Konstantinos BESSAS

Diploma Thesis

A study of Q-CSMA

DEPARTMENT OF COMPUTER & COMMUNICATION ENGINEERING
\ University of Thessaly, Greece

A study of Q-CSMA

KONSTANTINOS BESSAS
Department of Computer & Communication Engineering
University of Thessaly, Greece

kobessas@gmail.com

August 11, 2011

Acknowledgement

I would like to express the deepest appreciation to my committee chair, Prof. Leandros Tassiulas, for his invaluable help and the trust he showed to me prior to completing this dissertation.

I would like to thank my committee member, Dr. Dimitrios Katsaros, whose help throughout my studies was of crucial importance for what I have become.

Furthermore, I would like to thank Dr. Georgios Paschos, for his help and guidance throughout this dissertation. He was the one to introduce me to the topic of stochastic processes and the reason I chose this particular dissertation. Without his help this would have never been completed.

Last but not least, I am especially grateful to my family and my friends for their love and patience through difficult times and especially to Roubini Pandi, for the support she provided, the creation of the cover and the help with the visual enhancement of this thesis.

Contents

Abstract

No doubt wireless network is an amazing technology which has totally changed the means of communication. Nowadays, there is no business, industry or project that can be progressed without the needs of wireless networks. Advances in this area of network technology is an everyday task for researchers around the globe.

One of the vital aspects in wireless communications is link scheduling. There are several multiple access protocols that may be used in packet radio wireless networks, all of which are used in different situations. These protocols have been tweaked over time so as to provide the best metrics for their respective usage. But the utmost point hasn't been reached yet.

In this thesis we study a medium access control protocol that was proposed in 2009 by certain individuals. It extends currently used CSMA/CA with certain ideas, that help attain maximum throughput in wireless networks. We have analyzed and tested the algorithms proposed in different network scenarios. The simulator, written in Matlab, helped us get important results for the delay and throughput metrics of the proposed algorithms. In several occasions there are comparisons of the proposed algorithms with CSMA and ALOHA, to show how much improvement has been done.

Finally, we simulated some realistic cases with a version of the proposed algorithm that can, in a sense, replace the existing CSMA/CA protocol in existing environments. Results are highly encouraging for more work to be done in implementing the suggested protocols in devices.

Περίληψη

Τα ασύρματα δίκτυα είναι, αναμφίβολα, μια καταπληκτική τεχνολογία που έχει αλλάξει σε μεγάλο βαθμό τα μέσα επικοινωνίας. Σήμερα, δεν υπάρχει καμία επιχείρηση, βιομηχανία ή έργο που να μπορεί να προχωρήσει χωρίς τη χρήση ασύρματων δικτύων. Η πρόοδος σε αυτόν τον τομέα της τεχνολογίας των δικτύων απασχολεί καθημερινά μεγάλο αριθμό ερευνητών σε όλο τον κόσμο.

Ανάμεσα cms σημαντικές δυνατότητες που αφορούν cms ασύρματες επικοινωνίες είναι και η επιλογή ενός συνόλου συνδέσεων που να μπορούν να είναι ταυτόχρονα ενεργές. Υπάρχουν διάφορα πρωτόκολλα πολλαπλής πρόσβασης που μπορούν να χρησιμοποιηθούν σε ασύρματα δίκτυα, τα οποία επιλέγονται ανάλογα με τις εκάστοτε συνθήκες. Τα πρωτόκολλα αυτά έχουν βελτιωθεί με την πάροδο του χρόνου, αλλά υπάρχει ακόμα περιθώριο για περαιτέρω βελτίωση.

Στην εργασία αυτή μελετούμε ένα MAC πρωτόκολλο που είχε προταθεί το 2009 από μια ερευνητική ομάδα. Επεκτείνει το CSMA/CA, που χρησιμοποιείται ευρέως σήμερα, με ορισμένες ιδέες, που συμβάλλουν στην επίτευξη μέγιστης απόδοσης. Αναλύσαμε και δοκιμάσαμε τους αλγορίθμους σε διάφορα δίκτυα και σενάρια. Η προσομοίωση έγινε με χρήση του Matlab, και οδήγησε σε σημαντικά αποτελέσματα. Σε αρκετές περιπτώσεις υπάρχουν συγκρίσεις των προτεινόμενων αλγορίθμων με το CSMA και το ALOHA, προκειμένου να φανεί η μεγάλη βελτίωση που έχει γίνει. Τέλος, προσομοιώσαμε ορισμένες ρεαλιστικές συνθήκες με μια έκδοση του προτεινόμενου αλγορίθμου που μπορεί, κατά μία έννοια, να αντικαταστήσει το υπάρχων CSMA/CA πρωτόκολλο. Τα αποτελέσματα είναι ιδιαίτερα ενθαρρυντικά για περαιτέρω εργασία.

*Ελληνικός τίτλος εργασίας 'Ανάλυση επίδοσης του κατανεμημένου πρωτοκόλλου QCSMA και επέκταση σε εφαρμογές Wi-Fi'

1 Introduction

One of the challenges in multi hop wireless networks is to maximize *throughput* over a communication channel. The greatest challenge is to attain optimal throughput [1],in a distributed manner. Recently, it has been shown that Carrier Sense Multiple Access (CSMA)-type random access algorithms can achieve the maximum possible throughput in ad hoc wireless networks. Work has been done in this area of studies since the late
70s.

In general, single channel communication network is considered. In this kind of network, if more than one links are active in a certain neighborhood or more than one entities try to access a certain link, we say we have a *collision.* When a collision occurs, packets are not successfully transmitted over the medium.

To better explain what we are looking into here, let us consider a wireless network consisting of two nodes as shown in Figure 1.

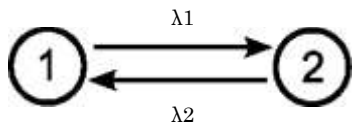

Figure 1: Two node network with two flows

There are two flows, from node 1 to node 2 and vice versa. Let's consider that the arrival rate in node 1 is $\boldsymbol{\lambda_1}$ and λ_2 in node 2. The throughput region of this system, when the channels permit a maximum of 1 unit per slot for one of the two flows, only by using simple time sharing has been proven to be the area shown in Figure 2. Any arrival vector outside this region will lead the system to instability, irrespectively of the protocol used.

Using centralized polling protocols in a network, this region can be achieved. However, when operating in a distributed environment, *valid schedules* must be chosen independently, using data and services available for that cause. *Scheduling* is the process of deciding which links can be active in a network at a certain time. When transmitting using a valid schedule, there will be no collisions whatsoever.

The first attempt to use a distributed MAC protocol for packet networks was ALOHA[1]. As defined in [6], pure ALOHA is a random access protocol, where a user accesses a channel as soon as a message is ready to be transmitted. After the transmission, the user waits for an acknowledgment on either the same or a separate feedback channel. Collided packets are retransmitted after waiting for a random period of time. The vulnerability period of this approach is double the packet duration, since a collision can occur at any time during the packet transmission. Slotted ALOHA was introduced afterwards to reduce the vulnerability period to only one packet duration, as there can not exist partial packet collision. Yet, slotted ALOHA systems suffer greatly from collisions as well, which leads to a loss in throughput as shown in Figure 3. To find the throughput region of ALOHA, different back-off windows were used. Larger back-off windows than those can not improve the throughput region, for this network.

---

[1]We say that a scheduling algorithm (explained in Chapter 3) is *throughput optimal,* if it can keep the network stable for all arrival rates in the throughput region.

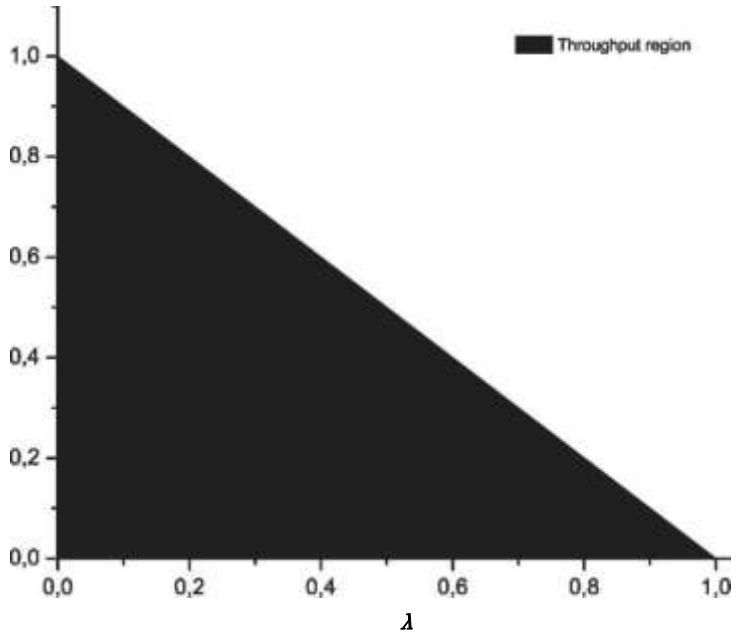

Figure 2: Throughput region without network coding

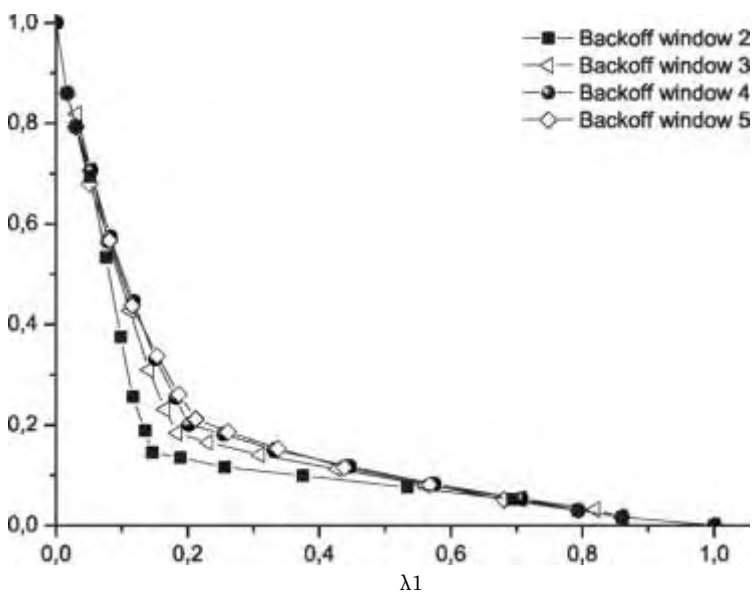

Figure 3: Throughput region of Slotted ALOHA

The second step was to alleviate collisions using Carrier Sense Multiple Access (CSMA). Over the time, different approaches of CSMA have been introduced. The 802.11 standard used in wireless networks worldwide uses a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) MAC based protocol called *Distributed Coordination Function.* Comparing CSMA to ALOHA, collisions are not fully elimi-

nated which causes throughput inefficiency as shown in Figure 4. There you can see the throughput attained for different back-off windows choices. Aggregate throughput gets vastly lower as the number of colliding nodes increases. Throughput can be optimized by adjusting the back-off window. In Figure 5 you can see how the back-off time affects the aggregate throughput attained for the network in Figure 1.

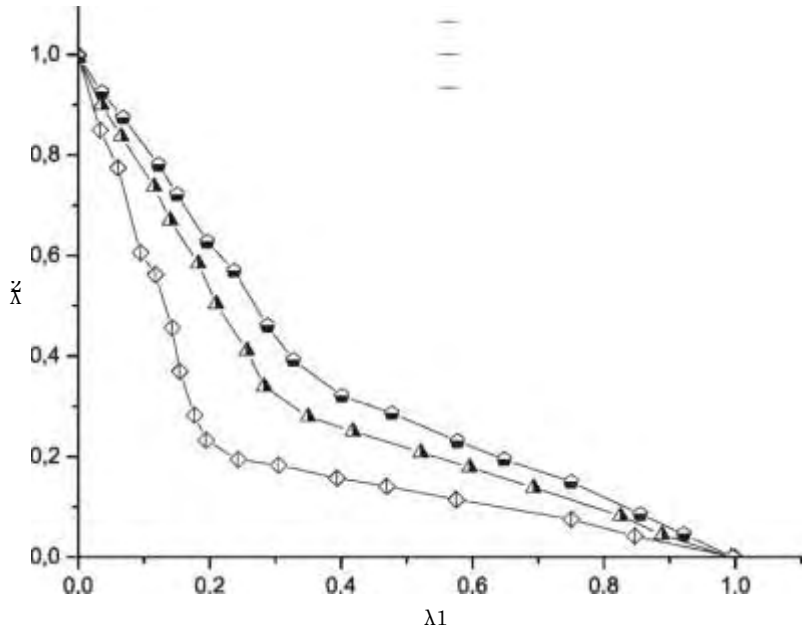

Figure 4: Throughput region of CSMA

On the other hand, the well-known max-weight and back-pressure scheduling algorithms introduced in [3] and [4] are throughput-optimal. However, they are centralized algorithms and have high computational complexity.

This thesis studies a particular algorithm proposed in December 2009, "Q-CSMA: Queue-Length Based CSMA/CA Algorithms for Achieving Maximum Throughput and Low Delay in Wireless Networks" by Jian Ni, Bo Tan and R. Srikant[2], that advertises throughput optimality. You can see a first comparison of aggregate throughput of Q-
CSMA with CSMA in Figure 5.

The rest of this thesis is organized as follows. Chapter 2 describes different graph models for networks that will be used in later chapters for experiments. Chapter 3 describes the Q-CSMA algorithm, the benefits that it introduces and any potential drawbacks that might be encountered. Chapter 4 describes the Hybrid Q-CSMA algorithm, which is a combination of a distributed version of Greedy Maximal Scheduling and Q-CSMA. Chapter 5 contains all the experiments that were carried out to calculate the metrics of the studied algorithms. Finally in Chapter 6 we describe node-based Q-CSMAandtestitinanexistingWi-Finetwork.

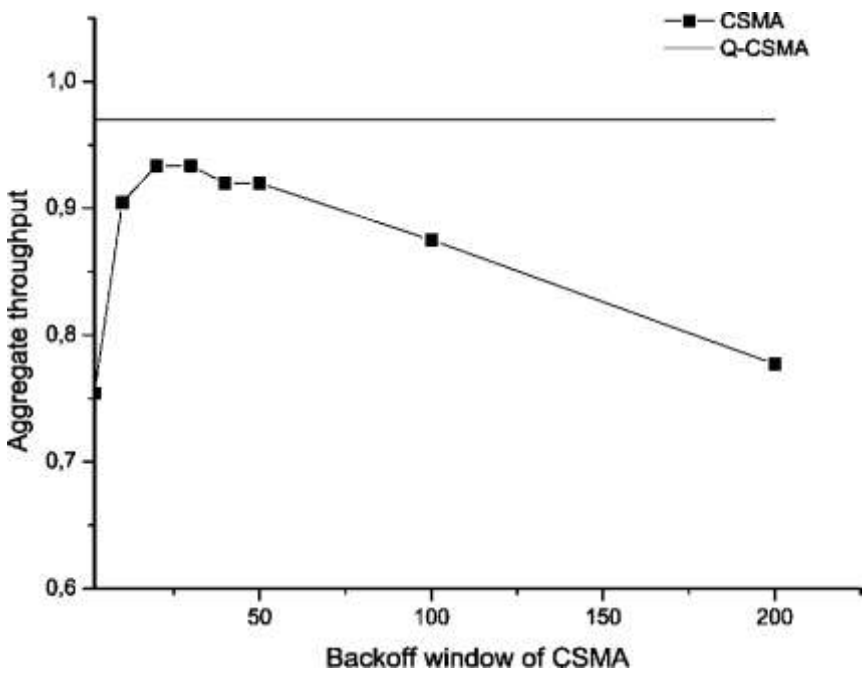

Figure 5: How the back-off window affects the throughput of CSMA

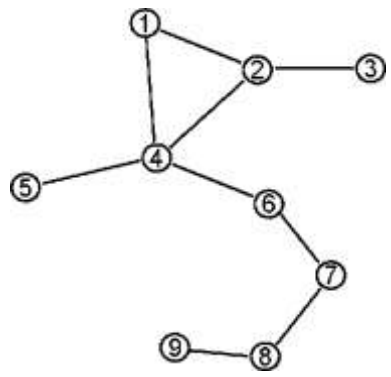

Figure 6: Example of a Network

2 Graphs

In order to get realistic simulation results and obtain insight to the problem, we have tested the algorithms in various network topologies. In this Section we are going to examine all the topologies used to test the algorithms, and study all the important principles that defines each of them.

