the optimal q which maximize the throughput with different λ and number of MTDs. For example, if there are 100 MTDs in the network and the packet arrival rate λ is 0.006. Each machine type device can choose the 0.028 as optimal q. Result section conducted that optimal factor q is reduced to achieve optimal throughput if the number of devices increases. Compare with figure 10, their optimal q is reduced as the number of MTDs increases. If the LTE network has larger number of MTD, the optimal q needs to be adjusted. For example, optimal q is adjusted to q=0.015 when the number of MTDs increases to 150 from 100 at same λ=0.006. The optimal q is decreased from 0.028 to 0.015. Figure 10 offers situation under λ=0.003 to λ=0.006. These data could be stored in MTDs and then MTDs choose optimal q according their condition of λ and number of MTDs, so the throughput could be optimized.

However, the λ and number of MTDs increase, we can find out their optimal q trends to be smaller which always be 0.01. In graph 10, the λ changes from 0.003 to 0.006. Their optimal goal still is 0.01 where N=250. And the optimal q still is 0.01 if the number of MTDs exceeds 250. It implies that setting q=0.01 could be a preferable choice when either N or λ is large.

4.2 Optimal q for access delay minimization

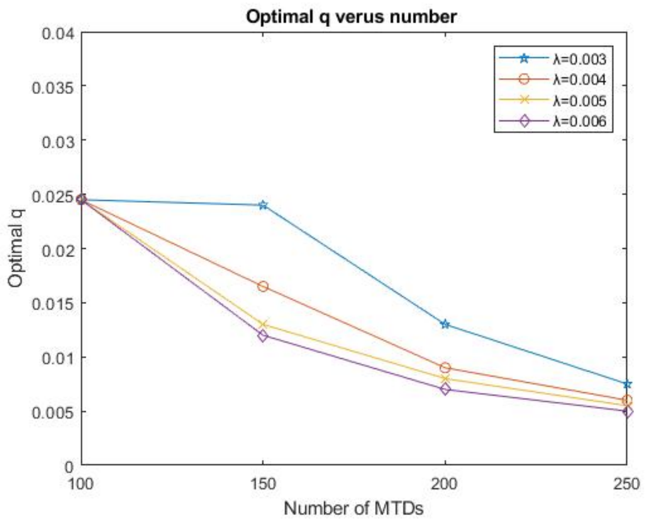


Figure 11 – Optimal factor versus the number of MTDs with λ = 0.003,0.004,0.005 or 0.006.

Figure 11 indicates that the optimal q decreases if number of MTDs and λ increase. We can observe that the optimal q is 0.02 if the network involved MTD number is 100. It does not change noticeably when the number of MTDs changes to 150. But the change becomes clear at N=200. It drops to 0.014. If λ is enlarged, the changed q becomes more obvious. It decreases to 0.018 from 0.02 when λ=0.004. In figure 11 where λ is higher, their optimal q similarly drops. For larger number of MTDs, the drop occurs in all λ. Since the larger λ and number of MTDs will generate more packet, the successful transmission will take longer time. Therefore, optimal q is reduced for more successful transmission.

However, the λ and number of MTDs increase, their optimal q trends to be smaller which always be 0.005. In figure 12, the λ changes from 0.003 to 0.006. Their optimal goal still is 0.005 where N=250. And the optimal q still is 0.005 if the number of MTDs exceeds 250. It tells us the optimal q set 0.005 preferably when either N or λ is large.

4.3 Performance Comparison

In above figures, they show that optimal factor *q* decreases while the number of MTDs increases. Hence the performance should be kept in optimized level with the decreased optimal factor *q*. And it is compared with fixed factor *q*. We can see their performance between using optimal *q* and fixed *q*.

