GCAD: A Novel Call Admission Control Algorithm in IEEE 802.16 based Wireless Mesh Networks

Floriano De Rango, Andrea Malfitano, Salvatore Marano
D.E.I.S. Department, University of Calabria, Italy
Email:{derango, amalfitano, marano}@deis.unical.it

Abstract—In this paper, a GCAD-CAC (Greedy Choice with Bandwidth Availability aware Defragmentation) algorithm is proposed. It is able to guarantee respect for data flow delay constraints defined by three different traffic classes. To achieve good results, the algorithm tries to accept all the new requests, but when a higher priority request is received, a lower priority admitted request is preempted. This preemption can leave some small gaps which are not sufficient for new connection admission; these gaps can be collected by the GCAD algorithm by activating a bandwidth availability based defragmentation process. The quality of the algorithm is shown by a comparison with two other algorithms found in the literature.

Index Terms—Call admission control, mesh networks, WiMAX, IEEE 802.16.

I. INTRODUCTION

In distributed mesh mode, defined by IEEE 802.16 protocol [1], when a mesh node has an amount of new data to transfer to a destination node, it requires the instauration of a new connection in the neighbour node. This last node has to decide whether to admit the new call, and obviously, how much bandwidth to allocate to the new connection, for the service lifetime. The first is the admission decision, the second one involves the bandwidth to grant to the node for the admitted connection.

Both the decisions are inherent in bandwidth utilization in the network and influence the desired QoS (Quality of Service) level: the arrival of a new connection, can modify the allowed bandwidth to the existing connections, thus, all the QoS constraints must be reviewed. Therefore there is a “risk” in this choice, because, by admitting a new connection, we risk a deterioration in the QoS provided to old connections. The first of the previous listed process decisions is called call admission control, and this decision influences the network bandwidth utilization for a long time, i.e. it is a long term decision. The second one, instead, is a short term decision and defines the amount of bandwidth to grant to the requesting node.

In this paper we present a new CAC algorithm, referring to a mesh scenario which takes into account a set of three traffic classes with different priority levels. The focus is guaranteeing the respect of QoS constraints defined in term of end-to-end delay to higher priority flows. To do this the distributed CAC algorithm admits all the new calls in a greedy way, thus the lower priority flows can exploit the bandwidth availability until a set of higher priority calls claim bandwidth. The lower priority admitted calls are thus preempted to leave room for the new calls. The preemption process has a negative side, it can leave little gaps in the data subframe which are not useful. The solution is the presence of a defragmentation process started by the granter node.

In the rest of this paper our algorithm GCAD-CAC (Greedy Choice with Bandwidth Availability aware Defragmentation) and an evaluation of its performance is presented in detail.

II. IEEE 802.16 MESH MODE

The IEEE 802.16 protocol supports two operating modes: Point-to-Multipoint (PMP) and mesh mode. In the first mode, the base station (BS) has the most important role in coordinating the transmission, i.e. in downlink, the BS is the only station capable of broadcasting data and control messages. On the other hand in uplink, the subscriber stations (SSs) have to use the bandwidth as defined by BS in a centralized way. In mesh mode the BS loses the central role of coordinator, and the coordination can be made in a distributed manner, BS being distinguishable among mesh nodes, because it is the only gateway to reach the rest of the world. In this way, neighbour (a neighbour of a node is a node one step away from it), neighbourhood (the set of all neighbouring nodes) and extended neighbourhood (containing, in addition to the neighbourhood node, the neighbours of neighbours) are new terms unknown to PMP mode. Given these definitions, we can enunciate the basic principle of transmission coordination in an 802.16 mesh network: no node can transmit on its own initiative, including the BS node, without coordinating its transmission within its extended neighbourhood.

A. Distributed Coordinated Scheduling

The mesh mode supports only TDD (Time Division Duplexing) mode; the frame is divided into two parts: the first one is the control subframe and the second one is the data subframe. The data subframe is divided into a fixed
number of minislots, the control subframe can be of two different types:
- network control subframe: used to transport network control messages such as MSH-NENT and MSH-NCFG, used by nodes to acquire network synchronization and network configuration properties.
- scheduling control subframe: used to collect bandwidth requests and to send grant messages. MSH-CSCF and MSH-CSCH are the messages related to the centralized scheduling mode, while MSH-DSCH is used in distributed mode.