2.1 Definitions

To begin with, some definitions have to be given. We model a single-channel wireless network by a graph $G=(V, E), where\ V$ is the set of nodes and E is the set of links. Nodes are wireless transmitters and/or receivers. There exists a directional link $(n,m) e E$ if node m can hear the transmission of node n. When two nodes can hear each other, we can say that there exists a unidirectional link n,m if and only if there exists two directional links $(n,m) e E$ and $(m,n) e E$ (ex. Figure 6).

G is also called the connectivity graph and is represented by the a $n \chi n$ matrix A, where n is the number of nodes the network consists of. When a link exists between anode$i$ andanode$j$, the respective (i,j) (and (j,i) for unidirectional links) position in the matrix is set to 1. All other are set to 0. The connectivity graph matrix of the network of Figure 6 is given in (1).

$$\begin{matrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix} \quad (1)$$

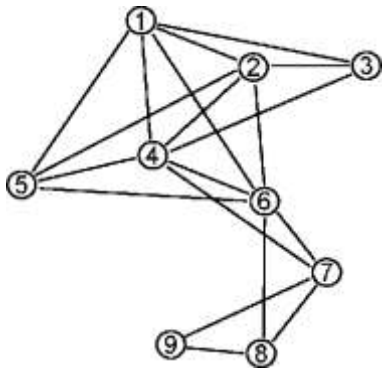

Figure 7: 2-hop collision graph of the network in Figure 6 2.2

Interference Model

Another notion of interest is the *conflict graph, o r* otherwise known as *collision graph.*

In any given network, a node can hear not only the nodes that it is connected to, but other nodes in close range. This can be modeled by using the *k-hop* collision rule. A *collision set C(i)* is the set of nodes that collide with node i.The *k-hop collision graph* is a graph *C* where a link exists if and only if between nodes *i* and j,there is a path of length smaller than k in the original graph G. The following is true[2]:

$$e \; C(i) \text{ and } \qquad e \; C(i) \; V \; i, j.$$

The actual collision graph is $G_c = (V, C(i))$. The 1-hop collision graph of any network graph is the network graph itself. For instance, the 1 hop collision graph of the graph in Figure 6 is the same graph. The 2-hop collision graph of the same network is shown in Figure 7 and the collision graph matrix in (2).

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ & & & & & & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & \\ 0 & & & & & & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ & & & & & & & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ & & & & & & & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & \end{pmatrix} \tag{2}$$

Finally the 3-hop collision graph can be seen in Figure 8 and the respective collision graph matrix in (3).

If we consider static signal strengths from each node.

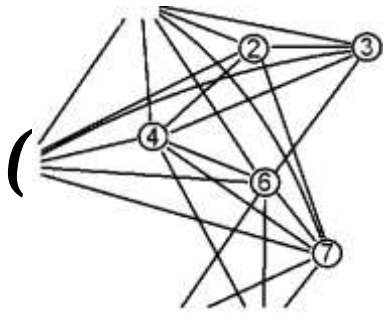

Figure 8: 3-hop collision graph of the network in Figure 6

$$\begin{matrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{matrix} \quad (3)$$

The collision graph is a simple way to decide the scheduled transmissions in contention-based channel access protocols. Simple as it may be, it is less accurate than more advanced, but complex, models that have been introduced. Collision graphs imply that the nodes are so close to each other that they experience similar noise and interference conditions. A receivers ability to decode a packet successfully depends on channel conditions near the *receiver* and not the *sender*. Several issues that can appear include a transmitter not transmitting even if the receiver is in a low interference neighborhood[3] or a transmitter without any or with low interference transmitting to a receiver in high interference neighborhood[4] (ex. Figure 9).

In this thesis, the collision set is used for the links to make contentions and decide their schedules. In general, we use the 1-hop collision rule, and in some cases the 2-hop collision rule. Later on you can see the collision sets for interesting topologies that are were used for the simulation of all algorithms tested.

2.3 Topologies of interest

There are several widely used graphs when it comes to wireless network simulations. *Linear Networks, Grids, Clique* and *Random Graphs* are some topologies that are widely used for this cause.

---

[3] *Exposed terminal* situation
[4] *Hidden terminal* situation

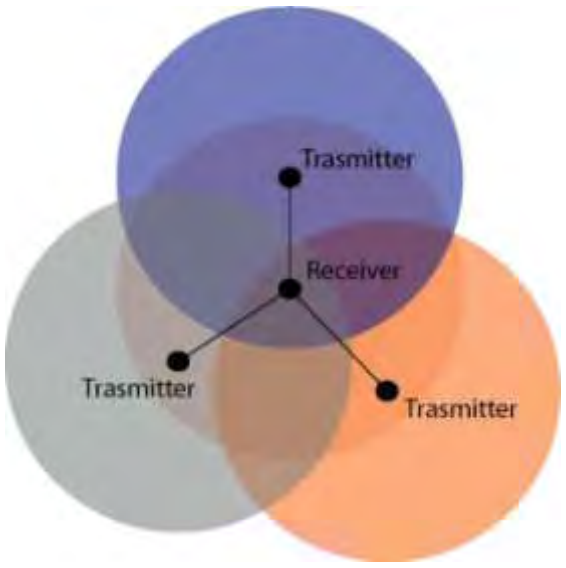

Figure 9: Hidden terminal situation

Figure 10: A 6-node Linear Graph 2.3.1

Linear Network

A *Linear Network* is a is a particularly simple example of a tree, namely a tree with two or more vertices that is not branched at all. It only contains only vertices of degree 2 and 1. Every node is connected to two other nodes with the exception of edge nodes that are connected to only one node (ex. Figure 10). Linear Networks are interesting to study from the perspective of how easy it is to monitor and understand the system behavior. It should be obvious to the reader that in a linear wireless network with 1-hop interference assumption, when a node is sending data then both neighbors should be idle. To paraphrase, if a link is active, then the next and the previous link in line should be inactive to avoid collisions in transmissions.

Let's assume a network flow from node 1 to node 6. Arrivals happen from an external source in node 1 and data leave the network from node '6'. The system is monitored in time slots. Using the counting of nodes from Figure 10, the optimum schedule for each time slot, if we consider equal capacity for every link, would be to just the odd or just the even nodes transmitting. The two different scenarios can be observed in Figure 11 and Figure 12.

For arrivals in node 1 that the system can keep the queues stable, hence 0.5 arrivals in every time slot (1 every 2 slots), the scheduler should choose either the first or the second schedule with the same probability. The maximum average throughput of this network is 0.5 packets in every time slot for the same arrival rate. For average arrivals of less than 0.5 packets/time slot the departure will be the same as the arrival rate. Finally when the arrival rate is higher than 0.5 packets/time slot the system will be unstable, under any valid scheduling decision.

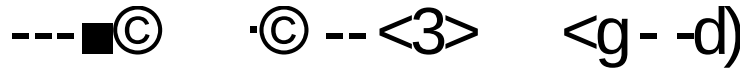

Figure 11: Odd nodes transmitting

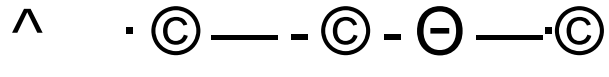

Figure 12: Even nodes transmitting

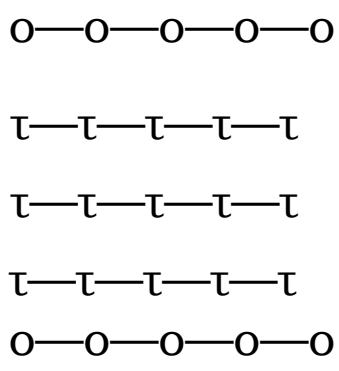

Figure 13: Example of 5 χ 5 Grid Network

2.3.2 Grid

A two-dimensional *Grid* is an $m \chi n$ graph G_{mn} that is the graph Cartesian product $P_m DP_n$ of linear graphs on m and n vertices. A 5 χ 5 2D grid can be seen in Figure 13.

Using the 1-hop collision graph to model the network, when a node n is transmitting, this means that every neighbor of n must remain silent (see Figure 14 where a node is marked with 'T' if it can transmit). For the k-hop model, every k_th neighbor of n must remain silent while n is transmitting.

When working with the various topologies to simulate algorithms and measure their effectiveness, it is mandatory to define network flows for the data packets. Let's consider 10 data flows for the Grid network of Figure 13 as shown in Figure 15. This is an example we will simulate later on.

2.3.3 Clique

A *Clique* is a network where all possible links exist. In a clique network all links are connected to each other (ex. Figure 16).

Wecanalsoconsidera*Clique* in an undirected graph $G=(V, E)$as a subset of the vertex set C C V, such that for every two vertices in C, there exists an edge connecting the two. In a network paradigm, if two nodes can hear each other then there will be formed a link between them. For instance the graph of Figure 17 has one 4-vertex maximum clique, consisting of 1,2,3,4, one 3-vertex maximal clique consisting of 5,6,7 and two 2-vertex cliques consisting of 4,7 and 7,8 respectively. We assume that there is

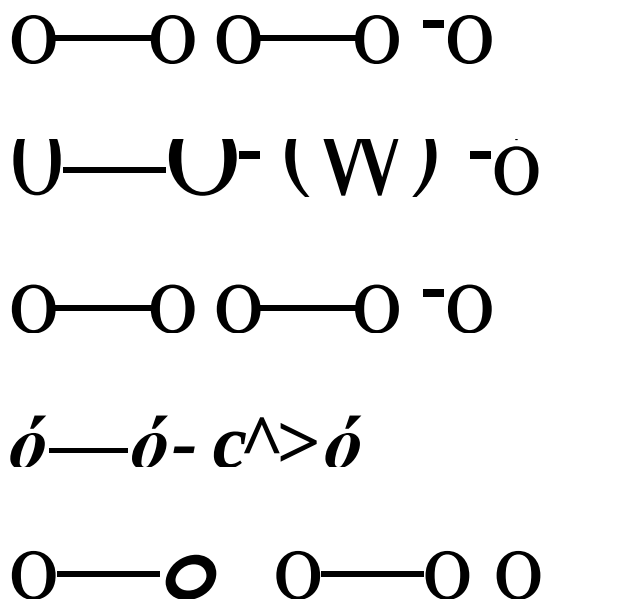

Figure 14: Valid schedules for a 1-hop interference 5 γ 5Gridnetwork

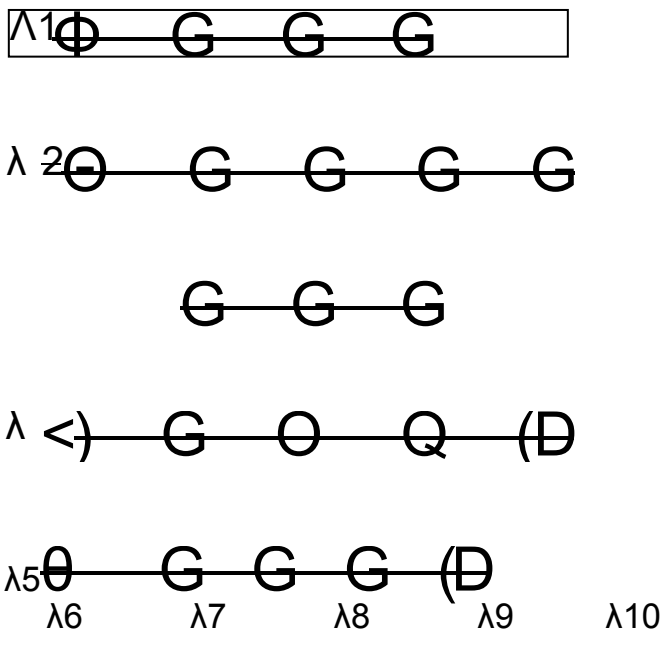

Figure 15: Example of 5 γ 5 Grid network with 10 flows

no communication or interference between any two nodes if there is no link connecting the two.

A *time expansion* graph is a graph the evolves during time. The general idea is that nodes are moving in and out of the graph, and as they move links are created and keep on breaking. The graph, as the time passes, creates cliques by adding new edges among the moving nodes and the one that are static in the network core. Imagine for example the graph of Figure 17, and let's add a moving node as shown in Figure 18.

2.3 Topologies of interest

Figure 16: Clique Network of 6 nodes

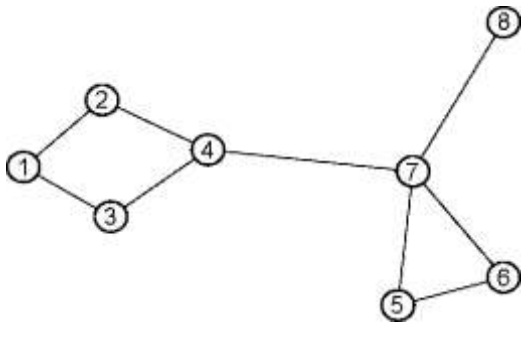

Figure 17: Example of graph with 4 cliques

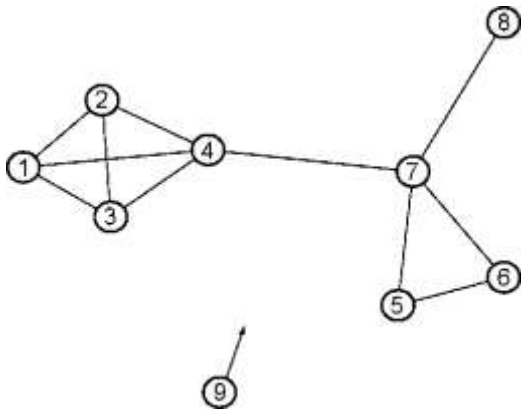

Figure 18: Example of graph with 4 cliques, while a node starts entering the network

As the node moves, links are created and destroyed while the node tries to be connected with as many nodes it can at a certain time. You can see the time expansion

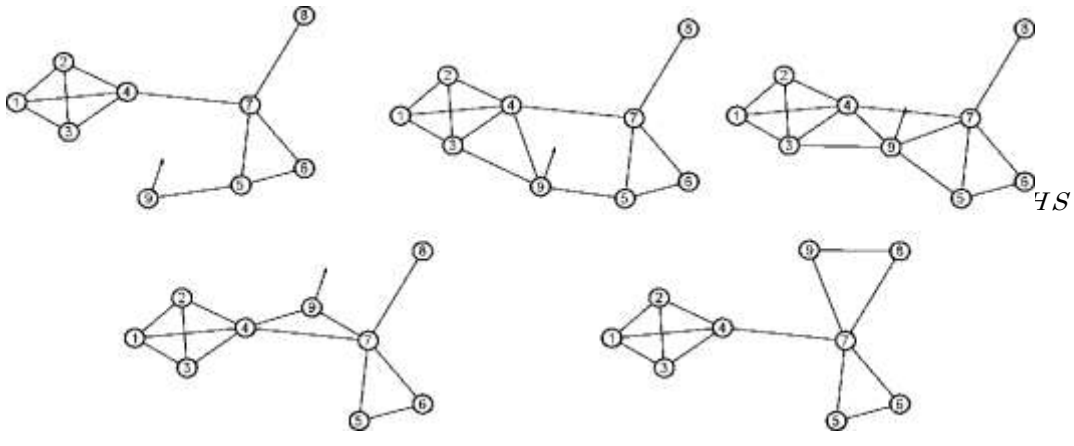

Figure 19: Time expansion as node 9 moves along the network

of this network, as the time lapses until node 9 stops moving in Figure 19.

