A probabilistic approach for evaluating parameters of the Distributed Scheduling Scheme of the 802.16

Valeria Loscri', Gianluca Aloi
University of Calabria, D.E.I.S. 87036, Arcavacata di Rende (CS)
{vloscri, aloi} @deis.unical.it

Abstract

In this paper we propose a dynamic approach for setting several parameters of the IEEE 802.16 Coordinated Distributed Scheduling scheme (CDS), using an opportunistic fashion. In particular, we evaluate the impact of the dynamic tuning of the XmtHoldoffExponent over the CDS scheme. The main idea of the dynamic approach is based on the simple observation that, nodes in a network, need different resources allocation in function of its own traffic load. Our approach is based on the observation of the data queue size of each node. Nodes with larger data queues will be able to set an opportunistic value (the XmtHoldoffExponent) to reduce the acquisition latency of a new control slot. Control slots are used by each node to try to reserve new data slots and to send scheduling information to neighborhoods. Extensive simulation study shows how the dynamic approach permits to select the appropriate XmtHoldoffExponent, improving system performances evaluated in terms of end-to-end delay and throughput.

Key-words: *Mesh networks, distributed scheduling, dynamic approach, 802.16*

1. Introduction

The IEEE 802.16 standard [1, 2], promoted by WiMAX (Worldwide Interoperability for Microwave Access) forum [3], will be the leading technology for the wireless provisioning of broadband services in wide area networks. IEEE 802.16 supports mesh connectivity in a distributed way, and can be implemented in WMNs. The MAC layer is based on time division multiple access (TDMA) to support multiple users. Furthermore, the MAC layer supports two kinds of modes, namely point-to-multipoint (PMP) mode and mesh mode (MM). There is a plethora of works about the IEEE 802.16 on the PMP mode [4, 5, 6]. In PMP mode, communication is only possible between a base station (BS) and a subscriber station (SS). In mesh mode multihop communication is possible between mesh subscriber stations (M-SSs).

A very important factor that influences the performance of mesh networks is the assignment of available network resources. The assignment of resources can be organized in centralized or distributed manner.

In this work we focus on the Coordinated Distributed Scheduling scheme (CDS). The main idea of the coordinated distributed scheduling is to let nodes calculate the usage of transmission opportunities on the neighborhood scheduling information. To achieve this goal, nodes will exchange 2-hop neighborhood scheduling information with each other. Since nodes shall run the scheduling algorithm independently, a common algorithm is necessary for each node in the neighborhood to calculate the same schedule. This algorithm must be random and predictable. One feature for the contention in this protocol is the pseudo-random election algorithm based on the transmission schedules of two-hop neighbors. Nodes in the network are allowed to send their own schedule in a control slot acquired through a pseudo-random algorithm called Mesh Election function. One key factor of this algorithm is that the node persistence for the control slot acquisition is related to a parameter called XmtHoldoffExponent (XHE). Currently, the standard does not give rules for XHE setting and users are free to choose this parameter as they like.

Our work start from the assumption that the correct choosing of XHE values should permit better performance in terms of throughput and average end-to-end data packet delay. In order to verify this assumption, we developed a probabilistic algorithm, based on the buffer data size of each node, that sets XHE values in a dynamic fashion. Through the use of a well-known simulation tool, ns2 [7], we will show how our algorithm permits to obtain higher throughput and lower data packet delays. Our experiments show that, an appropriate configuration of scheduling parameters, such as *XmtHoldoffExponent (XHE)*, may potentially improve the overall scheduling performance in terms of throughput and delay. The remainder of the paper is organized as follows. In Section 2, we briefly review IEEE 802.16 Mesh mode and the details of coordinated distributed scheduling algorithm. We then give, in Section 3, details of the proposed dynamic approach. In Section 4, we provide extensive simulation studies on scheduling performance both when the dynamic algorithm is used and not. We conclude the paper in Section 5.

2. IEEE 802.16 Mesh Mode

In order to achieve efficient collision-free multi-hop data transmissions, the Mesh Mode defines three scheduling schemes: Centralized, Coordinated Distributed, and Uncoordinated distributed. Our attention will be focused in Coordinated Distributed scheme. The frame structure is subdivided in two parts, respectively, *Control Subframe* and *Data Subframe*. In other words, control message and data packets are allocated in the same frame structure but in different time slots (see Figure 1).

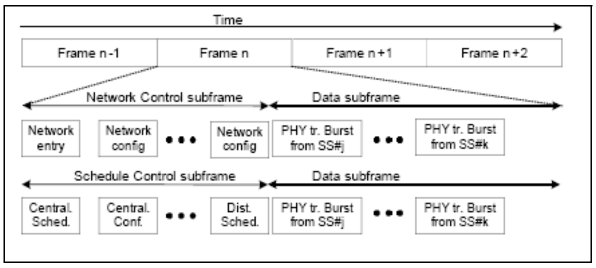

Figure 1. Frame structure in Mesh Mode.