The network control subframe is not present in each frame and it appears periodically. The Scheduling Frame parameter, broadcasted in the MSH-NCFG message, defines how many frames have a scheduling control subframe between two frames with network control subframes.

The dispatch of the previous messages (except MSH-NENT) occurs in a collision free manner and to guarantee it, each node inserts two fields in the control messages:
- \( xmt\text{ holdoff exponent} \)
- \( next\ xmt\ mx. \)

Each node, during message forwarding (except MSH-NENT), calculates its next transmission instant and expresses it in the form of an interval using the two mentioned fields. In practice, the node does not inform the neighbours about its next transmission instant, but it sends an interval in which the next transmission falls in; this interval is defined by the following constraints:

\[
next\ xmt\ time > 2^{xmt\ text{ holdoff exponent}} \times next\ xmt\ mx \quad (1)
\]

\[
next\ xmt\ time \leq 2^{xmt\ text{ holdoff exponent}} \times (next\ xmt\ mx + 1) \quad (2)
\]

Between one transmission and the next, a node must wait for a time interval equal to:

\[
xmt\ text{ holdoff time} = 2^{(xmt\ text{ holdoff exponent} + 4)} \quad (3)
\]

Scheduling information will be issued in a request - grant format. While respecting the constraints of coordinated distributed scheduling, uncoordinated distributed scheduling can ensure fast setup communications. Uncoordinated scheduling is determined by direct requests and grants between two nodes and it must also take place in a way so as not to cause collisions with the messages and the coordinated scheduling traffic. Both modes of distributed scheduling, coordinated or not, use a three-way-handshake protocol. A mesh node, that has data to transmit, sends a request in an MSH-DSCH message to a destination mesh node, indicating the requested number of minislots it needs. The destination node replies with a grant message which is acknowledged with a grant copy by the requester node.

**B. QoS in Mesh Mode**

The only claims made by the protocol for QoS issues, state that the quality of service must be guaranteed packet by packet, in the link context. It must be the node, within the constraints of the distributed bandwidth allocation algorithm, to ensure compliance with the constraints of the individual application quality. Thus, to realize and satisfy QoS constraints, the protocol defines specific fields within the PDU header. The generic header of a MAC PDU, contains a 16-bit CID field. In mesh mode, the CID field is split into two parts, the first portion of 8 bits is the logical network identifier and the second portion contains the link identifier.

**C. Call Admission Control in WiMAX mesh networks**

IEEE 802.16 technology is still an open question. There are few works that fill the gaps in the protocol. This is true for the mesh mode and even more for algorithms related to distributed mode. The same call admission topic in mesh mode is neglected by literature. Instead, some research into distributed scheduler performance is available, in particular [2] analyzes distributed scheduler performances and illustrates how to set the \( xmt\ text{ holdoff exponent} \) parameters dynamically. The optimization of mesh scheduling is described in [3] evaluating a combined centralized - distributed scheduling. The authors of [4] only propose an improvement in the distributed mesh scheduler. There is a proposal for a call admission control algorithm in distributed mode in [5]; in this last paper the concept of connection preemption with some limitations is presented; three traffic classes with assigned priorities are considered, the admission algorithm based on the concept that all the bandwidth can be divided among the three classes, but in this way in a steady state, the advent of new data flows with higher priority are refused because these classes have consumed the bandwidth reserved for it; instead, new data flows with lower priority can be admitted. Also the scenario used to validate the proposed CAC algorithm is very simple, the maximum path length in scenario is two hops.