2.3.4 Random graphs

Lastly, *Random graphs* are graphs without known formation beforehand. The results of simulations in random graphs are not easily comparable, but they can be used to find average metrics for complete random input. There will be some experiments in random created graphs later on.

For the creation of the graphs, a simple graph creator was used. The creation is done node by node. Each node *n* connects with at least one node *m* from the nodes already in the network, and connects to nodes in the collision set of *m* with probability *p.* In this way, probability *p* is a metric of the density of the graph that will be created.

3 Q - C S M A

3.1 Network Model

The network model, based on [2], is a single-channel wireless network. The concepts used are all those explained in the previous chapter. The goal of Q-CSMA, as stated by it's writers, is to propose a *scheduling algorithm*[5] for wireless networks that is throughput optimal. Q-CSMA uses the queue length to determine the activation probability of a link, hence the letter Q in its name.

The system is considered to be slotted, where every slot is divided into control mini-slots and a data slot. During the control mini-slots, feasible schedules[6] are selected, which are then used during data slot to transmit data packets. If a certain link is active, data slot is considered big enough so that a whole packet can be transmitted in one time slot.

The schedule in every time slot is represented in a vector x $C\{0,1\}^{|V\ 1}$ .If the i^{th} element of x equals to 1, then link i will transmit in this time slot. For a schedule to be feasible, if link i is going to be active, then no links in the collision set of link i can be active in this time slot. As so, the following must be true in every time slot:

$$Xi + xj < \qquad e\ V \text{and}\ j, i e\ C(i), C(j)$$

3.2 Filtering out the infeasible schedules

To better understand the basic idea, let's assume that the algorithm selects randomly a feasible schedule *m,* during the control slot. This is only a temporary schedule. The transmission schedule will be decided as follows. For every element of m that equals to1(m(i) == 1), if no nodes in C(i) were active in the previous slot, x(i) will be set to 1 with probability p.IfnodesinC(i) where active during the previous data slot, then node i will not transmit during the current data slot. For every other node that m(i) equals to 0, the decision for this data slot transmission schedule will be the same as for the previous data slot. Matlab code that decides feasible schedules can be seen in Listing 1 (Appendix A).

3.3 Distributed Implementation

As stated repeatedly, we are looking into a distributed algorithm that is throughput optimal. In order to implement the basic algorithm that was introduced in the previous Section, there is a need for a randomized and distributed procedure that will select a feasible schedule during the control slot.

For this to happen, the control slot is divided into control mini-slots. Each link selects a random back-off time. The number of these mini slots is random, based on the maximum value of the back-off time selected by each node. During these mini-slots nodes exchange certain messages. When the back-off time lapses, each link transmits an 'INTENT' message, stating its intention to decide whether it can transmit or not in this time slot. A node will transmit an 'INTENT' message only if it haven't heard

---

[5] A *scheduling algorithm* is an algorithm that decides which links may be activated at a certain time for data transmission.

[6] Set of links that can be active at the same time according to the collision set

Program 1 Basic Scheduling Algorithm (in Time Slot *t*)

1. **In the control slot, randomly select a decision schedule** m(t) **£** ***M*** **with probability** ***a*** (m(t)).

   Vi **£** m(t):

   If **no links in C**(i) **were active in the previous data slot**

   (a) ***Xi(t)*** **= 1 with probability** ***p***

   (b) **x**i(t) **= 0 with probability 1 -** ***p*** Else

   (c) **Xi**(t) **= 0**

   Vi **/** m(t):

   (d) **Xi**(t) **=** **Xi**(t -1)

2. **In the data slot, use x**(t) **as the transmission schedule.**

one already while waiting for the back-off time to expire. If a link successfully sends an 'INTENT' message, it will then run the basic algorithm by setting the element of the *m* vector that corresponds to itself to 1. If not, it will keep the decision from previous time slot. A rather self explanatory schematic of the above can be seen in Figure 20. The respective Matlab code for the Q-CSMA Algorithm can be seen in Listing 2 (Appendix B).

Program 2 Q-CSMA Scheduling Algorithm (at Link *i* in Time Slot *t*)

1. **Link** ***i*** **selects a random (integer) back-off time** ***Ti*** **uniformly in [0,W-1] and waits for** ***Ti*** **control mini-slots.**
2. If **link** ***i*** **hears and 'INTENT' message from a link in C**(i) **before the** (**Ti** + *1*)*th* **control mini-slot,** ***i*** **will not be included in** m(t) **and will not transmit an 'INTENT' message anymore.**

   Link ***i*** **will set** **x**i(t) = $x_i(t-1)$.
3. If **link** ***i*** **does not hear an'INTENT' message from any link in C**(i) **before the** (**Ti** + 1)**th control mini-slot, it will send an 'INTENT' message to all links in C**(i) **at the beginning of the** (**Ti** + 1)**th control mini-slot.**

   - If **there is a collision, link** ***i*** **will not be included in** m(t) **and will set** **x**i(t) = $x_i(t-1)$
   - If **there is no collision, link** ***i*** **will be included in** m(t) **and decide its state as follows:**

     if **no links in C(** ***i*** **) were active in the previous data slot**

     xi(t) **= 1 with probability** ***p***

     $x_i(t)$ **= 0 with probability 1 -** ***p*** else

     $x_i(t)$ **= 0**
4. If **xi**(t) **== 1, link** ***i*** **will transmit a packet in the data slot.**

To better understand Q-CSMA algorithm, you can see the instances of the algorithm in the example below. Let's consider a linear network with a flow as shown in Figure 21, using omni-directional antennas. The network is initialized with 20 items in each queue. At first, nodes select uniformly a back-off time between 1 and 3 for

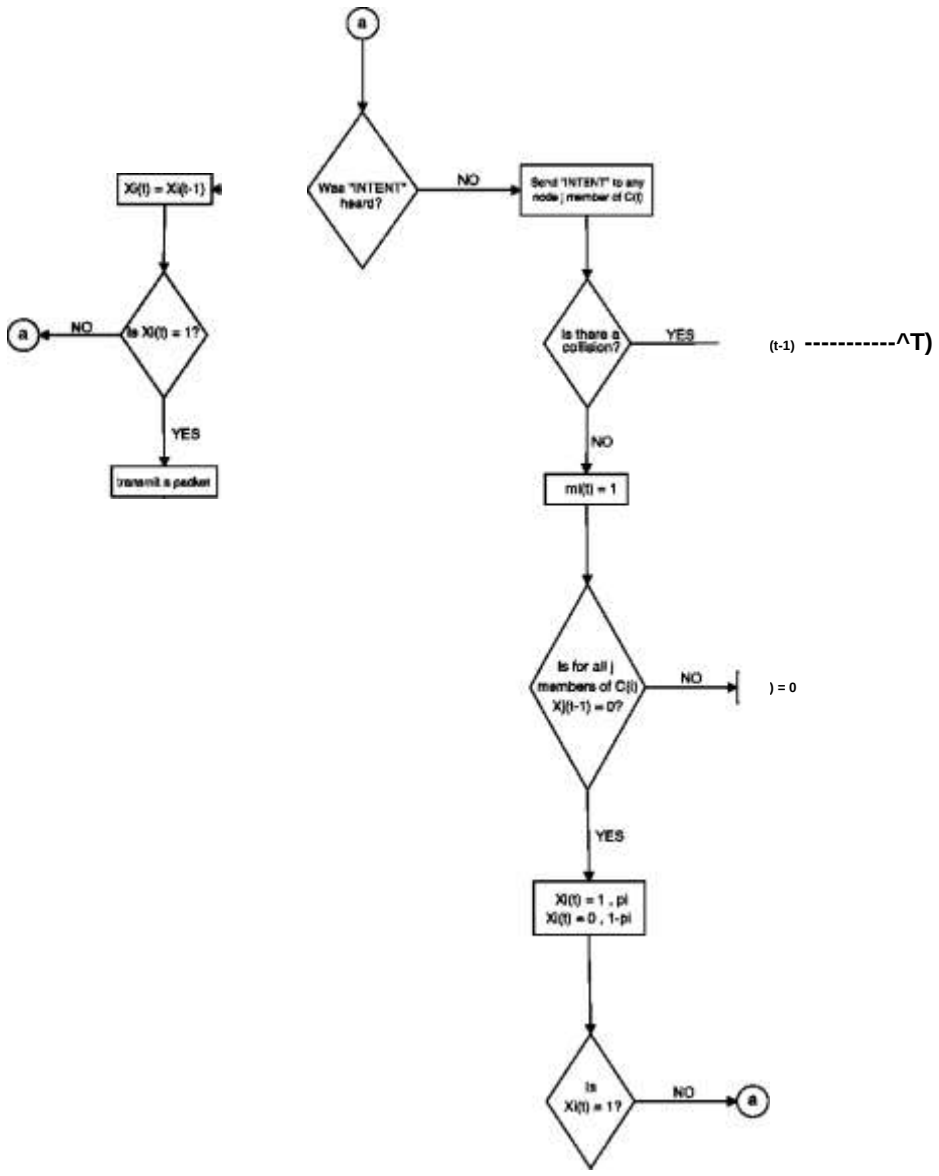

Figure 20: Schematic of Q-CSMA algorithm

their respective links. In the control mini-slot one, nodes that selected 1 will send their 'INTENT' messages. Each node that receives an 'INTENT' message is silenced, and keeps the same decision as it did in previous data slot. Nodes that successfully send an 'INTENT' message, and no collision happens upon transmission $(i.e.$ reception and transmission of 'INTENT' message in the same control mini-slot), set their decision

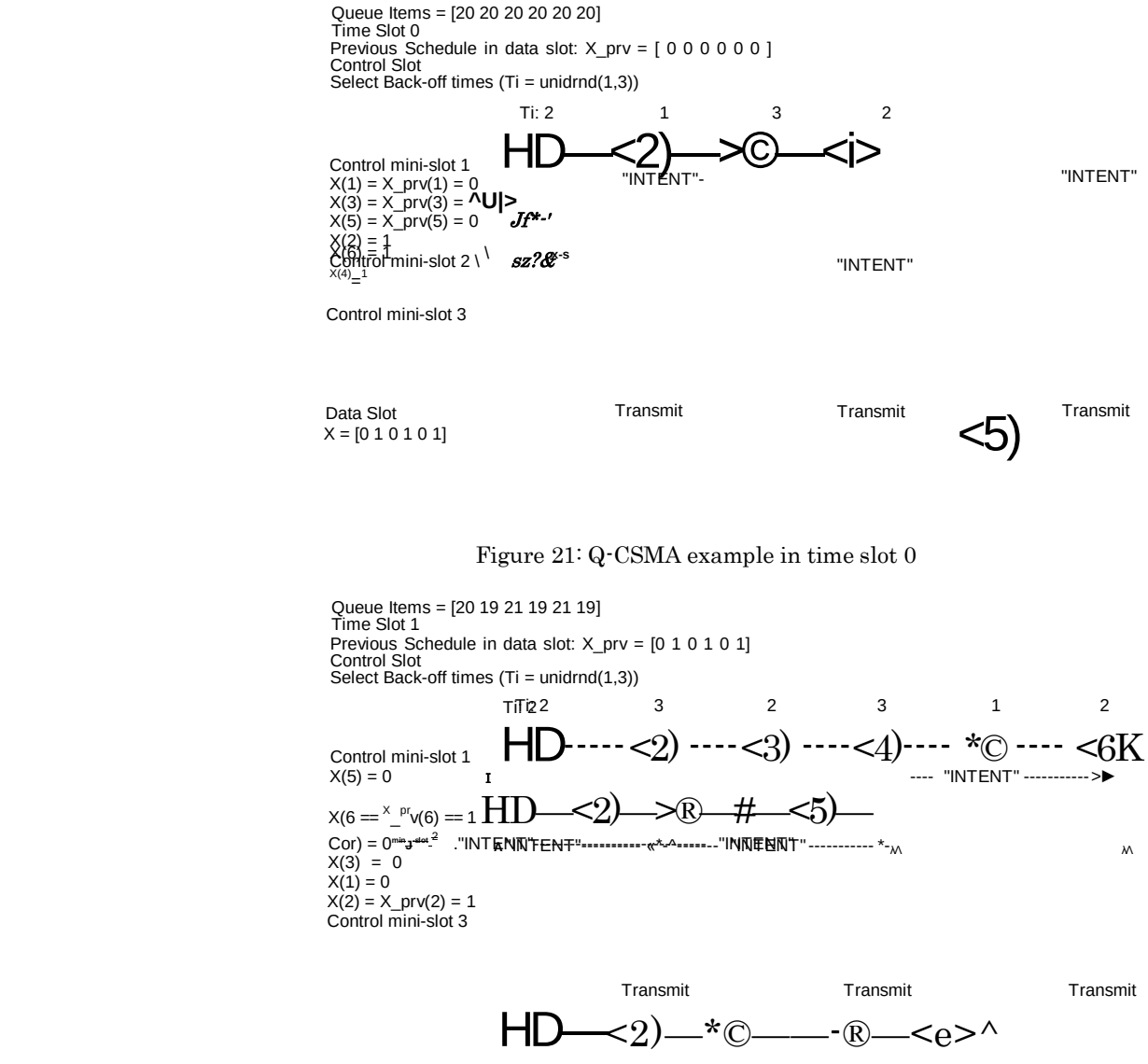

Figure 21: Q-CSMA example in time slot 0

Figure 22: Q-CSMA example in time slot 1

to 1, as in previous data slot there were no nodes active from those in their respective collision sets. Finally in the data slot, nodes that set decision to 1 transmit a packet.