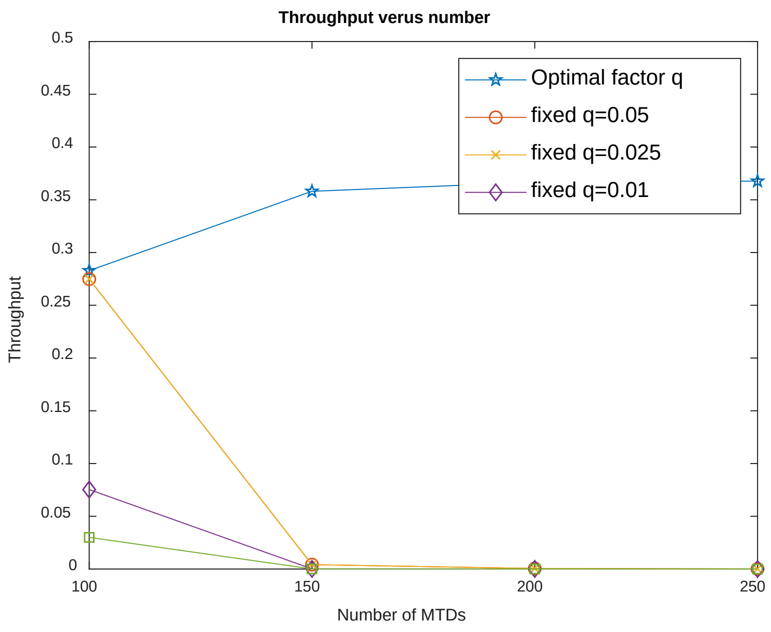


Figure 12 –The performance of throughput versus number of MTDs with λ=0.003

The graph shows lines which are throughput with optimal factor *q* and throughput with fixed factor *q*. We can see that using optimal factor can keep the highest throughput at different number of MTDs. The initial factor *q* at *N*=100 is 0.075 which can be obtained in figure 10. In figure 12, the fixed *q* line does not change if the number of MTDs is larger, then throughput will drop and become very low. This problem could be improved by adjusting the factor *q* parameter according the number of MTDs. As mentioned in 4.1, optimal factor q is reduced to optimize throughput. Hence, we can see that the throughput is optimized as 0.35 at *N=150* where the factor q is changed to 0.03. And throughput in *N*=200 *or N*=250 is the highest point while the optimal factor *q* changes into smaller which are 0.018 and 0.012. We can see that the fixed *q* lines degradation is serious. Their throughput still drops and become very low because the factor *q* is fixed. It tells us that the changing of factor *q* is very important when the performance of throughput needs to maximize the throughput.