Moreover, two complementary types of Control sub-frame are defined in Mesh Mode but each frame can only have one of them. One is the *Network Control* sub-frame, his transmission occurs periodically and it is used to create and to maintain the cohesion between different systems. The other, formerly the *Schedule Control* sub-frame, occurs in all other frames without a *Network Control* sub-frame. Every control sub-frame consists of 16 transmission opportunities and every transmission opportunity consists in seven OFDM symbols time. The *Data Sub-frame* is situated at the end of the *Control Sub-frame* and it is divided into mini-slots. The mini-slot is the basic unit for resource allocation. In Coordinated Distributed Scheduling all the stations shall indicate their own schedule by sending a MSH-DSCH regularly. Mesh Distributed Scheduling messages (MSH-DSCH messages) are transmitted in the *Schedule Control* sub-frame.

2.1 Coordinated Distributed Scheduling

In this sub-section, we briefly review the coordinated distributed scheduling method in IEEE 802.16 Mesh mode. The assignment of transmission opportunities in the data sub-frame is managed by a scheduling mechanism. As we already said, we focus on the coordinated distributed scheduling mechanism (C-DSCH) which employs a three-way handshake to request, grant and confirm transmission opportunities in the data sub-frame. MSH-DSCH messages are used to carry these requests, grants and confirmations. MSH-DSCH messages are sent within the C-DSCH part of the *Schedule Control* sub-frame (Figure 1). The transmission timing of the MSH-DSCH messages plays key role in our work. The MSH-DSCH messages scheduling is based on a distributed election mechanism which supports distributed coordinated transmission timing of periodic broadcast messages in a multi-hop network without explicit schedule negotiation. It provides collision-free and fair transmissions within the two-hop neighborhood of each node. To avoid collisions of MSH-DSCH messages, every node must inform its neighbors about the next MSH-DSCH transmission time. In order to save network resource and to reduce the signaling overhead, mesh nodes do not broadcast the exact *NextXmtTime Opportunity* (NXTO), in which a node is able to transmit in a collision-free fashion, but only a time interval, the *NextXmtTimeInterval* (NXTI). A node computes the NXTO during its *Current Transmission Time* (CTT). The standard imposes that a node has to wait a time interval (*XmtHoldoffTime* - XHT) after its CTT before to transmit. XHT is defined as:

$$XHT = 2^{(XHE\ +4)} \qquad (1)$$

Practically, a node, after its CTT, is not allowed to transmit for a time period equal to the XHT defined through its *XmtHoldoffExponent* (XHE).
The XHE term must assume a value in the range between 0 and 7. After XHT, a node has to compete with all of its two-hop neighbors to acquire a new transmission opportunity NXTO.
The next transmission interval to transmit is set as:

$$2^{XHE} * NXM < NXTI \le 2^{XHE} * (NXM +1) \qquad (2)$$

The NXTI is established based on a parameter called in the standard *NextXmtMx* (NXM). In order to establish its competing neighborhood the node runs a mesh election function in its CTT (current transmission time) based on the information about the NXTI of its neighbors and as well as their XHE values. For example, if NXM = 2 and XHE = 4 the station would be eligible between 33 and 48 transmission opportunities.

2.2 Distributed Election Algorithm

Every node calculates its NXTI during the current transmission according to the distributed election algorithm defined in [1, 2]. In this algorithm one node sets the first transmission slot, formerly the NXTO, just after the XHT as the temporary next transmission opportunity. Let us to call the current node that is running the mesh election function in its CTT, node A. Node A is trying to find a new transmission time after its XHT is spent. In order to do that, node A sets the next slot after the XHT as temporary slot and it will check whether this slot can be reserved or not. In order to explain the mechanism we define three different types of two-hop neighbors that have to be included in the competing neighborhood of the current node A, for different reasons. Specifically, node B has a NXTI (remember that mesh nodes do not broadcast the exact NXTO but only the NXTI) that includes the temporary transmission slot, so we have to include B as potential competing node. Node C computes its *EarliestSubsequenceXmtTime* parameter (ESXT, equal to NXTI + XHT) and it is ≤ the temporary transmission slot, and C has to be included ad competing node. Concerning node D, it has to be included as competing node because node A did not receive any information about it. The three different types of competing nodes are shown in Figure 2 a.

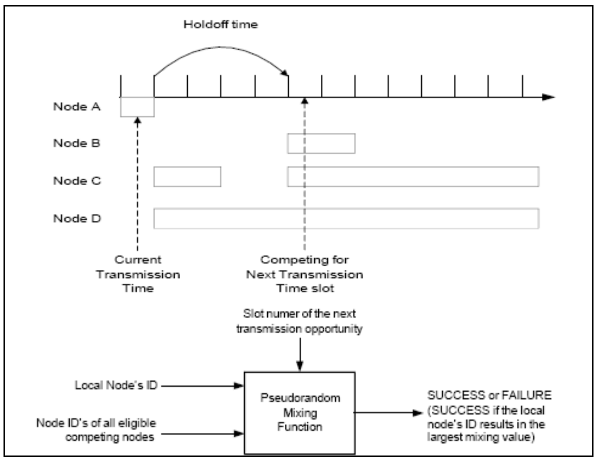
Holdoff time
Node A
Node B
Node C
Node D
Current Transmission Time
Competing for Next Transmission Time slot
Slot numer of the next transmission opportunity
Local Node's ID
Node ID's of all eligible competing nodes
Pseudorandom Mixing Function
SUCCESS or FAILURE (SUCCESS if the local node's ID results in the largest mixing value)