In [6] the authors propose an end-to-end bandwidth reservation scheme with a CAC algorithm which only refers to VoIP traffic. In [7] a simple CAC algorithm is proposed. The authors consider a traffic differentiation using the priority field of unicast CID. The CAC algorithm is based on a threshold mechanism. Requests with higher priority, if there are sufficient free minislots, are always admitted, whereas the low priority requests are refused in cases of congestion, which is verified with a bandwidth utilization computation. If bandwidth utilization is greater than a fixed threshold then low priority requests are refused. Also, the simulated scenario is too simple, each node is a neighbor of each other node in the network. The paper [8] describes a CAC algorithm related to PMP mode; it is very interesting for the connection preemption concept that is introduced and the admission decision is based on traffic class and bandwidth utilization of each traffic class. Each traffic class has a bandwidth portion reserved to it and can also preempt the lower priority admitted calls. Other interesting works focusing on call admission control in PMP mode are [9] - [13] and in particular, although it considers the PMP mode, the work [14] is to be taken into account to enrich our knowledge, in fact, the authors
of [14] apply the Games Theory ([15], [16]) to the call admission issue. The contribution of our work, can be considered important in the context of research into 802.16 mesh distributed architecture, because, to the best of our knowledge, in literature there are few works about CAC in mesh distributed mode. Our intention is to present a distributed call admission control algorithm which takes into account three different traffic classes. The proposed GCAD (Greedy Choice with bandwidth aware Availabilities Defragmentation) algorithm has two interesting processes: preemption and defragmentation. Preemption occurs when there is a new call with higher priority, which can preempt a call with lower priority. The preemption process can cause a fragmentation in the data subframe, i.e. we can find, in the data subframe, some small unusable gaps of free minislots. The defragmentation process collects these gaps creating continuous availability.

IV. GCAD: A NEW CALL ADMISSION CONTROL ALGORITHM

We propose a CAC algorithm for an IEEE 802.16 distributed mesh network, each mesh node can support three different data traffic classes with “1”, “2” and “3” as priority values. The values “1” and “3” are the highest and the lowest priority values respectively. When a new source starts to transmit data, it has to identify a path to send data to the destination node. Subsequently, the mesh node can submit a bandwidth request to the next hop node. In this section, we describe our proposal for the source and next hop node behavior. The first one is described in terms of bandwidth estimation, or more correctly minislot number estimation, and the last one in terms of the call admission control process. In the following we indicate the source node as requester and the next hop node as granter. Obviously the next hop node in turn becomes a requester and so on.

A. Minislot Number Request Estimation

Each node has a data queue and when a packet appears in the queue, the node creates a bandwidth request. The node can classify the queued packets using the priority field present in the unicast CID. The three traffic classes can have QoS constraints expressed in terms of end-to-end delay, thus, the node has to estimate the amount of minislot requests. Each data subframe is divided into a fixed number of 256 minislots. We define the MSNEA component; MSNEA is a Mini Slot Number Estimation Algorithm and its challenge is to determine the amount of bandwidth which a node needs. When a node, or for greater accuracy, the algorithm or agent designed with the task of observing the data queue, realizes that the data queue is not empty, it is necessary to request bandwidth to send the data packets. In particular in the IEEE 802.16 mesh scenario the frame is divided into two parts:

- control subframe
- data subframe.

The data subframe is the only part of the frame used to transmit data packets and it is divided into a well defined number of minislots. Consequently, when a node has to request bandwidth, the number of minislots needed has to be evaluated. The evaluation of this quantity may seem simple but it is very important for two reasons:

- if the estimated number of minislots is smaller than the number which the node actually needs then it becomes difficult to ensure that there is not an accumulation of data packets in the queue and it is also very hard to try to guarantee respect for quality constraints;
- if the estimated number of minislots is greater than the number which the node actually needs then there is a waste of bandwidth.

The previous concepts illustrate the importance of the presence of an efficient estimation algorithm. The MSNEA is described by the flow chart in figure 1. In addition to the flow chart it is necessary to provide a set of “conditions” which are used in it. It is also necessary to define the following parameters:

- \( MS \): OFDM symbol number for each minislot;
- \( p_{size} \): packet size (bits);
- \( eff \): efficiency of an OFDM symbol, expressed as number of data bits for each symbol;
- \( dl \): delay constraint;
- \( d_{sym} \): OFDM symbol duration (s);
- \( f \): frame duration (s);
- \( h \): path to destination hop count;
- \( t_i \): arrival time of the first queued packet of BE traffic;
- \( t_j \): arrival time of the last queued packet of BE traffic;
- \( n_{BE} \): number of BE queued packets;
- \( p_{mean} \): mean packet size of BE queued packets;
- \( R \): estimated BE rate;

And with these parameters we can estimate the \( nms \) request (number of minislots) for traffic with priority equal to 3 using the equations:

\[
R = \frac{p_{mean} \times (n_{BE} - 1)}{t_f - t_i} \tag{4}
\]

\[
nms = R \times \frac{f}{MS \times eff} \tag{5}
\]

and the \( nms \) request for traffic with priority equal to 1 or 2 resolving the following:

\[
(nms \times MS \times d_{sym}) + \left(\frac{p_{size} \times (n_1 + MS \times eff)}{n_{BE} \times MS \times eff} \right) \times f = \frac{dl}{h} \tag{6}
\]