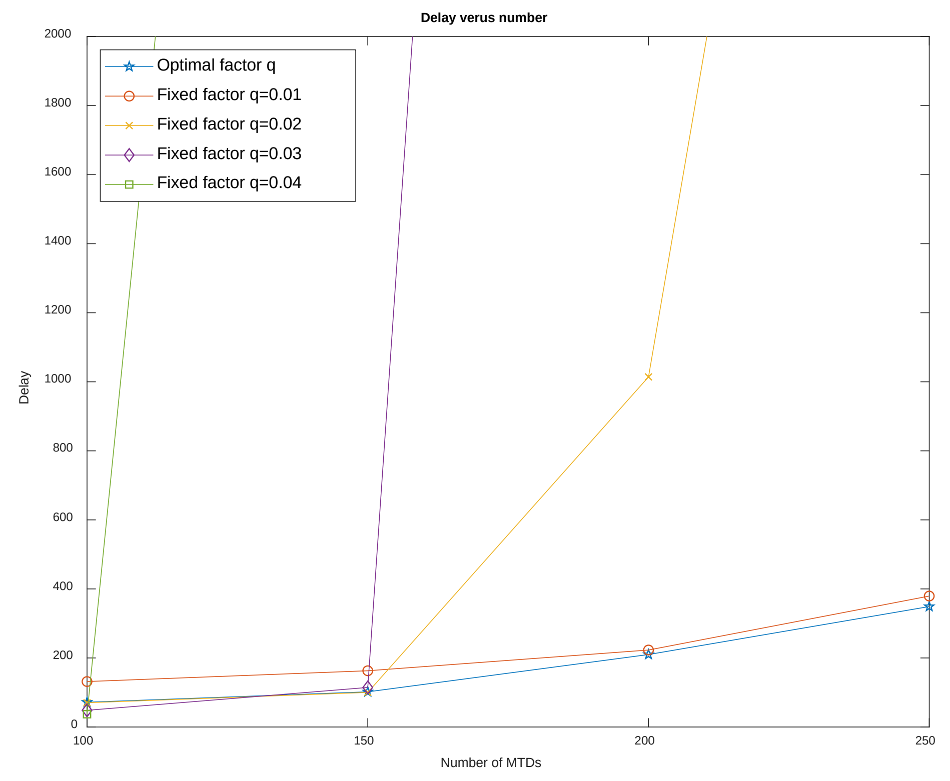


Figure 13 –The performance of access delay versus number of MTDs with λ=0.003

In figure 13, it presents lines access delay with changing optimal factor *q* and fixed factor *q*. At start *N*=100 and *N*=150, the lower factor *q* such as *q*=0.01 and *q*=0.02 access delay is very low about 150. The optimal factor initializes as 0.025 which could be found in figure 11. We can observe that the access delay increases very much if using fixed *q*. The access delay will increase sharply when the number of MTDs is larger although the fixed *q* is very low. For example, the fixed *q*=0.02 access delay sharply increases to 1000 from 150 at N=150 to N=200. The fixed factor *q* line access delay becomes very high. It is a negative effect. If the optimal factor *q* is used that always be tuned, the access delay does not increase much like fixed factor *q*. The negative effect is improved. From the optimal factor *q* line, its access delay only increases few. At *N*=200 and *N*=250, the access delay only increases from 120 to 380. So, the preamble signal could transmit in a low access delay network when the factor *q* is changed according the number of MTDs.

Section V Conclusion

In this project, the Random Access performance of M2M communications in LTE networks is evaluated.

We build a simulation program based on MATLAB to simulate the Random access procedure of the LTE system, where the ACB scheme is incorporated. We consider a network scenario with the number of MTDs N and the packet arrival rate of each device λ. Through the simulation results, we found out that how they affect the network performance and how to optimize network by properly choosing the ACB factor q.

(1) The performance can be affected positively or negatively by the number of MTDs N and λ. Three aspects will be considered that are throughput, success probability and access delay.

- (1.1) Throughput: Throughput will be higher if either number of MTDs or λ increases because they will generate more average preamble signals in each time slot. But the throughput will not rise again when the number of MTDs N or λ becomes larger. The maximum throughput would be around 0.37 no matter the number of MTDs N or λ is larger.
- (1.2) Success probability: Success probability is expected to decrease since larger number of MTDs N or the packet arrival rate of each MTD λ would produce more average preamble signal at each timeslot. The collision would occur more frequent. Because of the collision, retransmission should be taken more. Hence the success probability if the number of MTDs or λ grows.
- (1.3) Access delay: The minimum access delay will be higher while the number of MTDs N or λ increases. It is because there is more retransmission that the time between generation of preamble signal to successful transmission is higher.

(2) The ACB factor q, *i.e, the* transmission probability significantly affects performance. Comparing with throughput and success probability aspect, the affect is different.

(2.1) Throughput: The throughput increases at start q is very small. After increase q, the preamble signal could be transmitted in reasonable probability. That why the throughput rises if increases the q in small range. But the throughput will drop if q goes to a large q. In here, the number of collisions is numerous and makes congestion. Therefore, the throughput decreases at large q.

(2.2) Success probability: It will be affected negatively by q. When the q increases, the number of collisions increases. The number of successful transmissions must be downgraded. Hence the success probability will drop if the q increases.

(2.3) Access delay: The access delay declines at start until optimal q and rises after this optimal q. At start, the preamble signal is transmitted under very low probability q so the access delay is quite high. When increases q in small range, more preamble signal could be sent successfully so the access delay is lower. After the optimal q, the number of collisions is very large. The congestion becomes noticeable and increases the access delay.

(3) The optimal q is reduced if either number of MTDs N or λ increases in throughput and access delay part.

(3.1) Throughput: At small number of MTDs or λ like N=50, the q is not very low because the number of preamble signals on average is not large. It is unessential to narrow the q to control congestion, So the optimal q can be larger at small N or λ. If either λ or number of MTDs increases that the number of preamble signals increases, the optimal q is reduced to control rate of retransmission to maximize throughput.

(3.2) Access delay: The higher λ and number of MTDs N will increase probability of collision. The access delay will be higher. To avoid frequent collision, smaller optimal q is chosen.