Figure 2: a) Competing Nodes for the Next Transmission time slot. B) Pseudorandom Mixing Function

The mesh election algorithm used to establish whether a temporary slot can be reserved as NXTO is a pseudo-random function which uses the slot number and the Node's ID as the inputs and is executed at each node. It generates pseudo-random values depending on the input. The node wins when its result is the largest mixing value (Figure 2 b). When any node wins, it sets the NXTO as its next transmission time and logically it shall communicate this information to all the neighbors by sending the corresponding packet. In the case a node has not won, it chooses the next NXTO and repeats the algorithm as many times as it needs to win. Since the exact scheduled NXTO of the neighborhood is unknown, as implementation issue, one may define NXTO to be the last NXTO within the interval when calculating *EarliestSubsequentXmtTime*. The holdoff exponent value decides the channel contention time of node so it is an important parameter that can affect the system performance. MSH-DSCH messages are transmitted regularly by every node throughout the whole mesh network to distribute nodes schedules. As we already said, in this paper we focus on the CDS and analyze the transmission timing of the MSH-DSCH messages because this has much influence on the overall network performance.

3. A Dynamic Approach for setting parameter of the Coordinated Distributed Scheduling

As we already seen, to avoid collisions of MSH-DSCH messages every node must inform its neighbors about the next MSH-DSCH transmission time (or transmission opportunity). To save network resources and to reduce the control overhead mesh nodes do not broadcast the exact NXTO but only the *NextXmtTimeInterval* (NXTI) which is a series of one or more transmission opportunities. Therefore the IEEE 802.16 standard defines the parameters NXM and XHE. Since each node includes the own parameters and the parameters of all one-hop neighbors, it is able to calculate the NXTI of all nodes in the two hop neighborhood. In practice, a node has to wait a minimum of XHT (as defined in the previous section) after the current XMT before it can send the next MSH-DSCH message. In [8] the authors developed an analytical model to evaluate the system performance of the distributed scheduler of the IEEE 802.16 mesh mode. Specifically, they developed methods for estimating the distributions of the node transmission interval and connection setup delay. Based on their analysis, the nodes with real time traffic shall have smaller holdoff exponents because they can have more chance to obtain data channel. However, too many nodes with small exponent value generate intensive competition that wastes system resource. A good reservation scheme should guarantee the bandwidth allocation fairness, improve the channel utilization and should adjust the exponent values of the nodes adaptively according to the competition node number variation. Based on these considerations we developed a probabilistic dynamic algorithm based on the size of data queues of each node that in a dynamic way adjust their holdoff exponents. Loosely speaking, each node evaluates the size of the data queue and it adjusts its XHE according to the value of this size: the greater is the value of the size (the greater is the number of data packets a node has to send) the smaller will be the XHE because the node shall have more chance to obtain data channel. Vice-versa, the smaller is the size of data queue the greater will be the XHE. A pseudo-code that shows the fundamental operations of the Dynamic Approach is the following:

```
const int Max_Buffer_Size;
var    int queue_size;
//Node A is in the current transmission opportunity
Node A evaluates its queue_size;
//The number of data packets in the queue of the node
// A is higher than the half of the maximum of the
// queue size
if (queue_size  >  Max_Buffer_size/2) then
    XHE (Node A) = 0;
//The number of data packet stored in the A data
//queue is less than the half of the maximum size of the
//queue
else XHE (Node A) = probabilistic value between 1,
2 or 3;
//the probabilistic value is evenly chosen between 1,
//2and 3
```

This approach is probabilistic in the sense that after a node evaluated its data *queue_size* whether this size is less than the half size of the entire capability of the queue (i.e., if the queue size is 100 and the number of data packets is less or equal than 50), the node will set a XHE parameter between different values 1, 2 and 3 in an evenly distributed fashion, that is a node will choose the value equal to 1 with a probability equal to 0,333333. The same probability a node will choose the value equal to 2 and equal to 3. We adopted this probabilistic technique in order to avoid all the nodes that have a small number of data packets to transmit to set the same value. When adopting a probabilistic approach it is possible to “spread” the competing nodes. In practice, we tried to have smaller competition area when considering the probabilistic approach. Otherwise, when the number of data packets is higher than the half of the queue size, this means that a node need more data slots in a lesser time than other nodes. For this reason this node will set the XHE equal to 0. The *Max_Buffer_Size* is the maximum number of data packets a node is able to store. This means that if we set *Max_Buffer_Size* equal to 50 (this is the actual value of buffer size we used in simulations) and the buffer is full a node will drop next data packets. The parameter *queue_size* is the number of data packets stored at the current instant a node (*Node A*) has to send. We already said that this information is available at network layer. MAX_XHE is the maximum value of *XmtHoldoffExponent* that a node can acquire in our network. In [8] the authors argue that on choosing the MAX_XHE, 4 is large enough; otherwise, the connection setup latency will become too long. The threshold value of *Max_Buffer_Size/2* has been heuristically chosen. Further study will be conducted to optimize this choice. We set MAX_XHE equal to 3 because higher values introduce an excessive latency for a node and, if a node does not have data packets to send, the information about its condition has to be known from its neighborhood. In practice, if a node has to wait longer time before to acquire a new opportunity to transmit it is not able to send its schedule information to the neighborhood. This implies that its neighbor nodes will include him as competing nodes because they do not have update information.