Now our intention is to explain the behavior of MSNEA following the flow chart depicted in figure 1 and also using equations (4) and (6) and the new equation:

\[
\frac{dl}{h} - (t_{now} - t_{last}) \leq \frac{Total_{byte} + \frac{f}{nms \times MS \times eff}}{nms \times MS \times eff} \tag{7}
\]

which is defined using these parameters:

- \( t_{now} \): time instant in which the calculation takes place;
- \( t_{\text{last}} \): time instant corresponding to the arrival of the last queued packet;
- \( \text{Total\_byte} \): total bytes present in queue and referring to the same traffic class.

The first term of equation (7) is indicated in the flow chart as \( t_{\text{need}} \) and the second term as \( t_{\text{actual}} \). Also this equation allows us to calculate the \( nms \) request for traffic with priority equal to 1 or 2. The use of this equation and of the other will be explained below. In the flow chart we indicate equation (5) with the term condition (1); equation (6) as condition (2) and equation (7) as condition (3).

The MSNEA is invoked by the node at the instant in which the node has a MSH-DSCH to send, in this way MSNEA can evaluate the possibility of making a new bandwidth request for an existing data flow or for a new data flow. The first step made by algorithm is the extraction of the first PDU from the data queue, we indicate this PDU as PDU1. The PDU1 belongs to a traffic class with a priority indicated as PDU1.priority; MSNEA verifies the presence of an existing pending request for bandwidth for data flow with this priority. If a pending request exists then the MSNEA searches for the presence of other PDU in the queue with a different priority value. If the MSNEA finds this PDU then it also verifies for this priority the presence of pending requests. If the algorithm finds a pending request then it repeats the process for the last priority value. When the MSNEA does not find a pending request, it scans the data queue searching for all PDUs with priority equal to PDU1.priority. During the scan, the algorithm evaluates a set of parameters and in particular:

- \( \text{Start\_time} \): the arrival time instant of the first queued PDU with priority equal to PDU1.priority.
- \( \text{End\_time} \): the arrival time instant of the last queued PDU with priority equal to PDU1.priority.
- \( \text{Total\_byte} \): the total number of bytes related to all the PDUs queued with priority equal to PDU1.priority.
- \( \text{Card\_pack} \): is the number of queued PDUs with priority equal to PDU1.priority;
- \( \text{Interval} \): is defined as

\[
\text{Interval} = \text{End\_time} - \text{Start\_time}. \tag{8}
\]

It represents the time interval which elapses between the two arrival time instants of the first and last queued PDUs with priority equal to PDU1.priority.

- \( \text{PS\_mean} \): is defined as

\[
\text{PS\_mean} = \frac{\text{Total\_byte}}{\text{Card\_pack}}; \tag{9}
\]

represents an estimation of the mean packet size.

- \( \text{Dt\_mean} \): is defined as

\[
\text{Dt\_mean} = \frac{\text{Interval}}{\text{Card\_pack}}; \tag{10}
\]

represents an estimation of the time rate of queued PDU with priority equal to PDU1.priority.

At this point, if the MSNEA does not find already active grants for data flow with priority equal to PDU1.priority, it has to make the first request and it has to estimate the number of minislots \( (nms) \) on the basis of PDU1.priority. If the PDU1.priority is equal to 3, then the estimation is made by condition (1) i.e. by equation (5), instead if PDU1.priority is equal to 1 or 2, then the estimation uses condition (2), i.e. equation (6). Otherwise, if the node has an active grant for the same priority the MSNEA has to evaluate a set of conditions to decide if it is necessary to make a new request. To support the decision, the algorithm evaluates the state of three Boolean variables: \( \text{First\_alarm} \), \( \text{Second\_alarm} \) and \( \text{Constraint\_alarm} \). To assign a value to these variables the algorithm elaborates the following parameters:

- \( \text{temp\_nms} \): a first estimation of the number of slots that the node needs to deliver the queued PDU with priority equal to PDU1.priority is made by condition (1);
- \( \text{Total\_ms} \): the number of minislots that the node already has;
- \( \text{t\_need} \): represents the time slots that the node needs to deliver the last queued PDU to its destination, with priority equal to PDU1.priority, respecting the delay constraint;
- \( \text{t\_actual} \): the node, using the minislots previously granted to it for the PDUs with priority equal to PDU1.priority, has an available time interval to deliver the queued PDUs, this time interval is actual; \( \text{t\_need} \) and \( \text{t\_actual} \) are respectively the first and the second terms of equation (7);
- \( \text{Sampled\_queue} \): to verify if there is the need to request more minislots, the MSNEA samples the data queue length, the sampling takes place when the node receives a new grant; this parameter indicates the last sampled value.