Appendix I: Slotted aloha network simulation.m

```
function [probability,throughput,delay] =
slottedalohanetwork(sourceNumber,lamda,simulationTime,acbfactor)
sourceSent = zeros(1,sourceNumber);
sourceHavepacket = zeros(1,sourceNumber);
currentSlot = 0;
ackdPacketCount=0;
slotcurrent=zeros(1,sourceNumber);
totalslot=0;
generatedPacketCount=0;

while currentSlot < simulationTime
    currentSlot = currentSlot + 1;

    for i = 1:sourceNumber
        if  rand <= lamda && sourceHavepacket(1,i)==0
            slotcurrent(1,i)=currentSlot;
            sourceHavepacket(1,i)=1;
            if rand <= acbfactor
                sourceSent(1,i) = 1;
            generatedPacketCount=generatedPacketCount+1;

            else
                sourceSent(1,i) = 0;

            end

        else

            sourceSent(1,i) = 0;
        end
        if  sourceHavepacket(1,i)==1

            sourceHavepacket(1,i)=1;

            if rand <= acbfactor
                sourceSent(1,i) = 1;
                 generatedPacketCount=generatedPacketCount+1;


            else
                sourceSent(1,i) = 0;

            end

        else

            sourceSent(1,i) = 0;
        end

    end
    if sum(sourceSent == 1) == 1
        ackdPacketCount = ackdPacketCount + 1;
        for i = 1:sourceNumber
```

```
            if  sourceSent(1,i)==1
                sourceHavepacket(1,i)=0;
                totalslot=totalslot+(currentSlot-slotcurrent(1,i));

            end
        end

    end

end

probability=ackdPacketCount/generatedPacketCount;
throughput = ackdPacketCount /currentSlot;
delay=totalslot/ackdPacketCount;
probability
throughput
delay
```

Appendix II:plotthroughput.m

```
iteration=50;
number0=50;
number1=100;
number2=150;
number3=200;
number4=250;
lambda=0.005;
lambda0=0.002;
lambda1=0.003;
lambda2=0.004;
lambda3=0.005;
lambda4=0.006;
lambda5=0.007;
lambda6=0.008;
acbincrease=0.002;
acbfactor=zeros(1,iteration);
acbfactor(1,1)=0.0;
y0=zeros(1,iteration);
z0=zeros(1,iteration);
y1=zeros(1,iteration);
z1=zeros(1,iteration);
y2=zeros(1,iteration);
z2=zeros(1,iteration);
y3=zeros(1,iteration);
z3=zeros(1,iteration);
y4=zeros(1,iteration);
z4=zeros(1,iteration);
y5=zeros(1,iteration);
z5=zeros(1,iteration);
x0=zeros(1,iteration);
number=[100,150,200,250];


for loop = 1:iteration

    fprintf('This is 1 of %d th function \n',loop);
    [~,throughput0,~] =
slottedalohanetwork(100,lambda,10^6,acbfactor(1,loop));
      y0(1,loop)=throughput0;

    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end
for loop = 1:iteration

    fprintf('This is 2 of  %d th function \n',loop);
    [~,throughput1,~] =
slottedalohanetwork(150,lambda,10^6,acbfactor(1,loop));
      y1(1,loop)=throughput1;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end
for loop = 1:iteration
```

```
    fprintf('This is 3 of  %d th function \n',loop);
    [~,throughput2,~] =
slottedalohanetwork(200,lambda,10^6,acbfactor(1,loop));
     y2(1,loop)=throughput2;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end
for loop = 1:iteration

    fprintf('This is 4 of %d th function \n',loop);
    [~,throughput3,~] =
slottedalohanetwork(250,lambda,10^6,acbfactor(1,loop));
        y3(1,loop)=throughput3;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end



% this figure is for throughput versus factor
%  figure;
%  plotgraph=plot(acbfactor,y0,"-p",acbfactor,y1,"-o",acbfactor,y2,"-
x",acbfactor,y3,"-d");
%  ylim([0,0.5]);
% title('Throughput verus factor q ')
%  xlabel('factor q')
%  ylabel('Throughput')
%  legend('N=100',"N=150","N=200","N=250")
%
%

% this figure is for optimal factor
[M0,I0]=max(y0);
[M1,I1]=max(y1);
[M2,I2]=max(y2);
[M3,I3]=max(y3);
Optimalq=[acbfactor(1,I0),acbfactor(1,I1),acbfactor(1,I2),acbfactor(1,I3)];
 figure;
 plotgraph=plot(number,Optimalq,"-p");
 ylim([0,0.1]);
title('Optimal q verus number ')
 xlabel('Number of MTDs')
 ylabel('Optimal q')
 legend('£f=0.005')
```