4. Performance Evaluation

In this section, we provide ns-2 [7] simulation results for various scenarios. We deal simulations to investigate the performance of the CDS when our dynamic scheme for setting XHE is applied and the CDS scheme in which static values of XHE are chosen at beginning. It is worth to note that the choice of this parameter has been left un-standardized in the 802.16 and the main scope of this work is to show how different choices of XHE conduct to different behaviors of the network in terms of throughput and delay. Moreover, it is worth to note that we relate the setting of the XHE with data queue size because our feeling is that the latency of a node that needs to send more data packets than another one, should be smaller in order to avoid loss of data (overflow) or traffic congestion or at least reduce them. Note that our scheme does not introduce any starvation as nodes with smaller traffic to send are allowed to reserve data slots even if they are delayed compared to other nodes. In practice the inherent fairness policy of the standard is kept.

4.1 Simulation Network Model

The reference network architecture is a WiMAX mesh cloud interconnected to a Wide Area Network (WAN) through only one gateway node (Fig. 3). The Gateway node provide WAN connectivity to all other Mesh nodes. Practically, we suppose that all the data traffic is directed from a generic source node to the Gateway one. Nodes can act as traffic source node, traffic relay node or both. In other words, a Mesh node could be a direct traffic source, could have data packets to forward for other nodes in the network (Relay) or could be a direct traffic source while it is also forwarding traffic for other nodes. Nodes are randomly positioned in a square grid of 1000 meter x 1000 meter.

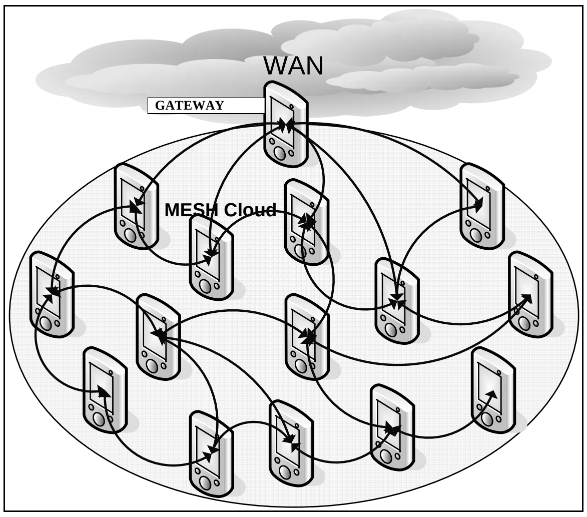

Figure 3. Network architecture.

The performance evaluation of our proposal was made utilizing the well-known simulation tool ns2. We first outline that the current MAC modules of ns-2 include 802.11, Ethernet, TDMA and satellite; however, no 802.16 MAC module is available. In our work, we implemented a new MAC module for the IEEE 802.16 mesh mode and use it to study the system performance. There is a logical component, the scheduling part, that handles the signaling channel contention and data allocation. During the holdoff time of a node, the MSH-DSCH messages received from PHY module are sent to the scheduling component. In the transmission slot, the scheduling component contends the next transmission time using the election algorithm defined in the standard based on the collected neighbor's information. The data channel component receives and transmits data packets in the allocated time slots. The XHE value determines the node transmission interval and holdoff time. We made simulations for both static and dynamic management of XHE parameter. In the static case, that is the case in which identical holdoff exponents have been assigned to every node in the network, we considered different values (0, 1 and 2) for the XHE value. Furthermore, we evaluated the CDS scheme with our probabilistic dynamic approach. In this work we are interested on evaluating the impact of the distributed scheduling scheme on the performance of the network. Specifically, we evaluate some significant parameters for wireless mesh networks:

1) *Throughput*: is the number of data packets correctly delivered to the destination over the number of data packets sent;
2) *Average end-to-end data packet delay*: the time that takes into account all the delay encountered in the network as buffering delay, transmission delay and propagation delay of data packets;
3) *Latency*: the average interval between an opportunity to transmit and the next opportunity to transmit for each node or, in other words, the average time interval that occurs between the transmission of a MSH-DSCH and the next transmission of a new MSH-DSCH (remember that the opportunity to transmit is used both for updating the neighborhood with the current schedule and for the reservation of data slots so the latency is an important parameter that can affect the performance of the network).

These three parameters have been chosen because the biggest challenge in building a wireless backhaul is to provide performance (throughput and delay) similar to those typically offered by a wired backhaul. Simulation Parameters are given in Table I. The considered network scenario consists of a variable number of mesh nodes (specifically we considered a number of nodes equal to 30, 40, 50, 60, 70 and 80 nodes) and one Gateway node. The Gateway interconnects the mesh network with other networks that could be potentially wired or wireless. We differentiate data rates for different sources in order to create scenarios in which data traffic is heterogeneous. Notice that a node can activate simultaneously more than one data source. We simulated two scenarios, respectively a *light* data traffic scenario and a *heavy* data traffic scenario.

In the *light* scenario we set 20 traffic sources: ten of these send data traffic at a rate of 5 pkts/sec, 5 sources that send at a rate of 50 pkts/sec and 5 sources that send data packets at a rate of 500 pkts/sec. The total number of data packets transmitted is 200000. In the *heavy* scenario we considered 40 traffic sources: 10 sources that send data traffic at a rate of 5 pkts/sec, 15 sources that send at a rate of 50 pkts/sec and 15 sources that send data packets at a rate of 500 pkts/sec. The total number of data packets is 400000.