In time slot 1, the queue lengths have changed as shown in Figure 22. As you can see in this time slot, although nodes 1, 3 and 5 manage to send 'INTENT' messages, they don't set their status to 1, as in the previous data slot nodes that are in their collision set where active. If we keep running this example, the states won't change unless the items in queues of nodes 2, 4 and 6 fall under 5. Then the probability p with which the node sets its status to 1, upon successful transmission of 'INTENT' message, will be low enough for the node to set its decision to 0.

Schedule m(i) decided during the exchange of 'INTENT' messages is a feasible schedule, as a node *id* sets its *idth* element to 1 only if it successfully transmits an 'INTENT' message without collision. Furthermore, nodes hearing 'INTENT' messages (C(id)) are silenced, disabling them from setting their id_{th} element to 1.

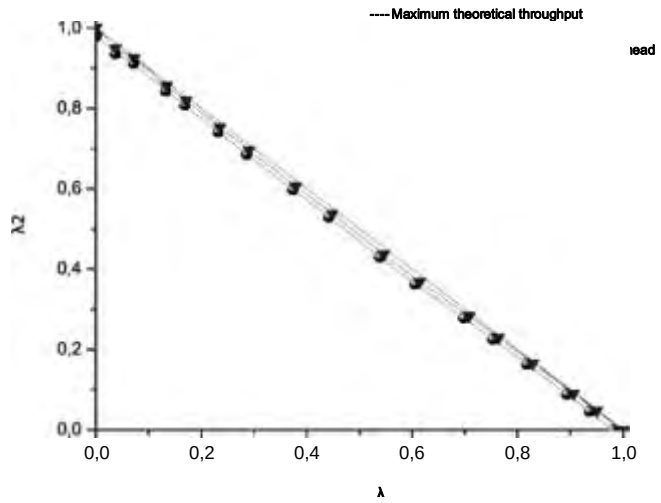

Figure 23: Throughput region of Q-CSMA

The probability *p(i),* used by a node to set its decision to 1, is a function of the queue length in node i. As you can see in (4), the probability is greater as the queue length increases and lower as it decreases[7]. This remark is what gives Q-CSMA the *throughput optimality* factor. Queues dropping under a certain number of elements are forcing their links to be enabled with low probability, giving every node a chance to transmit. There are not any queues overflowing with packets while other keep transmitting their own.

$$Pi \qquad \frac{e^{wt(t)}}{e^{wt(t)} + 1} \tag{4}$$

3.4 Experiments

Experiments run to measure the throughput region of Q-CSMA prove the throughput optimality argument. Running the same calculations as for the algorithms in Introduction, we have reached the results shown in Figure 23. To normalize the throughput we used the packet duration and slot duration from [5]. The back-off time of Q-CSMA was used as the overhead per time slot. The normalized throughput was calculated with (5).

$$normalized\ throughput = throughput * \frac{packet\ duration}{packet\ duration + average\ overhead} \tag{5}$$

[7] $w_t(t)$ is appropriate function of queue lengths, *i.e.* $w_t(t) = a *$ (queue length$_t$)

4 Hybrid Q-CSMA

4.1 General Idea

Q-CSMA was proven to be throughput optimal in [2]. Experiments have also shown this, as stated in the end of the previous chapter. However, Q-CSMA keeps the system on a rather high average number of packets waiting in queues to be transmitted. Little's law states that the average long term delay is highly affected by the average queue lengths inside the network. As it will be shown in the next chapter, Q-CSMA has a rather poor delay performance.

Driven by the cause of creating a not only throughput optimal but a low delay algorithm as well, authors of [2] introduced *Hybrid Q-CSMA,* which is a combination of *Greedy Maximal Scheduling (GMS)* and *Q-CSMA.*

4.2 D-GMS

Q-CSMA is a distributed algorithm, able of choosing feasible schedules that will keep the system stable in the long term under all arrivals in the capacity region. GMS on the other hand is a centralized greedy algorithm, with worse throughput region, but rather low delay metrics. So how these two were combined?

At first, a distributed version of GMS was introduced. This algorithm is called *D-GMS*. D-GMS is not an exact distributed version of GMS, but it rather approximately works like GMS. D-GMS, like Q-CSMA, handles contentions by assigning back-off counters to the nodes. Only here, back-off counters are chosen according to the queue lengths. The general idea is to assign smaller back-off times to links with larger queue lengths.

The control slot is divided in a fixed number of frames, with each frame being split into control mini-slots, same as Q-CSMA. These mini-slots are used to resolve contentions among links. Similarly to Q-CSMA, a link i is transmitting 'RESV' messages to all the links in C(i). Upon successful transmission without a collision, i will decide to be active in this data slot and transmit a packet.

D-GMS Algorithm can be seen below. Since the back-off counters are assigned according to queue lengths, low or empty queues will not compete for the link. The delay advantage arises from the greediness of assigning links with more packets in their queues. This way a packet never stays for too long in a queue, therefore the delay will be really low.

4.3 Hybrid Q-CMSA

The result of the mixture of Q-CSMA and D-GMS was *Hybrid Q-CSMA*.Linkswith
queue lengths greater than a threshold w_0 have their states determined by using the Q-CSMA procedure, whereas, those under that threshold will use D-GMS. This happens because of Q-CSMA's greater throughput region, so it can stabilize the network when queues go beyond a certain threshold. Moreover, for this greater amount of items in queues, it can provide the maximum throughput available. On the other hand, when the system queues are under the threshold, D-GMS will make sure that the delay will be kept in low levels.

Program 3 D-GMS Algorithm (at Link i in Time Slot t)

1. Link i selects a random back-off time

$$Ti = W * [B - \log_b(qi(t) + 1)\backslash^+ + Uniform[0, W - 1]$$

and waits for Ti controls mini-slots.

2. If link i hears a 'RESV' message from a link in $C(i)$ before the ($Ti + 1$)th control mini-slot, it will not be included in x(t) and will not transmit a 'RESV' message anymore. Link i will set $xi(t) = 0$.

3. If link i does not hear a 'RESV' message from any link in $C(i)$ before the ($Ti + 1$)th control mini-slot, it will send a 'RECV' message to all links in $C(i)$ at the beginning of $(T_i + 1)_{th}$ control mini-slot.

- **If there is a collision, link i will set $xi(t) = 0$**
- **If there is no collision, link i will set $xi(t) = 1$**

4. If $xi(t) == 1$, link i will transmit a packet in the data slot.

Hybrid Q-CSMA priorise queues with more packets. Since Q-CSMA procedure is run for links with queues above the threshold, the algorithm makes sure that all nodes above that certain threshold will run their Q-CSMA decision procedure before any node runs D-GMS. For this to happen there are W control mini-slots, just like the original Q-CSMA, used to solve contentions. After these W control mini-slots, there are $W * B$ more for nodes running D-GMS.

At the end of Q-CSMA, there is a mini-slot during which, links that have already been scheduled using Q-CSMA will let their collision set neighbors know. This mini-slot is both the end of Q-CSMA and the start of D-GMS. At the end of this mini-slot every link will have updated their NAi, which is a remember bit that is used in the next round to solve contentions during Q-CSMA. It serves the same purpose as the history knowledge of $C(i)$'s decisions in the previous time slot in simple Q-CSMA.

4.4 Single-case error

During simulations, chosen transmission schedules for the respective data slots were been checked for consistence, according to the collision set that was given as an input. In one particular scenario, Hybrid Q-CSMA produced invalid schedules, that were few but nevertheless existent.

The problem, explained graphically in Figure 24, appears when a node switches from D-GMS to Q-CSMA and one particular neighbor runs Q-CSMA. Suppose we have two nodes. Suppose queue threshold is 50 packets in queue. In time slot t node 1 one runs Q-CSMA with 80 packets in queue and node 2 runs D-GMS with 50 packets in queue. We assume the transmission schedule to be X(t) = [0 1]. For this transmission to be created, node 1 hears no 'RESV' message during $(Wo + 1)_{th}$ control mini-slot and sets $NAi = 0$ (line 1.4 of Program 4). On the other hand, for D-GMS to select 1, NA_2 can't be anything else but 0 (line 2.1 of Program 4). So NA to be used in the next round is NA = [0 0]. We will here consider stochastic arrivals in queues, which means that in the next time slot items in queue can be anything above the previous value. For

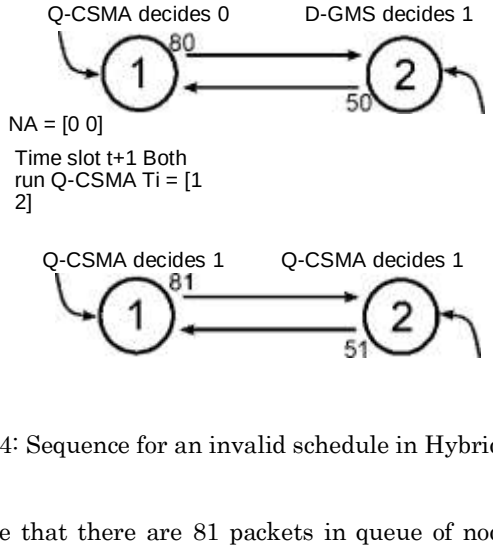

Figure 24: Sequence for an invalid schedule in Hybrid Q-CSMA

time slot (t+1) we assume that there are 81 packets in queue of node 1 and 51 packets in the respective queue of node 2. This means that both nodes will run Q-CSMA. If node 1 chooses a smaller back-off time from node 2, it will transmit an 'INTENT' message. Upon arrival node 2 will set $X_2(t+1) = X_2(t)$, hence $X_2(t+1) = 1$. But node 1 can also set it's decision to 1 with probability p ($X_i(t+1) = 1$ with p)since it successfully transmitted an INTENT message. This means that the transmission schedule in $(t+1)$ time slot is $X(t+1) = [1\ 1]$, which is not valid.

The previous problem can be solved easily by adding a check in the Q-CSMA procedure. If a link checks what procedure it was running in the previous time slot, in occasions like the one above, then there are no invalid schedules. So if a node running Q-CSMA is about to use its previous state as it s current state, because of an INTENT message was heard, it checks if in the previous slot it was running D-GMS. Then it sets it s decision to 0. Matlab code with this add-on for the Hybrid Q-CSMA can be seen in Listings 3, 4 and 5 (Appendix C).

4.5 Experiments

Experiments run to measure the throughput region of Hybrid Q-CSMA support the throughput optimality argument. Running the same calculations as for the algorithms in Introduction, we have reached the results shown in Figure 25. To normalize the throughput region, as we did in Q-CSMA, the 802.11 packet duration and slot time was used. We measured the overhead per time slot as the max back-off time of Q-CSMA or D-GMS procedure. The normalized throughput was calculated with (6).

$$normalized\ throughput = throughput * \frac{packet\ duration}{packet\ duration + average\ overhead} \quad (6)$$

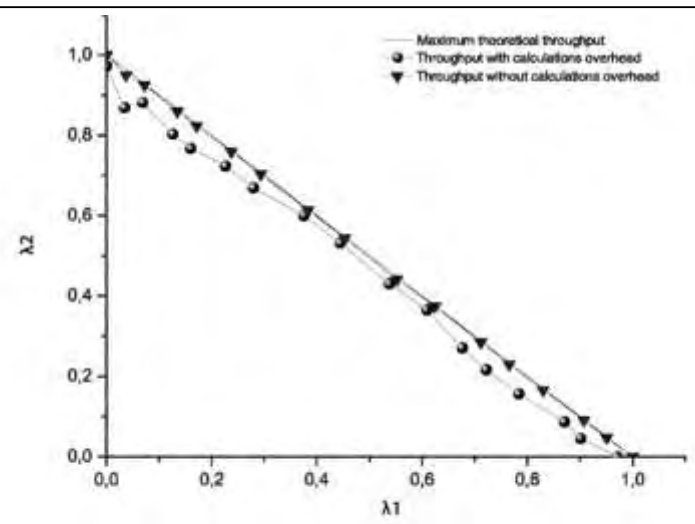

Figure 25: Throughput region of Hybrid Q-CSMA : average results

Program 4 Hybrid Q-CSMA Algorithm (at Link *i* in Time Slot *t*) If $W_i(t) > W_0$ (Q-CSMA Procedure)

1.1 Link *i* selects a random (integer) back-off time T **uniformly in** [0,W_0 — 1] **and waits for** *Ti* **control mini-slots.**

1.2 If **link *i* hears and 'INTENT' message from a link in C(i) before the** (Ti + 1)*th* **control mini-slot, *i* will not transmit an 'INTENT' message anymore.**

Link *i* will set Xi(t) = $X_i(t-1)$ **and will go to 1.4.**

1.3 If **link *i* does not hear an'INTENT' message from any link in C(i) before the** (Ti + 1)th **control mini-slot, it will send an 'INTENT' message to all links in C(i) at the beginning of the** (Ti + 1)th **control mini-slot.**

- If **there is a collision, link *i* will set** $X_i(t) = X_i(t-1)$
- If **there is no collision, link *i* will decide its state as follows:**

  if **no links in C(i) were active in the previous data slot due to the Q-CSMA procedure (i.e.** $NA_i = 0$)

  Xi(t) = 1 **with probability *p***

  $X_i(t)$ = 0 **with probability 1 - *p else***

  $X_i(t)$ = 0

1.4 If *xi(t)* **== 1, link** *i* **will send a 'RESV' message to all links in C**(i) **at the beginning of** *(Wo + 1) tk* **control mini-slot. It will set** *NAi* **= 0) and transmit a packet in the data slot. If xi**(t) **== 0 and link** *i* **hears a 'RESV' message from any link in C**(i) **in the (Wo + 1)tk control mini-slot, it will set** *NAi* **= 1); otherwise, it will set** *NAi* **= 0). If Wi**(t) $<$ W_o (D-GMS Procedure)

2.1 If **link** *i* **hears a 'RESV' message from any link in C**(i) **in the (Wo + 1)tk control mini-slot, it will set** *NAi* **= 1) and xi**(t) **= 0 and keep silent in this time slot.**
Otherwise, link *i* **will set** *NAi* **= 0) and select a random back-off time**

$$Ti \quad (W_o + 1) + W_1 * [B \quad log_b(q_i(t) + 1)]^+ + Uniform[0, W_1 - 1]$$

and waits for *Ti* **controls mini-slots.**

2.2 If **link** *i* **hears a 'RESV' message from a link in** *C(i)* **before the (Ti + 1)tk control mini-slot, it will not be included in x**(t) **and will not transmit a 'RESV' message anymore.**
Link *i* **will set xi(t) = 0.**

2.3 If **link** *i* **does not hear a 'RESV' message from any link in** *C(i)* **before the (Ti + 1)tk control mini-slot, it will send a 'RECV' message to all links in** *C(i)* **at the beginning of** $(T_i + 1)_{tk}$ **control mini-slot.**

If **there is a collision, link** *i* **will set xi**(t) **= 0** If **there is no collision, link** *i* **will set xi**(t) **= 1**

2.4 If **xi**(t) **== 1, link** *i* **will transmit a packet in the data slot.**

5 Experiments

In this section, we test the discussed algorithms in different network scenarios based on those analyzed in Chapter 2. For every simulation, the network on which is was run is given, with the appropriate flows for each case.

5.1 Linear Network - 1-hop collisions

We have tested both Q-CSMA and Hybrid Q-CSMA to measure throughput and delay. The network on which the simulation was done is the one in Figure 26. We consider a 1 hop collision set that is given in (7). Packets arrive at node 1 with a Poisson arrival distribution at a mean rate of $\lambda = 0.45$.