Appendix III :plottdelay.m

```
iteration=50;
number0=50;
number1=100;
number2=150;
number3=200;
number4=250;
lambda0=0.002;
lambda1=0.003;
lambda2=0.004;
lambda3=0.005;
lambda4=0.006;
lambda5=0.007;
lambda6=0.008;
acbincrease=0.0004;
acbfactor=zeros(1,iteration);
acbfactor(1,1)=0.000;
y0=zeros(1,iteration);
z0=zeros(1,iteration);
y1=zeros(1,iteration);
z1=zeros(1,iteration);
y2=zeros(1,iteration);
z2=zeros(1,iteration);
y3=zeros(1,iteration);
z3=zeros(1,iteration);
y4=zeros(1,iteration);
z4=zeros(1,iteration);




for loop = 1:iteration

    fprintf('This is 1 loop of %d th function \n',loop);
    [~,~,delay0] = slottedalohanetwork(100,0.006,10^6,acbfactor(1,loop));
    z0(1,loop)=delay0;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end


for loop = 1:iteration

    fprintf('This is 2 loop of %d th function \n',loop);
    [~,~,delay1] = slottedalohanetwork(150,0.006,10^6,acbfactor(1,loop));
     z1(1,loop)=delay1;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end


for loop = 1:iteration

    fprintf('This is 3 loop of %d th function \n',loop);
    [~,~,delay2] = slottedalohanetwork(200,0.006,10^6,acbfactor(1,loop));
```

```
     z2(1,loop)=delay2;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end


for loop = 1:iteration

    fprintf('This is 4 loop of %d th function \n',loop);
    [~,~,delay3] = slottedalohanetwork(250,0.006,10^6,acbfactor(1,loop));
      z3(1,loop)=delay3;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end




% this figure for delay versus factor
% figure;
% plotgraph=plot(acbfactor,z0,"-p",acbfactor,z1,"-o",acbfactor,z2,"-
x",acbfactor,z3,"-d");
% ylim([0,2000]);
% title('delay verus factor q')
% xlabel('factor q')
% ylabel('delay')
% legend('lambda=0.003','lambda=0.004','lambda=0.005','lambda=0.006')

% this figure for optimal factor
[M0,I0]=min(z0);
[M1,I1]=min(z1);
[M2,I2]=min(z2);
[M3,I3]=min(z3);
Optimalq=[acbfactor(1,I0),acbfactor(1,I1),acbfactor(1,I2),acbfactor(1,I3)];
 figure;
 plotgraph=plot(number,Optimalq,"-p");
 ylim([0,0.04]);
title('Optimal q verus number ')
 xlabel('Number of MTDs')
 ylabel('Optimal q')
 legend('£f=0.006')
```

Appendix IV :plottdelay.m

```
iteration=100;
acbincrease=0.001;
acbfactor=zeros(1,iteration);
acbfactor(1,1)=0.0;
y0=zeros(1,iteration);
z0=zeros(1,iteration);
y1=zeros(1,iteration);
z1=zeros(1,iteration);
y2=zeros(1,iteration);
z2=zeros(1,iteration);
y3=zeros(1,iteration);
z3=zeros(1,iteration);
y4=zeros(1,iteration);
z4=zeros(1,iteration);
y5=zeros(1,iteration);
z5=zeros(1,iteration);
x0=zeros(1,iteration);
x1=zeros(1,iteration);
x2=zeros(1,iteration);
x3=zeros(1,iteration);
number=[100,150,200,250];


for loop = 1:iteration

    fprintf('This is 1 loop of %d th function \n',loop);
    [probability0,~,~] =
slottedalohanetwork(100,0.006,10^6,acbfactor(1,loop));
    x0(1,loop)=probability0;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end

end

for loop = 1:iteration

    fprintf('This is 2 loop of %d th function \n',loop);
    [probability1,~,~] =
slottedalohanetwork(150,0.006,10^6,acbfactor(1,loop));
    x1(1,loop)=probability1;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end

end
for loop = 1:iteration
    fprintf('This is 3 loop of %d th function \n',loop);
    [probability2,~,~] =
slottedalohanetwork(200,0.006,10^6,acbfactor(1,loop));
    x2(1,loop)=probability2;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
```

```
end
for loop = 1:iteration

    fprintf('This is 4 loop of %d th function \n',loop);
    [probability3,~,~] =
slottedalohanetwork(250,0.006,10^6,acbfactor(1,loop));
    x3(1,loop)=probability3;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end

end







% this figure for success prob versus factor
%  figure;
% plotgraph=plot(acbfactor,x0,"-p",acbfactor,x1,"-o",acbfactor,x2,"-
x",acbfactor,x3,"-d");
%  ylim([0,1]);
% title('Success Probability verus factor q  N=100')
%  xlabel('factor q')
%  ylabel('Success Probability ')
%  legend('lambda=0.003',"lambda=0.004","lambda=0.005","lambda=0.006")

% this figure for optimal factor
[M0,I0]=max(x0);
[M1,I1]=max(x1);
[M2,I2]=max(x2);
[M3,I3]=max(x3);
Optimalq=[acbfactor(1,I0),acbfactor(1,I1),acbfactor(1,I2),acbfactor(1,I3)];
 figure;
 plotgraph=plot(number,Optimalq,"-p");
 ylim([0,0.01]);
title('Optimal q verus number ')
 xlabel('Number of MTDs')
 ylabel('Optimal q')
 legend('£f=0.006')
```