TABLE I. SIMULATION PARAMETERS

Input Parameters	
Simulation area	1000x1000 meter
Traffic sources	CBR
Number of traffic sources	20 (light scenario), 40 (heavy scnenario)
Number of nodes	30, 40, 50, 60, 70, 80
Size of data packets	64 bytes
Transmission range	250 m
Simulation Time	500 s
Data Buffer Size	50
Light data traffic scenario	
Number of traffic sources	20
Sending rate	5, 50, 500 pkts/sec
10 sources	5 pkts/sec
5 sources	50 pkts/sec
5 sources	500 pkts/sec
Heavy data traffic scenario	
Number of Traffic sources	40
Sending rate	5, 50, 500 pkts/sec
10 sources	Rate 5 pkts/sec
15 sources	Rate 50 pkts/sec
15 sources	Rate 500 pkts/sec
Simulator	
Simulator	NS-2 (version 2.1b6a)
Medium Access Protocol	802.16 (CDS)
Link Bandwidth	1 Mbps
Confidence interval	95%
#run	10

We will show that even though in some cases the average value of the latency increases when our mechanism to set the XHE is used, we obtain better performance in terms of throughput and data delay.

4.2 Simulation Results

In this sub-section we show simulations results.
First of all, we analyze results concerning the scenario we called light data traffic scenario. Specifically we considered throughput and average end-to-end data packet delay varying the number of nodes in the network. Indeed, we considered scenarios with 30, 40, 50, 60, 70 and 80 nodes in the network. Concerning the light scenario we also analyzed the average latency, that is the average interval time to obtain an NXTO and the next.
After, we considered the "heavy" data traffic scenario in which 40 sources have been introduced during each simulation run. Once throughput and end-to-end delay have been presented, we consider a specific simulation scenario with 50 nodes and we draw for 1 and 2 hops neighbors of the gateway the average latency in a specific run and the total number of data packets a node managed during a simulation run. This representation is useful to understand how our approach works, that is the effect of applying our algorithm.
Finally, concerning this specific scenario with 50 nodes, we give an instantaneous "snapshot" representing the simulation scenario of nodes at 1, 2 and 3 hops away from the gateway. For each node we consider whether the node is at 1, 2 or 3 hops gateway's neighbor, the total number of data packets involved this node during the simulation run we are considering and the average latency. This snapshot should be more useful to understand internal mechanism of our dynamic algorithm.
In Figure 4 we can see the Throughput obtained when we considered static values of the XHE identical for each node and respectively equal to 0, 1 and 2. Higher values of XHE are not shown because they introduce an excessive latency and performance of the networks are worst in terms of throughput and delay. Furthermore, we evaluated our probabilistic dynamic approach called ProbApproach in each plot. It is worth to remember that our approach is based on the data queue size. In practice, the contention window of each node is modified based on the data traffic a node has to manage in a certain time. As we can see in Figure 4 a dynamic setting of the XHE allows to overcome in terms of throughput 802.16-XHE-0, 802.16-XHE-1 and 802.16XHE-2.

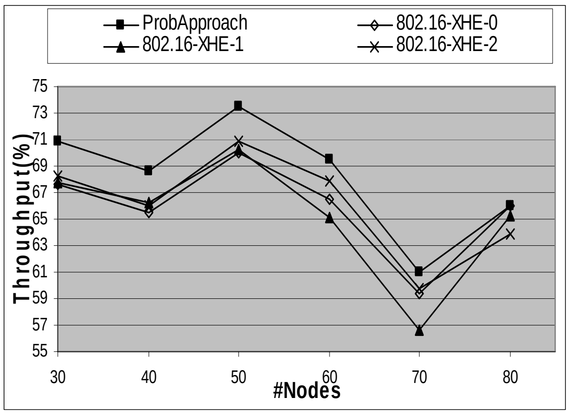

Figure 4. Throughput (light scenario).

This result is not surprisingly because we choose to modify the contention window of a node based on the need of each node to transmit data packets. It is worth to note that our approach is not optimal, in the sense that the choice of the data queue size fixed equal to the half of the maximum size of data buffer has been heuristically found. However, we believe that results obtained are encouraging and permit to trace some ideas to set the parameters in the distributed scheduling scheme in a dynamic fashion.

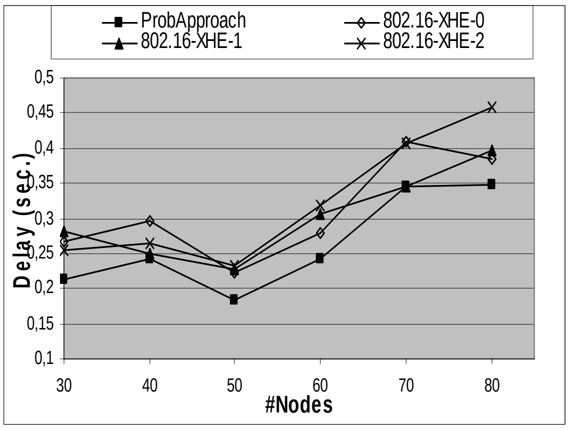

Figure 5. Average end-to-end data packet delay (light scenario).