Figure 26: Linear Graph that consists of 8 transmitting nodes plus one dummy node

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix} \tag{7}$$

5.1.1 Q-CSMA

Figure 27 shows average delay in every queue and Figure 28 the average items in queues throughout the simulation (queue size of 55 packets). In Figure 29 you can see the real time and average delay of packets as they enter the network through node 1 until they are delivered to the dummy node.

Figure 27: Q-CSMA Delay in each queue - Linear network

φï—φí—θ

Figure 28: Average queue lengths of Q-CSMA - Linear network

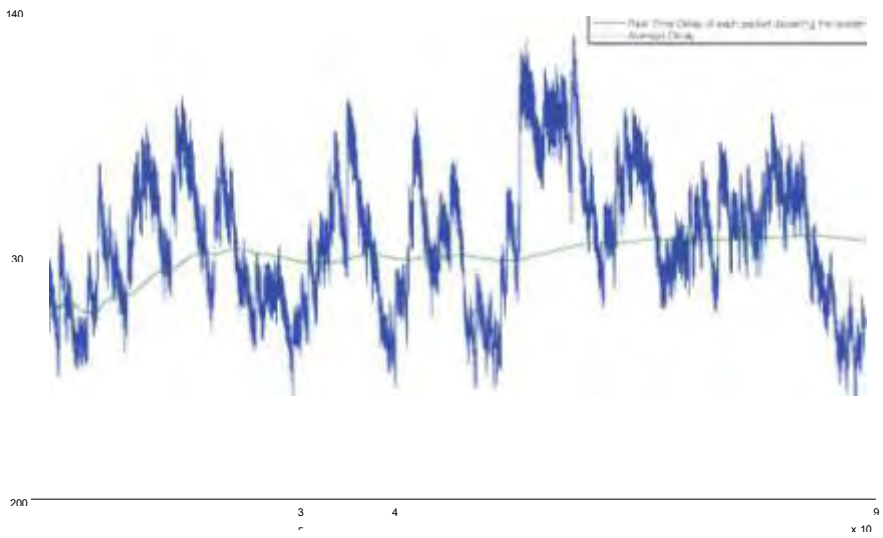

Figure 29: Q-CSMA Delay start to end - Linear network

5.1.2 Hybrid Q-CSMA

Figure 30 shows the average delay in every queue and Figure 31 the average items in queues throughout the simulation (queue size of 55 packets). In Figure 32 you can see the real time and average delay of packets as they enter the network through node 1 until they are delivered to the dummy node.

Comparing these results to the the same using Q-CSMA, one easily notices the huge delay gap between the 2 algorithms, with Hybrid Q-CSMA winning this test. Both manage to keep the system stable under the Poisson arrival rate at the mean value of 0.45.

In Figure 33 one can see a comparison of the throughput of the 2 new algorithms with CSMA and ALOHA for various number of nodes in a Linear Network. Finally in Figure 34, you can see the delay performance for different arrival rates, for the network in Figure 26.

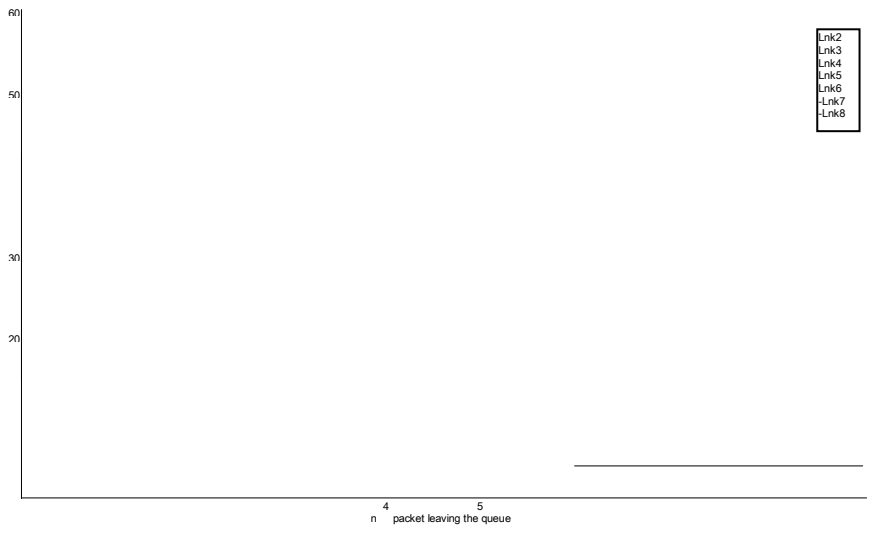

Figure 30: Hybrid Q-CSMA Delay in each queue - Linear network

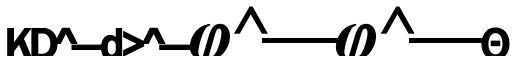

Figure 31: Average queue lengths of Hybrid Q-CSMA - Linear network

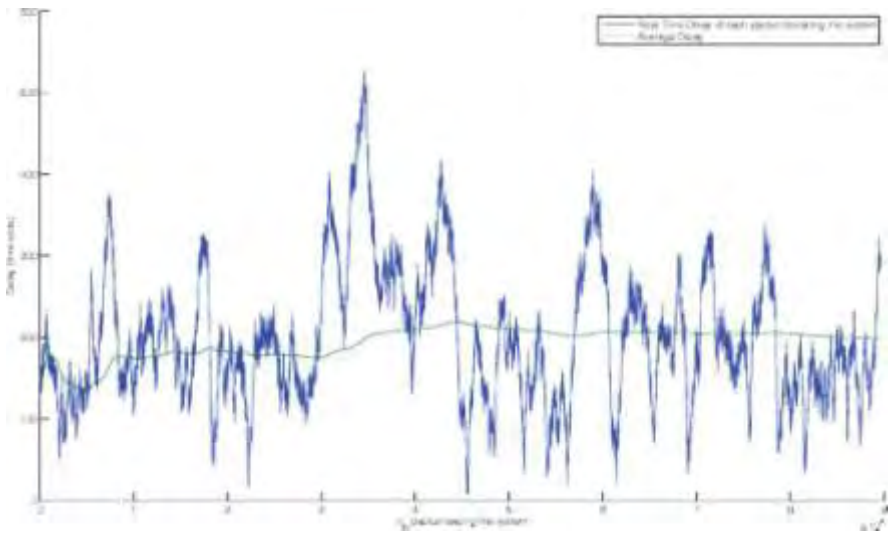

Figure 32: Hybrid Q-CSMA Delay start to end - Linear network

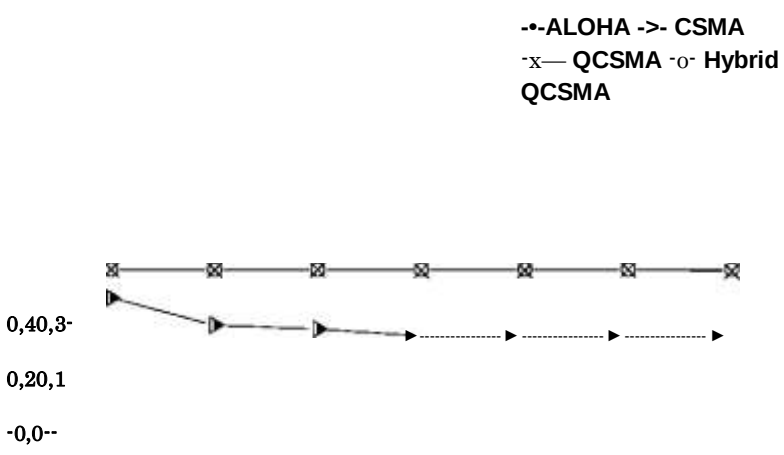

Figure 33: Throughput vs Nodes in linear networks

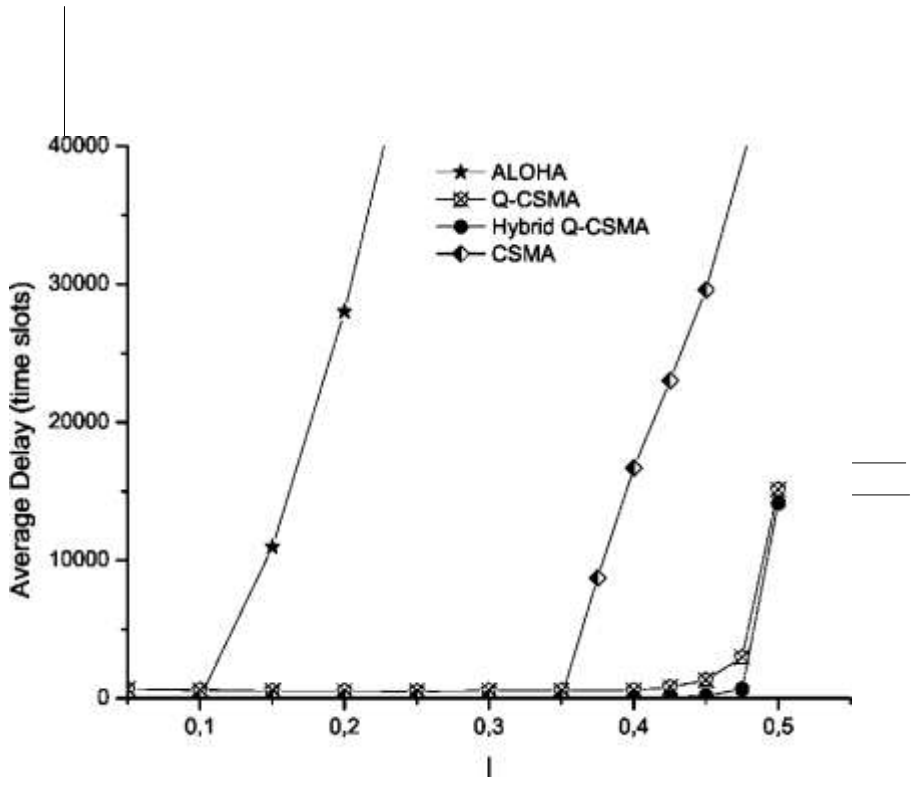

Figure 34: Delay versus arrival rate - Linear Network

5.2 Linear Network - 2-hop collisions

We have tested both Q-CSMA and Hybrid Q-CSMA to measure throughput and delay. The network on which the simulation was done is the one in Figure 35. We consider a 2 hop collision set that is given in (8). Packets arrive at node 1 with a Poisson arrival distribution at a mean rate of $\lambda = 0.28$.

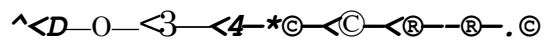

Figure 35: Linear Graph that consists of 8 transmitting nodes plus one dummy node

$$\begin{vmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{vmatrix} \quad (8)$$

5.2.1 Q-CSMA

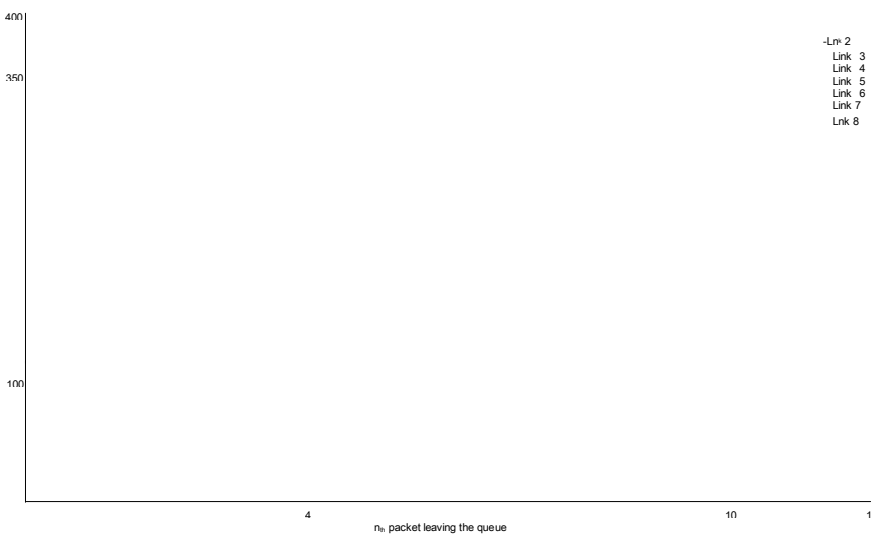

Figure 36: Q-CSMA Delay in each queue - Linear network

Figure 37: Q-CSMA Delay start to end - Linear network

5.2.2 Hybrid Q-CSMA

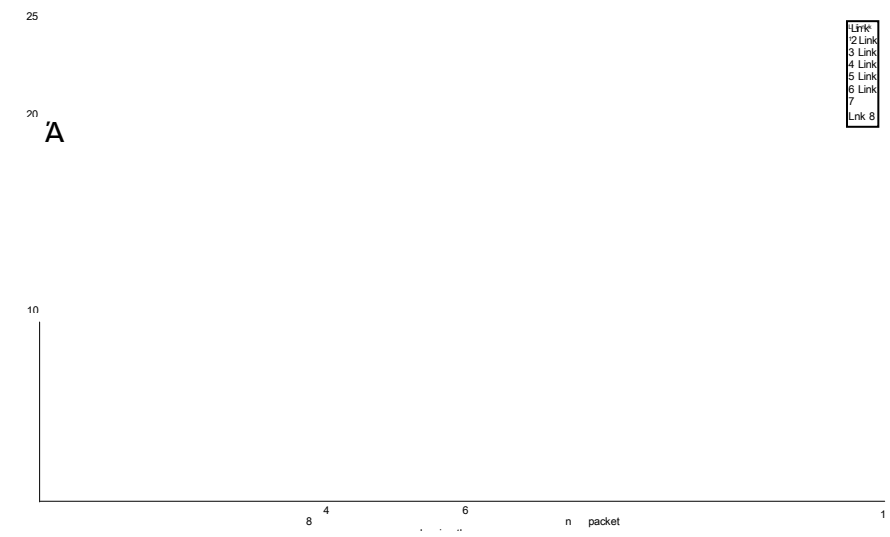

Figure 38: Hybrid Q-CSMA Delay in each queue - Linear network

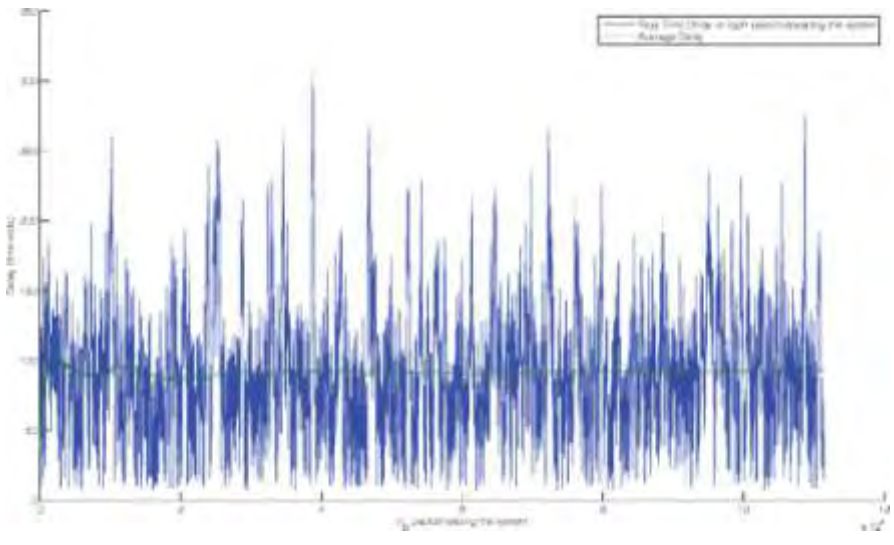

Figure 39: Hybrid Q-CSMA Delay start to end - Linear network

5.3 Grid

We have tested both Q-CSMA and Hybrid Q-CSMA to measure throughput and delay. The network on which the simulation was done is the one in Figure 40. You can notice there exists see 8 packet flows. The one hop directional flows of the same network can be seen in Figure 41.