Appendix V :performance_comparison_delay.m

```
tic;
iteration=40;
acbincrease=0.0005;
acbfactor=zeros(1,iteration);
acbfactor(1,1)=0.0;
z1=zeros(1,iteration);
z2=zeros(1,iteration);
z3=zeros(1,iteration);
z4=zeros(1,iteration);
z5=zeros(1,iteration);
number=[100,150,200,250];
opt_q_delay=zeros(1,4);
fix_q_delay=zeros(1,4);

for loop = 1:iteration

    fprintf('This is 1 of  %d th function \n',loop);
    [~,~,delay1] = slottedalohanetwork(100,0.003,10^6,acbfactor(1,loop));
      z1(1,loop)=delay1;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end
for loop = 1:iteration

    fprintf('This is 2 of  %d th function \n',loop);
    [~,~,delay2] = slottedalohanetwork(150,0.003,10^6,acbfactor(1,loop));
      z2(1,loop)=delay2;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end
for loop = 1:iteration

    fprintf('This is 3 of  %d th function \n',loop);
    [~,~,delay3] = slottedalohanetwork(200,0.003,10^6,acbfactor(1,loop));
      z3(1,loop)=delay3;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end
for loop = 1:iteration

    fprintf('This is 4 of  %d th function \n',loop);
    [~,~,delay4] = slottedalohanetwork(250,0.003,10^6,acbfactor(1,loop));
      z4(1,loop)=delay4;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end

[~,I1]=min(z1);
[~,I2]=min(z2);
[~,I3]=min(z3);
[~,I4]=min(z4);
Optimalq=[acbfactor(1,I1),acbfactor(1,I2),acbfactor(1,I3),acbfactor(1,I4)];
```

```
    fprintf('This is 1 th opitmal function \n');
    [~,~,opt_delay1] = slottedalohanetwork(100,0.003,10^6,Optimalq(1));
        opt_q_delay(1,1)=opt_delay1;

    fprintf('This is 2 th optimal function \n');
    [~,~,opt_delay2] = slottedalohanetwork(150,0.003,10^6,Optimalq(2));
        opt_q_delay(1,2)=opt_delay2;

    fprintf('This is 3 th optiaml function \n');
    [~,~,opt_delay3] = slottedalohanetwork(200,0.003,10^6,Optimalq(3));
        opt_q_delay(1,3)=opt_delay3;

    fprintf('This is 4 th optiaml function \n');
    [~,~,opt_delay4] = slottedalohanetwork(250,0.003,10^6,Optimalq(4));
        opt_q_delay(1,4)=opt_delay4;



[~,~,fix_q1_delay1] = slottedalohanetwork(100,0.003,10^6,0.01);
fix_q1_delay(1,1)=fix_q1_delay1;
[~,~,fix_q1_delay2] = slottedalohanetwork(150,0.003,10^6,0.01);
fix_q1_delay(1,2)=fix_q1_delay2;
[~,~,fix_q1_delay3] = slottedalohanetwork(200,0.003,10^6,0.01);
fix_q1_delay(1,3)=fix_q1_delay3;
[~,~,fix_q1_delay4] = slottedalohanetwork(250,0.003,10^6,0.01);
fix_q1_delay(1,4)=fix_q1_delay4;

[~,~,fix_q2_delay1] = slottedalohanetwork(100,0.003,10^6,0.02);
fix_q2_delay(1,1)=fix_q2_delay1;
[~,~,fix_q2_delay2] = slottedalohanetwork(150,0.003,10^6,0.02);
fix_q2_delay(1,2)=fix_q2_delay2;
[~,~,fix_q2_delay3] = slottedalohanetwork(200,0.003,10^6,0.02);
fix_q2_delay(1,3)=fix_q2_delay3;
[~,~,fix_q2_delay4] = slottedalohanetwork(250,0.003,10^6,0.02);
fix_q2_delay(1,4)=fix_q2_delay4;

[~,~,fix_q3_delay1] = slottedalohanetwork(100,0.003,10^6,0.03);
fix_q3_delay(1,1)=fix_q3_delay1;
[~,~,fix_q3_delay2] = slottedalohanetwork(150,0.003,10^6,0.03);
fix_q3_delay(1,2)=fix_q3_delay2;
[~,~,fix_q3_delay3] = slottedalohanetwork(200,0.003,10^6,0.03);
fix_q3_delay(1,3)=fix_q3_delay3;
[~,~,fix_q3_delay4] = slottedalohanetwork(250,0.003,10^6,0.03);
fix_q3_delay(1,4)=fix_q3_delay4;

[~,~,fix_q4_delay1] = slottedalohanetwork(100,0.003,10^6,0.04);
fix_q4_delay(1,1)=fix_q4_delay1;
[~,~,fix_q4_delay2] = slottedalohanetwork(150,0.003,10^6,0.04);
fix_q4_delay(1,2)=fix_q4_delay2;
[~,~,fix_q4_delay3] = slottedalohanetwork(200,0.003,10^6,0.04);
fix_q4_delay(1,3)=fix_q4_delay3;
[~,~,fix_q4_delay4] = slottedalohanetwork(250,0.003,10^6,0.04);
fix_q4_delay(1,4)=fix_q4_delay4;
```

```
 figure;
 plotgraph=plot(number,opt_q_delay,"-p",number,fix_q1_delay,"-
o",number,fix_q2_delay,"-x",number,fix_q3_delay,"-d",number,fix_q4_delay,"-
s");
 ylim([0,2000]);
title('Delay verus number ')
 xlabel('Number of MTDs')
 ylabel('Delay')
 lgd=legend('Optimal factor q','Fixed factor q=0.01','Fixed factor
q=0.02','Fixed factor q=0.03','Fixed factor
q=0.04','Location','northwest');
 lgd.FontSize=14;




 toc;
```