The first variable that the MSNEA considers is the \( \text{First\_alarm} \), this variable is set with “true” if the \( \text{Total\_ms} \) is smaller than the \( \text{temp\_ms} \) and this means that the number of minislots owned by the node is not sufficient to deliver all the queued PDU with priority equal to PDU1.priority. This condition represents a first alarm for the MSNEA. The second evaluated variable is the \( \text{Constraint\_alarm} \), it is considered only for PDU with priority equal to 1 or 2 and is set to true if the minislots previously granted to the node are not sufficient to guarantee compliance with the delay constraint.
The last variable is *Second_alarm* and it is set to true if in comparison with the actual value of the queue length it is greater than the *sampled_queue* parameter; this means that the previously granted minislots are not sufficient to guarantee the disposal of queued PDUs, i.e. if there is not a new grant then there will be a continuous accumulation of PDUs in the queue.

The last verification is useful to understand if it is necessary to make a new estimation for a new request. For MSNEA, if *First_alarm* and *Second_alarm* are both
true or the Constraint_alarm is true then it is necessary to make a new request and the nms is evaluated using condition (2) or condition (1) on the basis of PDU1.priority. It is interesting to note that if First_alarm and Second_alarm are both false then if Constraint_alarm is true then MSNEA effects a new request, this means that the number of minislots previously granted to the node are sufficient to dispose of the queued PDUs but not sufficient to guarantee the compliance with the delay constraint. The Constraint_alarm is the most important condition in deciding for a new request. It is necessary to clarify that for PDUs with priority equal to 1 or 2 the minislot number estimation is made using condition (2) and not condition (3). As can be seen condition (3) is only used to set the alarm variable, because the estimation made by condition (2) is smaller than the value obtained by condition (3) and thus we obtain a conservative estimation. By this we mean that it is better to make a new estimation that will probably be insufficient and not make a request that leads to a waste of bandwidth; to remedy insufficient bandwidth there is the possibility of subsequently retesting the needs of a new request.

B. Call Admission Control Algorithm

The proposed GCAD-CAC algorithm is described by the flow diagram depicted in figure 2. The parameters expressed in the flow diagram are defined as the following:
- $B_A$: minislot number available at arrival instant of a new request;
- $B_P$: minislot number collected by preemption;
- $B_1A$: total minislot number obtained after a preemption to admit a new request with priority equal to “1”;
- $B_2A$: total minislot number obtained after a preemption to admit a new request with priority equal to “2”;
- $B_D$: total minislot number which can be obtained by defragmentation process.

When a mesh node receives a new request, expressed as a number of requested minislots: $R_n$, it admits all kinds of requests if there is sufficient available bandwidth. This explains why the algorithm is defined as greedy; many CAC algorithms define utilization constraints and refuse a new connection if the traffic class has achieved the utilization threshold. We instead, try to take advantage of the actual minislots available, also, trying to respect all QoS delay constraints.

If a new request arrives with higher priority than a previously admitted one and there are not sufficient available minislots, then admitted calls with lower priority can be preempted. Thus, before preempting a connection, the granter evaluates the amount of minislots obtainable by preemption: $B_P$.

If the total available minislots ($B_1A$ or $B_2A$ for requests with priority equal to “1” and “2” respectively) is greater or equal to $R_n$, then the preemption of a previous admitted request $c_i$ with:

$$\text{Priority}(c_i) < \text{Priority}(R_n)$$

then:

Figure 2. Proposed algorithm for call admission control.

An important condition to be considered is the following: $B_1A$ and $B_2A$ are evaluated considering only contiguous minislots.