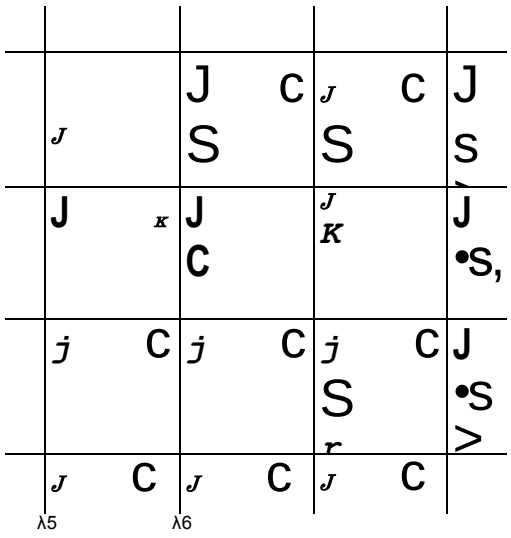

Figure 40: 4 χ 4Gridnetwork with8flows

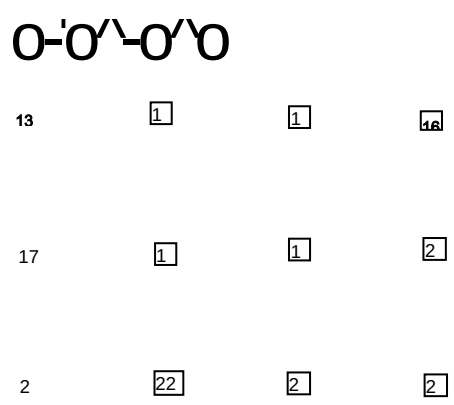

Figure 41: 4 γ 4 Grid network with 24 1-hop directional links

$$
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\tag{9}
$$

We consider the collision set that is given in (9). Packets arrive at links 1,4,7,10,21,22,23 and24withaPoisson arrival distributionatameanrate of $\lambda = 0.22$, under which the system is stable.

5.3.1 Q-CSMA

In Figure 42 you can see the average delay start to end for each network flow. Figure 43 shows the average packets in queues in every link. We consider a queue size of 125 for Q-CSMA.

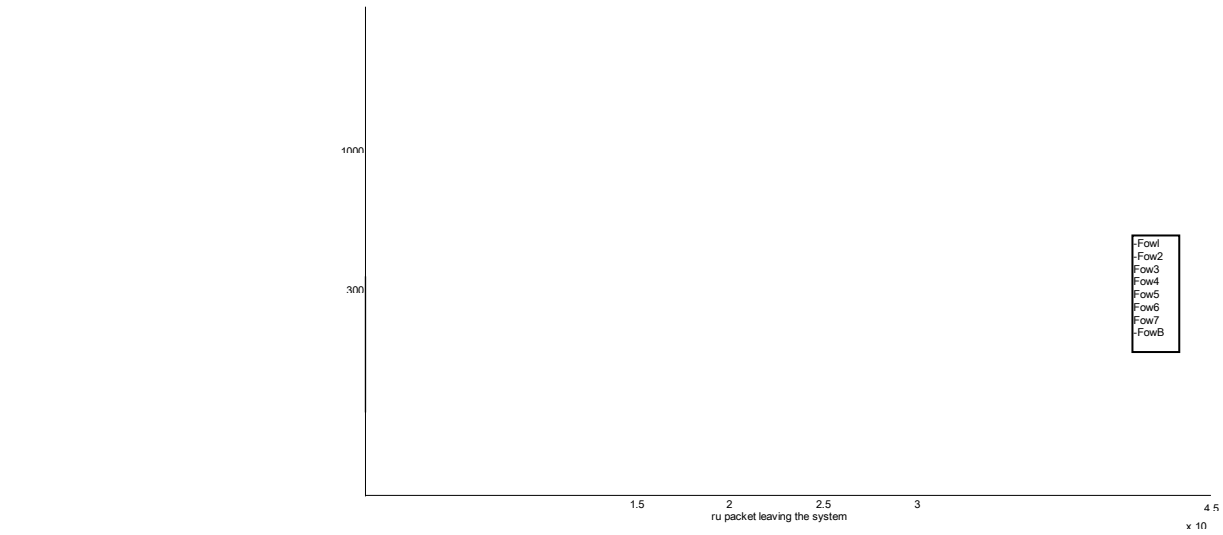

Figure 42: Q-CSMA Delay start to end - All flows

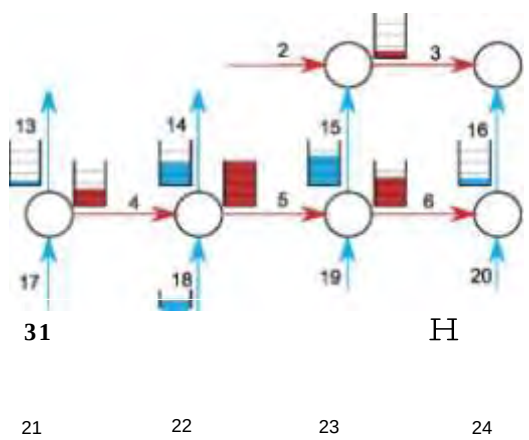

Figure 43: Average queue lengths of Q-CSMA - Grid network

5.3.2 Hybrid Q-CSMA

In Figure 44 you can see the average delay start to end for each network flow. Figure 45 shows the average packets in queues in every link. We consider a queue size of 10 for Hybrid Q-CSMA.

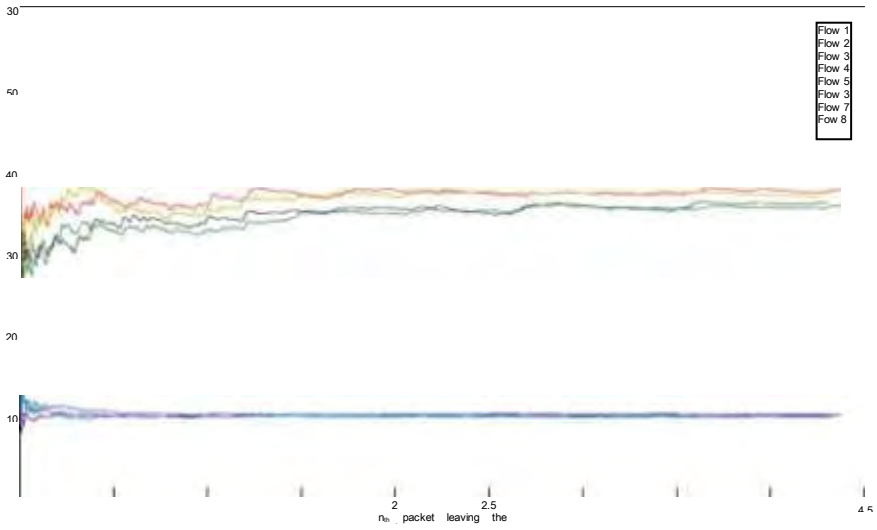

Figure 44: Hybrid Q-CSMA Delay start to end - All flows

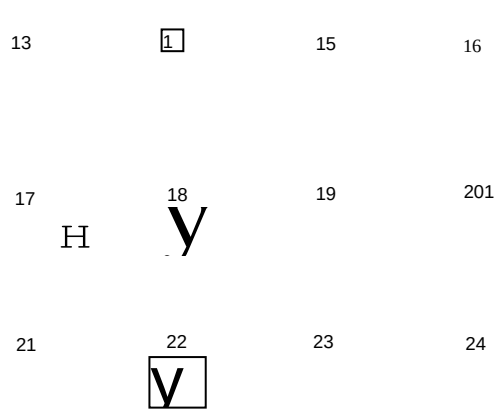

Figure 45: Average queue lengths of Hybrid Q-CSMA - Grid network

5.4 Clique

We have tested both Q-CSMA and Hybrid Q-CSMA to measure throughput. We consider a graph where all possible links exist like the one in Figure 16. In Figure 46 you can see the per-link throughput of a clique network depending on the number of nodes the network contains. We compare the throughput of the 2 previous algorithms with ALOHA and CSMA. The aggregate throughput can be seen in Figure 47.

For the simulation it is considered that all nodes transfer their data in a dummy node. Only one node can be active for a single packet transfer.

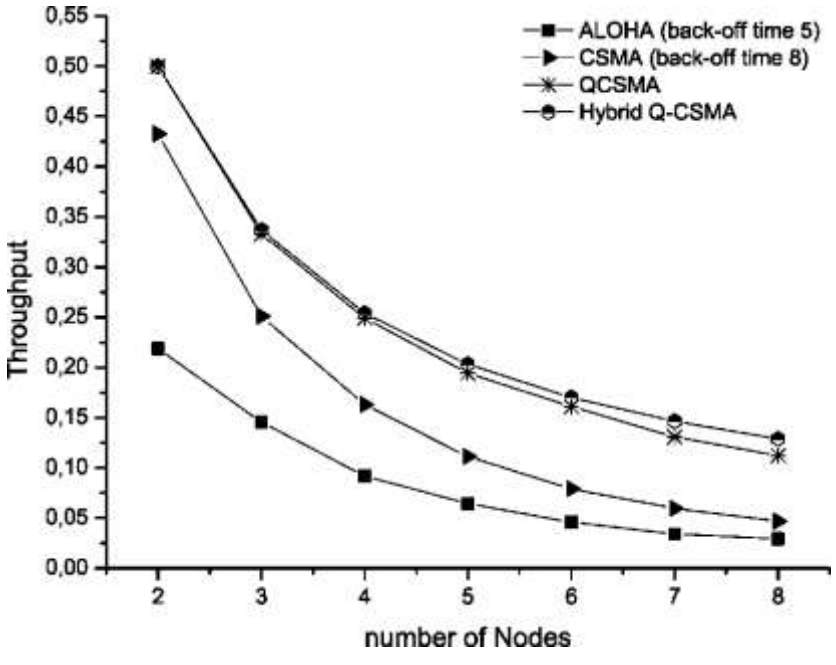

Figure 46: Throughput in a clique network

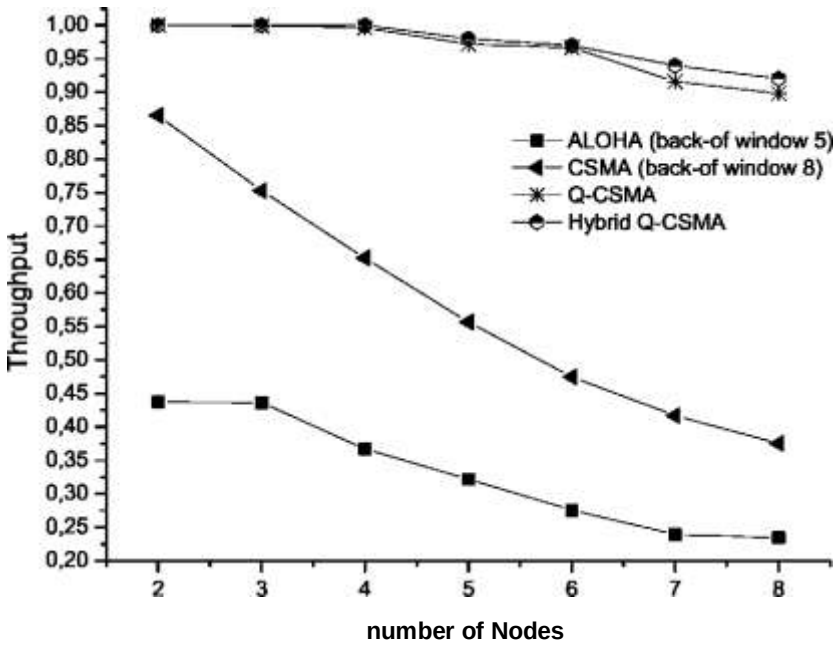

Figure 47: Aggregate throughput in a clique network

6 Node Based Q-CSMA in a realistic environment

6.1 Wi-Fi

Wi-Fi is a branded standard used in wireless communications. If you are reading this, then you have definitely used a Wi-Fi device. The Wi-Fi uses the 802.11 family of standards for implementing wireless local area network (WLAN) computer communication. The final version of the standard uses the so called distributed coordination function (DCF) as the method to access the medium. DCF describes two techniques to employ for data transmission.

The more advanced and widely used technique involves the transmission of hand shacking messages for a link to become active. This technique is used to eliminate collisions and the so called *hidden terminal* problem explained in Section 2. For a link to be used for data transmission, the sender has to reserve it by sending an *RTS* message. Only if the receiver node replies with a *CTS* message will the sender transmit apacketinthedataslot.

6.2 Node Based Q-CSMA

The above idea has been used by the authors of [2] to introduce a node-based implementation of Q-CSMA. The protocol uses an exchange of messages during the control slot in order to silence colliding nodes, just like the RTS/CTS mechanism of 802.11
MAC protocol.

Node based Q-CSMA works like this: At the beginning of every time slot, each node is marked as available sender or receiver of any possible link. Every node n selects randomly a link i for which the contention will take place. For the selected link a back-off counter will be selected, like in pure Q-CSMA. The rest of the links will update their decision using the decision from the previous time slot.

When the back-off counter expires, node n will send a *request to decide* (RTD) message to the receiver node of link **i,** and will wait for a *clear to decide* (CTD) message. This procedure takes a total of control slot time, as half a control slot is used to transmit the message and the rest half to receive an answer. A sender node will only send an RTD message if it haven't heard any RTD or CTD transmissions from neighbor nodes. Upon successful transmission and reception of RTD and CTD messages respectively, sender node n will set it's decision to *1* if neither n nor the receiver of link i were active in the previous data slot. Furthermore, none of sender's neighbors may have been active as receivers in the previous data slot, and none of the receivers neighbors may have been active as senders in the previous data slot.

Following the previous contentions, a sender node n will set it's decision to 1 with probability p, as in link based Q-CSMA. You can find a sample Matlab code of Node Based Q-CSMA in Appendix D (Listing 6).

6.3 Experiments

6.3.1 Linear network

Node based Q-CSMA, cannot achieve the throughput region of link based Q-CSMA, as it uses a different interference model. In Figure 48 you can see the respective results

of node based Q-CSMA for the experiment of Section 5.2.1 (Figure 35), only here the arrival rate is distributed according to the Poisson process with a mean arrival rate of $\lambda = 0.26$, for the system to be stable.