Appendix VI :performance_comparison_throughput.m

```
tic;
% 0.02  0.1
iteration=40;
number0=50;
number1=100;
number2=150;
number3=200;
number4=250;
lambda=0.005;
lambda0=0.00;
lambda1=0.003;
lambda2=0.004;
lambda3=0.005;
lambda4=0.006;
lambda5=0.007;
lambda6=0.008;
acbincrease=0.0025;
acbfactor=zeros(1,iteration);
acbfactor(1,1)=0.0;
y0=zeros(1,iteration);
z0=zeros(1,iteration);
y1=zeros(1,iteration);
z1=zeros(1,iteration);
y2=zeros(1,iteration);
z2=zeros(1,iteration);
y3=zeros(1,iteration);
z3=zeros(1,iteration);
y4=zeros(1,iteration);
z4=zeros(1,iteration);
y5=zeros(1,iteration);
z5=zeros(1,iteration);
x0=zeros(1,iteration);
number=[100,150,200,250];
opt=zeros(1,4);
fixt=zeros(1,4);

for loop = 1:iteration

    fprintf('This is 1 of  %d th function \n',loop);
    [~,throughput1,~] =
slottedalohanetwork(100,0.003,10^6,acbfactor(1,loop));
      y1(1,loop)=throughput1;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end
for loop = 1:iteration

    fprintf('This is 2 of  %d th function \n',loop);
    [~,throughput2,~] =
slottedalohanetwork(150,0.003,10^6,acbfactor(1,loop));
     y2(1,loop)=throughput2;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end
for loop = 1:iteration
```

```
    fprintf('This is 3 of %d th function \n',loop);
    [~,throughput3,~] =
slottedalohanetwork(200,0.003,10^6,acbfactor(1,loop));
        y3(1,loop)=throughput3;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end
for loop = 1:iteration

    fprintf('This is 4 of %d th function \n',loop);
    [~,throughput4,~] =
slottedalohanetwork(250,0.003,10^6,acbfactor(1,loop));
        y4(1,loop)=throughput4;
    if loop<=iteration-1
        acbfactor (1,loop+1)=acbfactor(1,loop)+acbincrease;
    end
end

[~,I1]=max(y1);
[~,I2]=max(y2);
[~,I3]=max(y3);
[~,I4]=max(y4);
Optimalq1=[acbfactor(1,I1),acbfactor(1,I2),acbfactor(1,I3),acbfactor(1,I4)]
;



    fprintf('This is 1 of %d th opitmal function \n',loop);
    [~,opthroughput1,~] = slottedalohanetwork(100,0.003,10^6,Optimalq1(1));
        opt(1,1)=opthroughput1;

    fprintf('This is 2 of %d th optimal function \n',loop);
    [~,opthroughput2,~] = slottedalohanetwork(150,0.003,10^6,Optimalq1(2));
        opt(1,2)=opthroughput2;

    fprintf('This is 3 of %d th optiaml function \n',loop);
    [~,opthroughput3,~] = slottedalohanetwork(200,0.003,10^6,Optimalq1(3));
        opt(1,3)=opthroughput3;

    fprintf('This is 4 of %d th optiaml function \n',loop);