On the other hand, good results have been obtained also when the average end-to-end data packet delay has been considered as shown in Figure 5. It is worth to note that the average end-to-end data packet delay of our mechanism outperforms results of the others three schemes even though the average latency (Fig. 6) of our scheme is higher than the latency of the 802.16-XHE-0 and 802.16-XHE-1. In Figure 6 the average latency of the light scenario is shown. Of course, we have to take into account that we estimated the average latency. This means that sources that have to send less data packets will be characterized with higher latency and sources that have to send more data packets will be associated with smaller latency values.

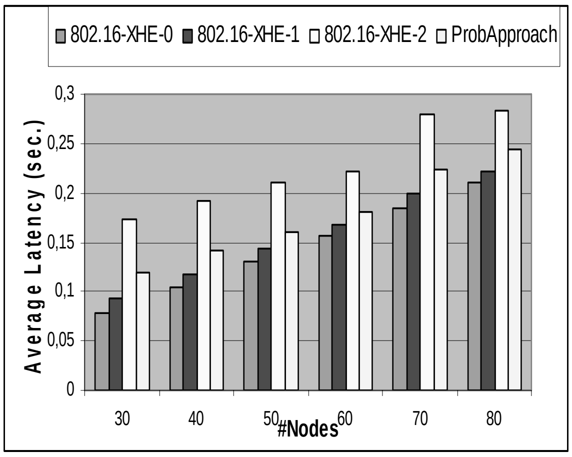

Figure 6. Average latency (light scenario).

We can observe that when the XHE increases the average latency increases too. As we can observe in Figures 4 and 5 the increasing of the latency does not necessarily correspond to worst behavior in terms of throughput or end-to-end delay. Our approach permits a more loaded node to contend in a more "aggressive" way than a node that has a smaller number of data packets to send, but this latter does not starve, in the sense that it is only delayed to reserve data slots.

In order to confirm the effectiveness of the probabilistic dynamic algorithm proposed results for the heavy data traffic scenario are shown in figures 7 and 8. This second scenario has been considered as the average latency should increase when an higher number of data packets involves nodes.

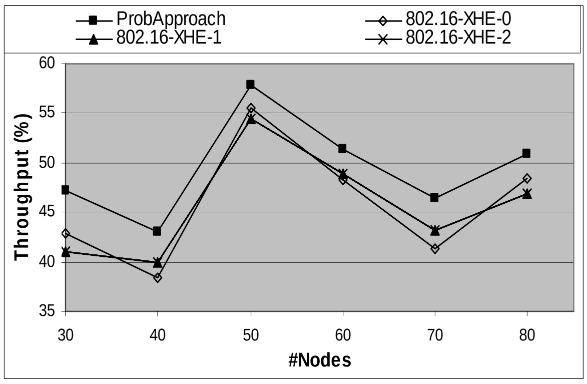

Figure 7. Throughput (heavy data traffic)

The "strange" slope obtained in Figure 4 and 7 when passing from 40 nodes to 50 nodes is due to the fact that even if the number of competing nodes increases, the possibility to find different paths and to use different nodes to send data traffic increases too.

For this reason, when considering 50 nodes, performance in terms of throughput and end-to-end data packet delay are better than at 40 nodes. On the other hand, when the number of nodes increases the contention between nodes increases too, but when we have 50 nodes in the network the average number of neighbors for each node is less or equal to the number of control slots (opportunities to transmit, that are fixed equal to 16 for each frame in the standard). Generally, when the density of the network increases the average number of neighbors for each node increases too. Hence, even though the possibility to find a path with a smaller number of hops increases in this case, the possibility to catch an opportunity to transmit decreases. In Table II we show the different number of neighbors when different scenarios of networks are considered. Specifically, we computed the number of neighbors N' (1 and 2 hop) analytically and through the simulation scenarios created in ns2. Concerning the analytical scheme let N1 the number of 1-hop neighbors (it is the number of nodes in the coverage area of a Mesh Router). N1 can be expressed as $N1 = \rho \pi r^2$ where ρ is the network density and can be expressed as $\rho = \frac{N}{\pi R^2}$. N' is the number of 1 and 2-hop neighbors and can be computed as $N' = 4N\frac{r^2}{R^2}$, in which r is the radius of each node (in our simulation we supposed all the nodes have the same radius or transmission range equal to 250 meter), R is the range of the network and N is the number of nodes in the networks.

TABLE II. NUMBER OF NEIGHBORS (1AND 2 HOP) CONSIDERING DIFFERENT SCENARIOS

# Nodes in the Network	N' (Analytical)	N' (Simulation)
30	7,5	7,6
40	10	11,3
50	12,5	17
60	15	20
70	17,5	22
80	20	24

We can observe that when the number of nodes is smaller or equal to 50 the number of control slots (opportunities to transmit) is, in average, sufficient for each node in each frame. When the number of nodes in the network increases the average number of neighbors (1 and 2 hop neighbors) increases too and number of control slots is not sufficient to cover each node in each frame.