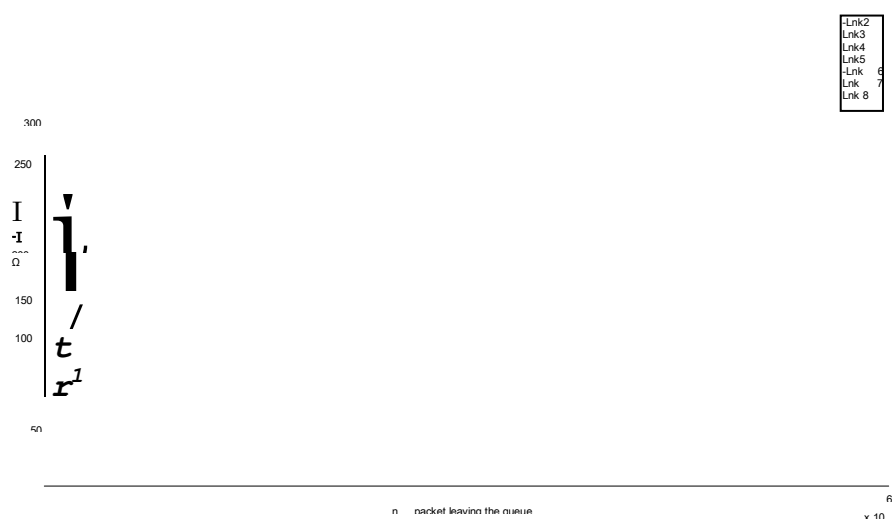

Figure 48: Node based Q-CSMA delay in each queue - Average values

As you can see, the delay metrics of node based Q-CSMA are even worse than those of link based Q-CSMA. In Figure 49 you can see the start to end delay for the same network.

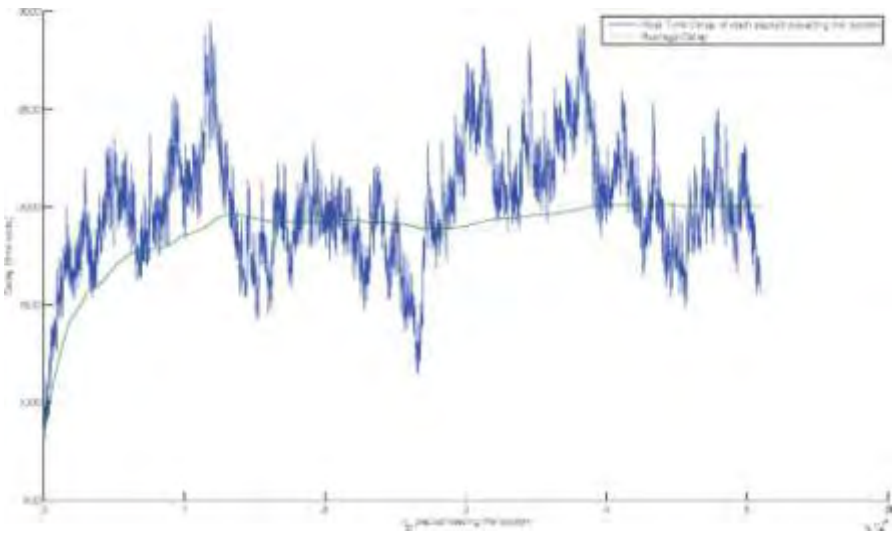

Figure 49: Delay of node based Q-CSMA Delay start to end - Linear network

6.3.2 Grid

Similarly to the previous experiment, node based Q-CSMA cannot achieve the throughput region of link based Q-CSMA in a Grid network, only here things are even worse. For the system to be stable for the experiment of section 5.3.1, the arrival rate in the queues was distributed according to the Poisson process with a mean arrival rate of $\lambda = 0.12$. The delay results for each flow can be seen in Figure 50 and average packets in queues throughout the simulation in Figure 51, for queue size of 85.

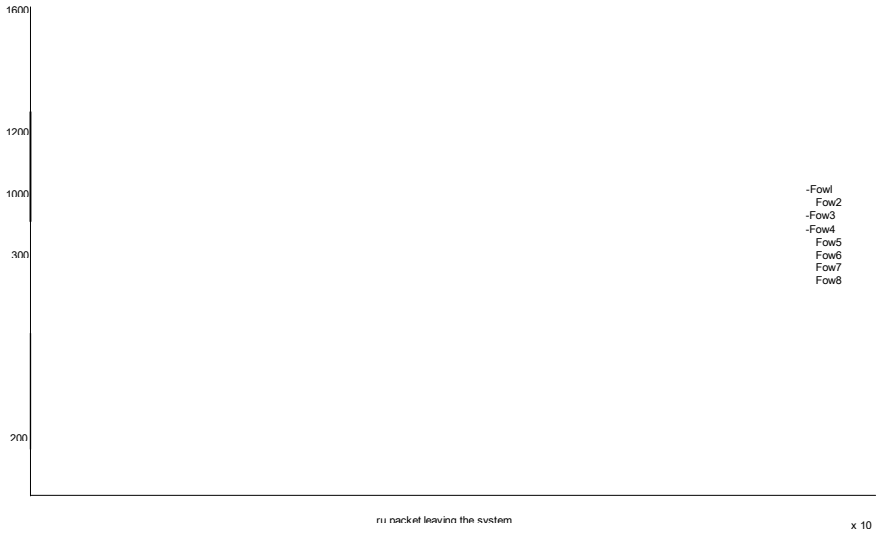

Figure 50: Delay of node based Q-CSMA - Grid network

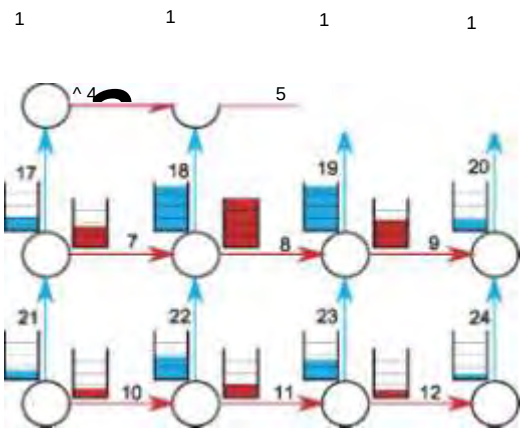

Figure 51: Average queue lengths of node based Q-CSMA - Grid network

6.4 Nitos

In order to get some more realistic results from an existing Wi-Fi topology, we tested different scenarios to acquire metrics for the Nitos testbed.

NITOS (Network Implementation Testbed using Open Source code), is a wireless experimental testbed[8]. It is located in the Department of Computer and Communications Engineering in University of Thessaly. For the purposes of running and testing how node based Q-CSMA works in an existing wireless network topology, we used Nitos as the base network (Figure 52.

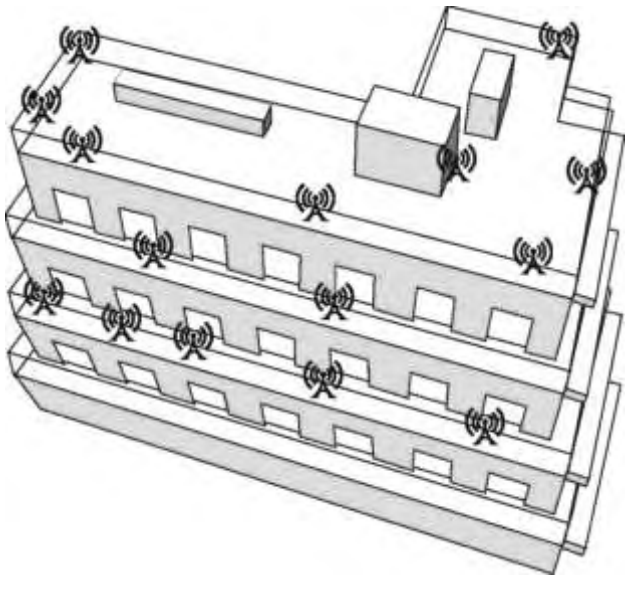

Figure 52: Nitos nodes

The resulted network topology can be seen in Figure 53. For this topology, we used links that had a quality of more than 20% for a *24Mbps* link to be feasible for data transmission. Links with less than 20% quality where still used for interference purposes. The neighborhood $N(n)$ of a node n is given in (10). As you can easily notice, there is no symmetry, as nodes have different transmission power.

[8] http://nitlab.inf.uth.gr/

$$\begin{vmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0
\end{vmatrix} \tag{10}$$

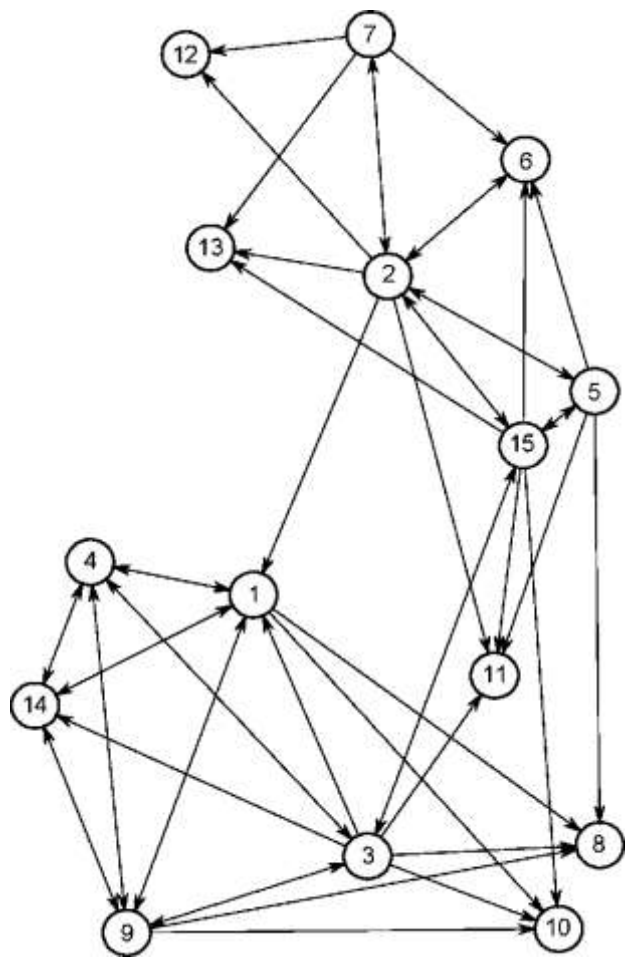

Figure 53: Nitos topology - 24mbps links

Let's assume we have 2 data flows, as shown in Figure 54. The simulations have shown that the throughput region of these 2 flows is the one in Figure 55. The average delay start to end for arrivals distributed according to the Poisson distribution with a mean rate of $\lambda = 0.17$ for each flow can be seen in Figure 56, and the average packets in queues in Figure 57, for queue size of 125.

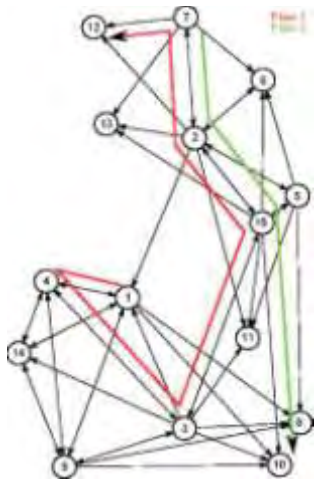

Figure 54: Nitos - 2 flows

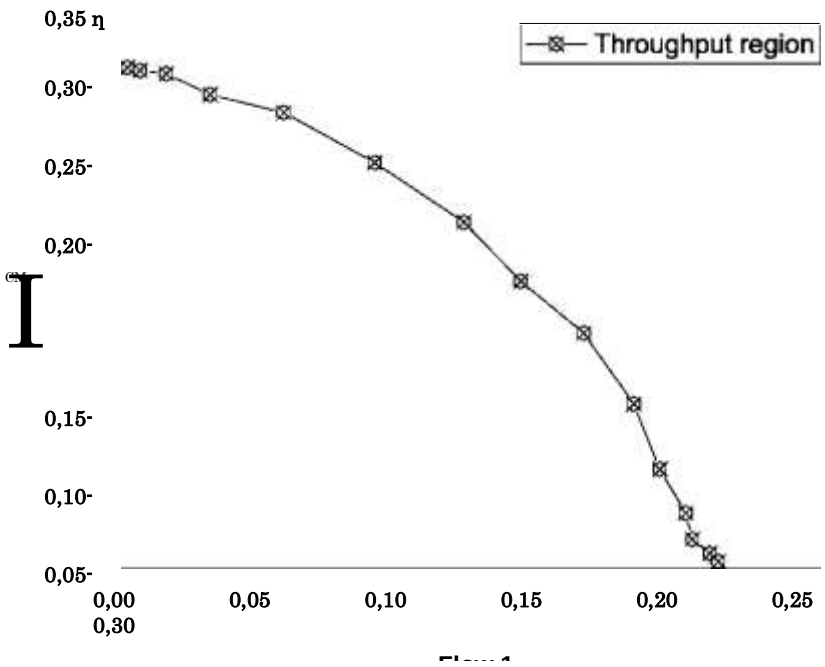

Figure 55: Throughput region of node based Q-CSMA in Nitos - 2 flows

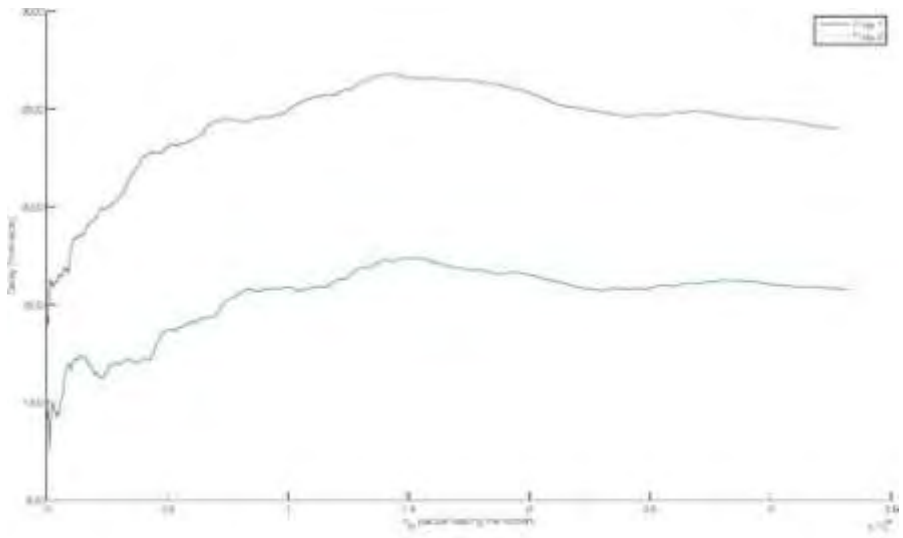

Figure 56: Delay of node based Q-CSMA in Nitos - 2 flows

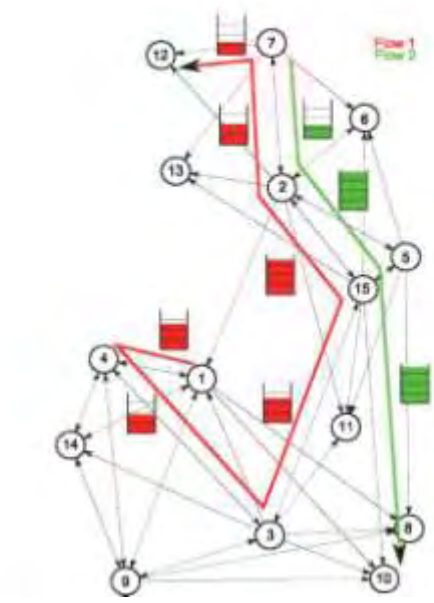

Figure 57: Average queue lengths of node based Q-CSMA in Nitos - 2 flows

Appendix

A Basic Algorithm Matlab Code

```
% Select randomly a feasible schedule m(t) t
m(t);
% Decide X according to X_prv and m(t)
for n = 1:numOfNodes;
    % Nodes that cause collision to node n
    if (m(n) == i)
        % Find any nodes colliding to η that were active
        % in the previous data slot
        Ci = find(C(n,:) == 1);
        Any.of.ci.active = 0;
        for j = find(C(n,:) == 1)
            if(X_prv(j) == 1)
                Any.of.ci.active = 1;
            end
        end
        % Select X(n) = 1 with no nodes in
        % Ci were active in previous
        if (rand <= ρ && Any.of.ci.active == 0)
            X(n) = 1;
        % Select X(n) = 0 with p' = 1-p
        else
            X(n) = 0;
        end
    else X(n) = X.prv(n);

    end
end
% X contains feasible schedule to be used slot
a
in the data
```