    [~,opthroughput4,~] = slottedalohanetwork(250,0.003,10^6,Optimalq1(4));
        opt(1,4)=opthroughput4;



[~,fixq1throughput1,~] = slottedalohanetwork(100,0.003,10^6,0.05);
fixt1(1,1)=fixq1throughput1;
[~,fixq1throughput2,~] = slottedalohanetwork(150,0.003,10^6,0.05);
fixt1(1,2)=fixq1throughput2;
[~,fixq1throughput3,~] = slottedalohanetwork(200,0.003,10^6,0.05);
fixt1(1,3)=fixq1throughput3;
[~,fixq1throughput4,~] = slottedalohanetwork(250,0.003,10^6,0.05);
fixt1(1,4)=fixq1throughput4;
```

```
[~,fixq2throughput1,~] = slottedalohanetwork(100,0.003,10^6,0.075);
fixt2(1,1)=fixq2throughput1;
[~,fixq2throughput2,~] = slottedalohanetwork(150,0.003,10^6,0.075);
fixt2(1,2)=fixq2throughput2;
[~,fixq2throughput3,~] = slottedalohanetwork(200,0.003,10^6,0.075);
fixt2(1,3)=fixq2throughput3;
[~,fixq2throughput4,~] = slottedalohanetwork(250,0.003,10^6,0.075);
fixt2(1,4)=fixq2throughput4;

[~,fixq3throughput1,~] = slottedalohanetwork(100,0.003,10^6,0.1);
fixt3(1,1)=fixq3throughput1;
[~,fixq3throughput2,~] = slottedalohanetwork(150,0.003,10^6,0.1);
fixt3(1,2)=fixq3throughput2;
[~,fixq3throughput3,~] = slottedalohanetwork(200,0.003,10^6,0.1);
fixt3(1,3)=fixq3throughput3;
[~,fixq3throughput4,~] = slottedalohanetwork(250,0.003,10^6,0.1);
fixt3(1,4)=fixq3throughput4;
%
 figure;
 plotgraph=plot(number,opt,"-p",number,fixt1,"-o",number,fixt1,"-
x",number,fixt2,"-d",number,fixt3,"-s");
 ylim([0,0.5]);
title('Throughput verus number ')
 xlabel('Number of MTDs')
 ylabel('Throughput')
 lgd=legend('Optimal factor q','fixed q=0.05','fixed q=0.025','fixed
q=0.01');
 lgd.FontSize=14;



 toc;
```

Reference:

Wen Zhan, Lin Dai, Massive Random Access of Machine-to-Machine Communications in LTE Networks: Modeling and Throughput Optimization, 2018.

Wen Zhan, Lin Dai, Throughput optimization for massive random access of M2M communications in LTE networks, 2017

Gebreselassie Haile, Jaesung Lim, C-OFDMA: Improved Throughput for Next Generation WLAN Systems Based on OFDMA and CSMA/CA, 2013.

Fuchun Joseph Lin, Bo-Yan Chen, Bo-Ting Lin, Wan-Hsun Hu, Charging architecture for M2M communications, 2017

Faezeh Morvari, Abdorasoul Ghasemi, Two-Stage Resource Allocation for Random Access M2M Communications in LTE Network, 2016

Mukesh Taneja, Lightweight protocols for LTE M2M networks, 2015

Fayezeh Ghavimi, Hsiao-Hwa Chen, M2M Communications in 3GPP LTE/LTE-A Networks: Architectures, Service Requirements, Challenges, and Applications, 2014