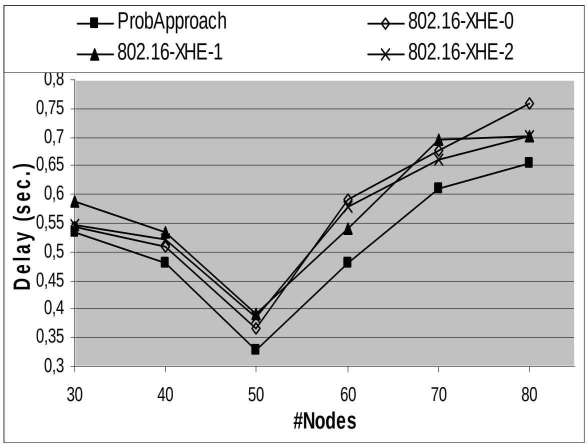

Figure 8. Average end-to-end data delay (heavy data traffic).

For this reason when we consider performance in terms of throughput and end-to-end delay and the number of nodes is higher than 60 the behavior is worst than in the case of 50 nodes or a smaller number of nodes. On the other hand this behavior is confirmed when the average end-to-end data packet delay is considered in Figure 6 and 8.

In Table III and Table IV we reported the latency and the total number of data packets that a node managed during a simulation run at 1 and 2 hops respectively. We considered a simulation scenario with 50 nodes when heavy traffic is considered. In practice results reported in the tables refer to the same scenario considered to obtain results in Figures 7 and 8. Due to the lack of space we only reported results of nodes at 1 and 2 hops away from the gateway but they are sufficient in order our algorithm to be explained.

Generally, we can observe as the latency decreases while the number of data packets increases. This is a side effect of our algorithm, because the XHE parameter is set to smaller values when a node needs to send more data packets and vice-versa.

TABLE III. 1-HOP NEIGHBORS (NETWORK WITH 50 NODES ANDHEAVY DATA TRAFFIC)

Node	Latency (Prob-App)	#Data Packets (Prob-App)	Latency (802-16-XHE-1)	#Data Packet s (802-16-XHE-1)
0	0,226046	10000	0,14505	10000
3	0,209472	91185	0,14647	86982
10	0,202951	29688	0,14442	83576
12	0,238848	50999	0,14345	43508
21	0,241095	10000	0,13595	10000
23	0,234214	0	0,13498	0
25	0,224778	20000	0,13687	20000
37	0,195972	51307	0,15136	1530
44	0,229106	13046	0,15243	17488
45	0,248828	10000	0,15235	10000
47	0,250282	20000	0,15128	20000
	Avg Latency	Tot Data Pakts	Avg Latency	Tot Data Pakts
	0,22557	306225	0,14496	303084

Of course, the fact we assign smaller values to XHE parameter does not mean that the latency always decreases. Indeed, smaller values of XHE correspond with smaller contention window, but a node needs to contend with its neighborhood and we computed the average latency taking into account the instant time a node reserves a new opportunity to transmit (control slot).

TABLE IV. 2-HOP-NEIGHBORS (NETWORK WITH 50 NODES AND HEAVY DATA TRAFFIC)

Node	Latency (Prob-App)	#Data Packets (Prob-App)	Latency (802-16-XHE-1)	#Data Packets (802-16-XHE-1)
4	0,23271	10000	0,14519	27931
14	0,23750	31076	0,14345	10000
15	0,23901	10000	0,14324	10000
20	0,23597	10000	0,13614	40331
22	0,23196	10000	0,13516	30616
24	0,22564	57919	0,13712	36270
28	0,22306	10000	0,13514	10000
29	0,24620	13063	0,13502	7488
30	0,20180	76461	0,13582	51879
43	0,24987	0	0,15136	0
	Avg Latency	Tot Data Pkts	Avg Latency	Tot Data Pkts
	0,23233	228529	0,13916	224515

Moreover, latency values reported in the table are averaged on the duration of a run, that is the latency is an average latency and so it takes into account smaller values of XHE and higher values of XHE based on the specific time the XHE value is considered. Let consider the following example: imagine a node A has to send a lot of data packets in a certain time, so its XHE will be a small value. On the other hand it will happen that this node A in a certain time does not have to send a large number of data packets, so in this instant time if we considere the XHE, we obtain an higher XHE value and the latency will increase. So, it can happen that even if a node managed more data packets than another node its average latency is slightly higher or close to the average value of another node.

In order to verify this behavior we can consider nodes 4 and 14 in Table IV. Nodes 14 manages more data packets than node 4 but its average latency is slightly higher than those of node 4. Generally, the effect of applying our algorithm is that more data packets correspond with smaller latency.

This result is due to the dynamic mechanism introduced allowing to change in a dynamic fashion the probability to reserve data slots and so to transmit data packets. This means that a more loaded node (that is, a node that has the necessity to send more data packets) will contend in a more aggressive way in respect of a node that has a smaller number of data packets to send.

In Figure 9, we give an instantaneous "snapshot" of the network when 50 nodes are considered and the scenario is those referred as "heavy" data traffic scenario. Due to the lack of space, we did not report all the nodes in the networks, but we plotted only some nodes. Specifically, we showed nodes at 3 hops away from the gateway (node 1) and for each node the total number of data packets that this node managed during a simulation run and the average latency have been shown. We can observe as the greater is the number of packets a node managed during a simulation run the lower is the average latency related with this node. Indeed, observe at the level of 1-hop node 37 and node 21.