Listing 1: Matlab code for *Basic Algorithm*

B Q-CSMA Matlab Code

```
% Backoff vector — uniform random integers [1, W]
Ti = unidrnd(W, 1, numOfNodes);
% intent message heard in this time slot?
intent = zeros(1, numOfNodes);
% 'INTENT' messages
for i = 1:W
    % Send 'INTENT' message to all Ci
    % intent_temp is used so that in the mini—slot nodes
    % can all send INTENT message to
    intent.temp = zeros(1, numOfNodes);
    % For all the nodes that decide in this send intent
    % messages to all the collision set
    for n = find(Ti==i);
        % Nodes that cause collision to node
        for p = find(C(n,:) == 1)
            % Send only if it hasn t in previous control
            % mini— slots. Intent will be bit later
            if(intent(n)==0)
                intent_temp(p) = 1;
            end
        end
    end
    % Up date Intent value —> Intent or I
    intent = intent intent.temp;
end
% Select randomly a feasible schedule m(t) in
m(t);
% Decide X according to X_prv and m(t)
for n = 1:numOfNodes;
    % If it heard no INTENT message
    if(intent(n)==0)
        % No c o llisio n s o r any o the r — set decision to 1
        m(n) = 1;
        % Find any nodes colliding to active
        % in the previous data slot
        Ci = find(C(n,:) == 1);
        Any.of.ci.act ive = 0;
        for j = find(C(n,:) == 1)
            if(X_prv(j) == 1)
                Any.of.ci.act ive = 1;
            end
        end
        ρ(η) = exp(items.in.queue(n)) / (1 + exP(items .in.queue(n)));
        % Select X(n) = 1 with probability nodes in
        % Ci were active in previous
        if(rand <= ρ(η) && Any.of.ci.act ive == 0)
            X(n) = 1;
        % Select X(n) = 0 with p = 1—p
        else
            X(n) = 0;
        end
    else
        X(n) = X.prv(n);
    end
end
% X co nt a ins a feasible schedule to be used in the data slot
```

Listing 2: Matlab code for Q-CSMA Algorithm (at link i in Time Slot t)

CHybridQ-CSMAMatlabCode

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%START OF QCSMA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Backoff vector for the nodes that will run
QCSMA (1.1) TLQCSMA = zeros ( 1 , numOfNodes ) ;
for i = find(log(0.1*items_in_queue) > log(0.1*q_thresold)); TLQCSMA(
    i ) = unidrnd(W);
end
%was intent message heard in this
time slot? intent = zeros (1, numOfNodes ) ;
for i = 1 : W
    %send INTENT message to all Ci
    %intent-temp is used so that in the same control
    mini— s l o t nodes %can all send INTENT message to
    their neighbors intent_temp = zeros ( 1 , numOfNodes ) ;
    %for all the nodes that decide in this
    mini—slot send intent %messages to all the
    collision set for η = find (Ti_QCSMA==i )
        %Nodes that cause collision to node n
        for p=find(C(n,:)==1)
            %Send only if it hasn t heard already in previous control
            %mini—slots. Intent will be updated a bit
            later
            if(intent(n)==0)
                intent.temp (p) = 1;
            end
        end
    end
    %Up date Intent value — > Intent or I
    nt ent _t emp intent = intent | intent.temp ;
end

%%Now we now whether a node has or has not heard
an INTENT msg pi = zeros (1, numOfNodes ) ;
%%for all the nodes running QCSMA do the
following for η = find (Ti_QCSMA>0)
    QCSMA_cnt = QCSMA_cnt + 1;
    %if it heard no 'INTENT' message
    if(intent(n)==0)
        %prevent 'Inf' return value of
        exp (710) if ( items.in.queue (n) <10)
            ρ i ( η ) = exp ( items.in.queue (n)) / (1 + exp ( items.in.queue (n) ) ) >
        else
            %pi(n) = exp (10++)/(1+exp (10++)) == 1
            pi(n) = 1;
        end
        %select Xi = 1 with pi if none of Ci were
        active in previous %time slot
        if ( rand <= pi (n) &&NA(n) = = 0)
            X(n) = 1;
        %select Xi = 0 with 1—pi
        else
            X(n) = 0;
        end
    %if it heard an INTENT message or if there
    was a collision else
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% POSSIBLE ERROR
        IN PAPER %%%% %addon so we get
        valid schedules%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if (DGMS.prv.slot(n) == 1)
            X(n) = 0;
        else
            X(n) = X.prv(n) ;
        end
    end
end

%o%END OF QCSMA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 3: Matlab code for Hybrid Q-CSMA Algorithm (Q-CSMA part)

```
9ffio%%%%%%%%%%%%%^

%W0+1 -th control mini slot (1.4 +
% first send all res v messages
resv = zeros (1, numOfNodes ) ;
for n = 1 : numOfNodes
    if (X(n) = =1)
        for ρ = find (C(n,:) = = 1)
            resv(p) = 1;
        end
        NA(n) = 0;
    end
end
%then check if X(n) = = 0 && resv (n) = = 1
for η = 1 : length (Nodes_running_QCSMA)
    if (X(Nodes_running_QCSMA(n)) = = 0 && resv (Nodes_running_QCSMA (n)) = = 1)
        NA(Nodes_running_QCSMA(n)) = 1;
    else
        NA(Nodes_running_QCSMA(n)) = 0;
    end
end

for η = 1 : length (Nodes.running.DGMS )

    if ( resv (Nodes.running.DGMS (n)) = = 1)
        NA(Nodes_running_DGMS (n) ) = 1;
        X(Nodes_running_DGMS (n) ) = 0;
    else
        NA(Nodes_running_DGMS (n) ) = 0;
    end
end
%%END OF W0+1 -th control mini .slot (1.4 2.1)
9 ^Ü%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 4: **Wq**+1 control mini slot

```
%%START OF D-GMS
%$%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Backoff vector for the nodes that will
run DGMSTL.DGMS = zeros ( 1 , numOfNodes ) ;
for i = find(log(0.1*items_in_queue) <= log ( 0 . 1 * q.thresold ));
      if (NA( i) = = 0)
            TLDGMS(i) =W+ 2 + Wl*(B-logb ( it em s _i n_q ueue ( i ) + 1, b ) ) + unidrnd(Wl) 1;
      end
end
%s o rt the Ti_DGMS in ascending
order Ti.ordered = sort ( TLDGMS ) ; %d e l et
e all the zeros Ti.ordered ( Ti.ordered
= = 0) = [] ;
%temp value used within the
for loop k_old = [];
%find the order the nodes are supposed to speak and
send resv messages for η = 1 : le ngt li ( T i _o r d er e d );
      k = find (Ti_DGMS==Ti_ordered (n)) ;
      %f lag used within the if 's below — flag is by
      default 1 unless a %vector of nodes that has the
      same backoff time occurs flag = 1;
      if ( length ( k_old) = = length (k) && le ngt h ( k _o 1 d ) > 1)
            if(eq(k_old ,k)) flag = 0;
            end
      end
      % sending resv messages
      to Ci if ( length (k) = = 1 && flag =
      = 1) for p = find (C(k,:) = = 1) if
      (resv (k) = = u)
                  resv(p) = 1;
               end
            end
      end
      %sending resv messages to Ci - vectors of nodes with
      the same %backoff time will only be checked once —
      flag will be set to 0 the %second , third and so
      on times the same vector is checked if ( length (k)>1&& flag==1)
            resv.temp = zeros (1, numOfNodes ) ;
            for i=1:length(k)
                  for p=find(C(k(i),:)==1)
                        if (resv(k( i)) = = 0)
                              resv.temp (p) = 1;
                        end
                  end
            end
            resv = resv | resv.temp ;
      end
      k_old=k ;
end
DGMS_prv_slot = zeros (1, numOfNodes ) ;
%%Now we now whether a node has or has not heard an
'RESV' msg
%%for all the nodes running DGMS that are allows to speak do
the following
for η = find (Ti_DGMS>0)
      DGMS_cnt = DGMS.cnt + 1;
      %if it heard no 'RESV' message
      if( resv(n)==0)
            X(n) = 1;
            DGMS_prv_slot (n) = 1;
      end
end

%%END OF D-GMS
```

Listing 5: Matlab code for Hybrid Q-CSMA Algorithm (D-GMS part)

D Node Based Q-CSMA Matlab Code

```
%%%%%%%%
% 1 .        %
%%%%%%%%
%vector  AS = 1    if  node  n  is  availlable
as  a  sender AS =  ones ( 1 , numOfNodes ) ;
%%Ed it  here  AS (num OfNodes)  according  to  the
network  topology %vector  AR= 1  if  node  n  is
availlable  as  a  receiver AR =  ones ( 1 , numOfNodes ) ;
%% Edit  here  AR (1)  =  0  according  to  the
network  topology %Backoff  vector  —
Ti  =  ones ( 1 , numOfLinks ) * (W+ 1) ;
for  n =  1 : numOfNodes
    %find  all  links  node  n
    is  a  sender k =  find (L(n,:) = =1);
    %select  a  link  to  use  in
    random if ( isempty (k)  ==  0 )
        selected.link  = k( unidrnd ( length (k ) ) ) ;
        %remember  that  link
        remember.link (n)  =  s e l e c t e d _ 1 i n k ;
        %select  a  back—off  time  for
        this  link Ti( selected-link)  =  unidrnd
        (W) ; for  i = 1 :  length (k)
            i f ( k ( i )  ~=  selected-link)
                X(k(i))  =  X_prv(k(i));
            end
        end
    end
end

%%%%%%%%%%%%
%  2  3  4  %
%%%%%%%%%%%%
for i = 1 : W
    RTD = zeros ( 1 , numOfNodes ) ;
    CTD =  zeros ( 1 , numOfNodes ) ;
    %FIRST  sub—mini—slot
    for  n=find(Ti==i)
        if(AS(S(n)) == 1)
            RTD (R ( n) )  = RTD (R ( n ))  +  1;
            AS(S(n))  =  0;
            AR(S(n)) = 0;
            %%%% 2 %%%%
            for  p  =  find (N(S(n) ,:) = = 1)
                if(p  "= R(n))
                    RTD(p)  = RTD(p)  +  1 ;
                    AR( p ) = 0 ;
                end
            end
        end
    end
    %SECOND sub—mini—slot
    for  n=find(Ti==i)
        if (AR(R(n))== 1 && RTD(R(n))==  1)
            CTD(s(n))  = CTD(S(n))  +  1;
            AS(R(n))  =  0;
            AR(R(n)) = 0;
            %%%% 3 %%%%
            for  p  =  find (N(R(n) ,:) = =1)
                %%Edit  here  numOfNodes  according  to  the
                network  topology if (p  -=  S(n)  && p-=  numOfNodes)
                    AS ( p ) = 0 ;
                    CTD(p) = CTD(p) + 1;
                    X( remember _1 ink (ρ ) )  =  X.prv ( remember.link (p ) ) ;
                end
            end
        end

    end
    %DECISION
    for  n=find(Ti==i)
        if (CTD(S(n)) == 1)
            if (X_prv (n)  ==  1
                | (  ACT(S(n))  ==
                0
                    && ACT (R (n ))  == 0 &&
                        NR(s(n))  == 0
                            && NS(R(n))  ==  0))
                ρ i ( η )  = exp (log(0.1* items_in_queue(n)))/(1 + exp (log(0.1* items.in.queue (n) ))) ; if(rand <=
                pi(n)) X(n)  =  1;
                %select Xi = 0 with 1—pi
                else
                    X( n) = 0;
                end
            else
                X(n) = 0;
            end
        else
            X(n)  =  X.prv(n);
        end
```

```
    end
end
X_prv = X;
%r      es      et   vertices
ACT =  zeros ( 1 , numOfNodes ) ; NR
=  zeros ( 1 , numOfNodes ) ; NS =
zeros ( 1 , numOfNodes ) ;

for   n =   1: numOfLinks

     if (X(n)  ==  1)
         ACT( S ( n ) ) = 1 ;
         ACT(R(n)) = 1;
          for   p =   find (N(S(n) ,:) = = 1)
               if (p      R(n))
                    NS(p)      1;
               end
          end
          for   p =  find (N(R(n) ,:) = = 1)
               if(p  "=  S(n))
                    NR( p ) = 1 ;
               end
          end
    end
end
```

Listing 6: Matlab code for Node Based Q-CSMA Algorithm

List of Figures

References

[1] N. Abramson, *The Aloha System - Another Alternative for Computer Communications,* 1970.

[2] Jian Ni, Bo Tan and R. Srikant, *Q-CSMA: Queue-Length Based CSMA/CA Algorithms for Achieving Maximum Throughput and Low Delay in Wireless Networks,* 16 Dec 2009.

[3] L. Tassiulas and A. Ephremides, *Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,* IEEE Trans. Autom. Control, vol. 37, no. 12, pp. 19361948, Dec. 1992.

[4] L. Tassiulas, *Scheduling and performance limits ofnetworks with constantly changing topology,* IEEE Trans. Inf. Theory, vol. 43, no.3, pp. 10671073, May 1997.

[5] Giuseppe Bianchi, *Performance Analysis of the 802.11 Distributed Coordination Function,* IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 18, NO. 3, MARCH 2000.

[6] Theodore S. Rappaport *Wireless Communications* Principles and Practice Second Edition

[7] Charles Bordenave, David McDonald, Alexandre Proutiere *Asymptotic stability region ofslotted-Aloha* CoRR abs/0809.5023 (2008)

[8] Mathilde Durvy, Olivier Dousse, Patrick Thiran *OntheFairnessofLargeCSMA Networks*

[9] Leonidas Georgiadis, Michael J. Neely, Leandros Tassiulas *Resource Allocation and Cross-Layer Control in Wireless Networks* Foundations and Trends in Networking Vol. 1, No 1 (2006) 1144

[10] Harry Perros *Computer Simulation Techniques: The definitive introduction!*