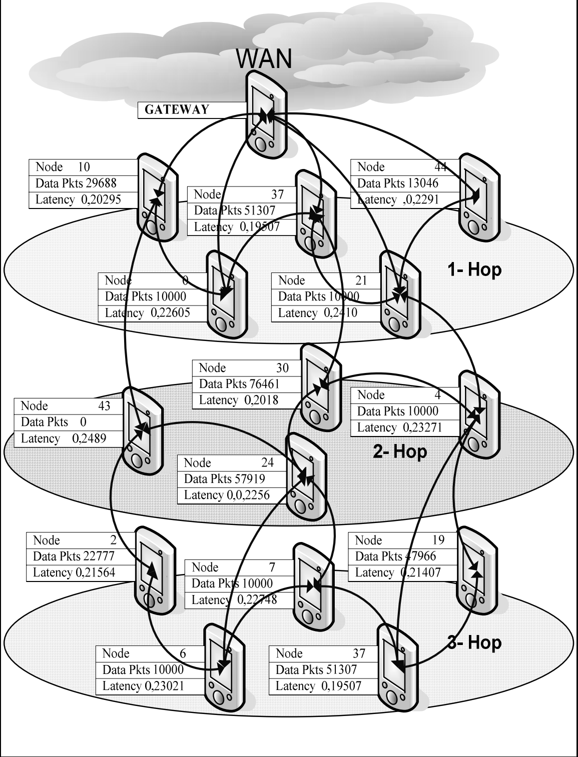

Figure 9. Actual example of simulation results in which a network of 50 nodes in a 1000 meter x 1000 meter has been considered with heavy traffic.

The first one manages 51307 data packets (a node can manage some data packet as relay node for another node, for example 37 could be charged to send data packet for node 30 and it can send their own packets) and the average latency is 0,19507 sec. On the other hand the second one, node 21, manages 10000 data packets and its average latency is 0,2410 sec. This behavior is an effect of the application of our dynamic algorithm. In fact, our algorithm will set more times smaller values of parameter XHE in the schedule scheme for node 37 in respect of node 21. It is worth to observe that the average latency is not linearly related with the speed to send data packets. In fact, we already outlined that the scheduling frame is characterized with two kinds of frames, a control frame where we consider the latency expressed as the time occurring between a control slot reservation and the next control slot reservation. The other part of the frame is characterized with the data frame, in which the reservation of data slots takes place. In conclusion we can affirm that the effect of our algorithm is that the average latency of each node during a simulation run decrease when the total number of data packets the node has to manage increases. A positive effect of the application of this algorithm is the increasing of the data packets delivered to the destination and the time required to deliver these packets decreases. In practice, we show a simple way to set an un-standardized parameter.

5. Conclusions

In this paper we analyze the performance of the Coordinated Distributed Scheduler of the Std. IEEE 802.16. In order to do that we implemented this scheduler in a well-known network simulation tool, ns2. We had to implement a MAC module for the 802.16 because nowadays there is not an available MAC module for IEEE 802.16. Some results obtained confirm results known in literature. In fact, the latency increases when the number of nodes in the network increases too. In order to obtain a higher scalability we developed a simple dynamic and probabilistic queue-size based algorithm that in a dynamic way permits to set the *XmtHoldoffExponent* (XHE).

The setting of XHE is a critical point of the standard scheduler because it determines the contention window of each node. Furthermore, the contention window of each node in a distributed environment depends of the neighborhood. In this work we proposed a mechanism to set in a dynamic and probabilistic fashion XHE parameter. Of course this work is not at all exhaustive as far as the setting of parameter is concerned. Indeed, the contention window of each node strictly depends of the number of neighbors. As further work we would like to improve the parameters tuning taking into account other aspects like the number of neighbors or the presence of expiring packets into buffers.

10. References

[1] "802.16 IEEE Standard for Local and metropolitan are networks" IEEE Std 802.16-2004.

[2] IEEE 802.16-2005, " IEEE Standard for Local and Metropolitan Area Networks – Part 16: Air Interface for Fixed Broadband Wireless Access Systems for Mobile Users," Dec. 2005.

[3] 3WiMAX Forum, http://www.wimaxforum.org

[4] C. Cicconetti, L. Lenzini, E. Mingozzi, and C. Eklund, "Quality of Service Support in IEEE 802.16 networks ," IEEE Network, Vol. 20, Issue 2, pp. 50 – 55, 2006.

[5] Guosong Chu, Deng Wang, and Shunliang Mei. "A QoS architecture for the MAC protocol of IEEE 802.16 BWA System". IEEE *International Conference on Communications Circuits & System and West Sino Expositions*, vol.1, pp. 435-439, China 2002..

[6] Arunabna Ghosh, David R, Wolter, Jeffrey G. Andrews and Runhua Chen, " Broadband Wireless Access with WiMax/802.16: Current Performance Benchmarks and Future Potential" *IEEE Communications Magazine*, February 2005, p129-136.

[7] " http://www.isi.edu/nsmam/ns/ ".

[8] Min Cao, Wenchao Ma, Qian Zhang, Xiaodong Wang, and Wenwu Zhu. "Modeling and performance analysis of the distributed scheduler in IEEE 802.16 mesh mode". In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing,* pages 78-89, New York, NY, USA,2005, ACM Press.