For example, considering the allocation scheme depicted in figure 3, a new request, with priority equal to “1”, can preempt the “e” and not “d” allocation because only “e” is contiguous with the available minislots. After preemption the new request is admitted. The granter, in this case, advises connection “e”, that is preempted. In order to advise the preempted connection, we send a grant message with minislot range field equal to “0” to
the owner node and the preempted connection can make a new request to try to obtain free minislots.

Figure 3. Data subframe with minislot allocations.

After the preemption test, if a new request with priority equal to “1” or “2”, does not have sufficient available minislots, the granter node can activate defragmentation to collect fractioned available minislots in a whole availability.

Figure 4. Data subframe states: (a) before preemption; (b) after preemption and finally (c) after defragmentation process.

In figure 4, it is possible to note the case in which there is an advantage in defragmentation utilization. The rectangle without numbers represents free minislots. Case (a) represents the data subframe before a preemption due to the arrival of a new request with priority equal to “2”; the data subframe state after preemption is represented in case (b), the preemption causes the presence of a free minislot gap between two allocations; with defragmentation two gaps are unified and a new request can be admitted as in case (c). To effect defragmentation, the granter sends a grant message with a range equal to “0” to all the interested nodes. In this way the granter node advises the defragmented connection owners to bargain for the new grants. The granter, obviously, in the “bargaining” process, allocates minislots in a contiguous way, and admits a new request, in advanced minislots, only after re-allocation of the defragmented connections.

V. PERFORMANCE EVALUATION

To test the proposed algorithm, we designed a network simulator for IEEE 802.16-2004 [1] protocol in JAVA language. In the simulator we implemented our algorithm and also other two algorithms to make a performance comparison. In figure 5 the mesh simulated scenario is depicted. We consider a mesh network with 25 nodes, one of these (number 1) is the BS. The depicted lines represent the active links. The scenario is a square with area: 5 km x 5 km. All the traffic is from mesh nodes to BS node.

A. Simulation Scenario

Table I summarizes all the simulation settings. In each simulation, the source nodes and the packet generation start instants are randomly selected.

Figure 5. Simulated scenario.

The presented algorithm is compared with two other algorithms identified in literature. The first is extracted from paper [7] and in the following is indicated as THR algorithm (THR because it is based on a threshold mechanism). The second algorithm is a CAC algorithm for a 802.16 PMP scenario and is proposed in [8]. It is very promising and it has been adapted to a distributed mesh scenario; in the subsequent sections we refer to it as PMP algorithm. In the final part of this section, THR and PMP algorithms are briefly introduced.

THR is a call admission control algorithm for 802.16 distributed mesh mode. Calls are classified into three different classes and the admission decision is based on a few concepts:
- there is the presence of two checkpoints fixed along the available minislots: cp1 and cp2;
- there is a threshold value for bandwidth utilization;
- if the bandwidth utilization at checkpoint cp1 is less than threshold, all calls are admitted without distinguishing between priorities, otherwise, to admit a low priority call, the node searches for a frame from checkpoint cp2, and if there are sufficient availabilities, the request is admitted otherwise it is refused.

PMP is a call admission control algorithm referred to 802.16 Point to MultiPoint mode. In a call admission decision, the algorithm distinguishes between four different service classes: UGS (Unsolicited Grant Service), rtPS (real time Polling Service), nrtPS (not real time Polling Service) and BE (Best Effort). In our work, instead, three different traffic classes are considered and thus, to import the PMP call admission control proposed in [8] in mesh mode, UGS, rtPS and BE are mapped in traffic classes with priority “1”, “2” and “3” respectively.
With respect to the previous service mapping, the call admission decision is taken following these criteria:

### TABLE I. SIMULATION SETTINGS

<table>
<thead>
<tr>
<th>SIMULATION SETTINGS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PHY SETTINGS</strong></td>
<td></td>
</tr>
<tr>
<td>Modulation</td>
<td>OFDM, QPSK 1/2</td>
</tr>
<tr>
<td>BW (Channel Bandwidth)</td>
<td>25 MHz</td>
</tr>
<tr>
<td>NFFT</td>
<td>256</td>
</tr>
<tr>
<td>G</td>
<td>1/8</td>
</tr>
<tr>
<td>Frame length</td>
<td>20 ms</td>
</tr>
<tr>
<td>Symbol efficiency</td>
<td>184 bits</td>
</tr>
<tr>
<td>Coverage radius</td>
<td>500 m</td>
</tr>
<tr>
<td><strong>MAC SETTINGS</strong></td>
<td></td>
</tr>
<tr>
<td>msh-ctrl-len</td>
<td>4</td>
</tr>
<tr>
<td>msh-dsch-num</td>
<td>4</td>
</tr>
<tr>
<td>msh-csch-data-fraction</td>
<td>0</td>
</tr>
<tr>
<td>scheduling-frame</td>
<td>1</td>
</tr>
<tr>
<td>data queue size</td>
<td>50</td>
</tr>
<tr>
<td><strong>SOURCE SETTINGS</strong></td>
<td></td>
</tr>
<tr>
<td>Number of sources</td>
<td>3 - 24</td>
</tr>
<tr>
<td>QoS delay constraints</td>
<td></td>
</tr>
<tr>
<td>Priority: 1</td>
<td>40 ms</td>
</tr>
<tr>
<td>Priority: 2</td>
<td>80 ms</td>
</tr>
<tr>
<td>Priority: 3</td>
<td>no</td>
</tr>
<tr>
<td>Source rate CBR</td>
<td></td>
</tr>
<tr>
<td>packet size</td>
<td>packets/s</td>
</tr>
<tr>
<td>Priority: 1</td>
<td>64 byte</td>
</tr>
<tr>
<td>Priority: 2</td>
<td>2500 byte</td>
</tr>
<tr>
<td>Priority: 3</td>
<td>2500 byte</td>
</tr>
<tr>
<td>Simulation run duration</td>
<td>500 s</td>
</tr>
<tr>
<td>Num runs for each configuration</td>
<td>10</td>
</tr>
<tr>
<td>Confidential Interval</td>
<td>95%</td>
</tr>
</tbody>
</table>

- Advent of request with priority equal to “1”: B₁ is the bandwidth request with priority “1”. When a node receives the new request, it verifies if the remaining bandwidth is less than the B₁ request. If the condition is verified, then the request is admitted, otherwise the mesh node verifies if this condition can become true considering the preemption of previously admitted requests with a priority of less than “1”. If the condition, with the new bandwidth availabilities, becomes true then the request is admitted otherwise it is refused.
- The mesh node receives a request with priority equal to “2”: B is the total bandwidth, B₂ is the request with priority “2” and R₁ is the bandwidth reserved for calls with priority “1”. If the bandwidth admitted to the previous requests with priority “2” plus B₂ is less than B–R₁ and if the remaining bandwidth is not less than B₂, the request is admitted; otherwise if the first condition is true, we can calculate the remaining bandwidth plus the amount of bandwidth released by preempted connections with priority “3”, if it is not less than B₂, the request is admitted otherwise it is refused.
- The request with priority “3” can use the remaining bandwidth, and can be preempted if necessary.

To make a comparison, we test the algorithms using an increasing source number: from 3 to 24. It is equally divided between the three traffic classes. In this way, with a source number equal to 24, we mean that the scenario contains 8 sources with priority “1”, 8 sources with priority “2” and 8 with priority “3”.

### B. Simulation Parameters

To evaluate performance of the algorithms, we select a set of parameters and use it to make a comparison between the proposed GCAD, PMP and THR algorithms. The performance parameters are the following:

- Packet loss percentage: defined as the percentage of total packets generated by sources and not delivered to destinations. A packet can be lost because the data queue of a mesh node is full, or because a request is not admitted;
- Throughput: the percentage of sent packets received at destination;
- Average number of refused requests: takes into account the average number of requests which are not admitted;
- Average end-to-end delay: the average time interval required by a packet to complete the path from source to destination;
- Delay jitter: is a variability measure of packet delay. The delay jitter is very important for real time application.

### C. Simulation Results

In figures 6, 7 and 8 the algorithms behaviour, in terms of packet loss percentage, is represented. Figure 6 considers the case of traffic class with a priority value equal to “1”. It is the highest priority traffic class. GCAD gives the best performance and always maintains the percentage under 5%.
The worst case is obtained by the PMP algorithm, as the number of sources increases, the percentage of packet loss, has a tendency to reach high values. Instead the GCAD trend grows slowly as network congestion increases. Also, by observing figures 7 and 8 GCAD shows the best trends. In figure 7 the worst case is related to the THR algorithm, while in figure 8 all the algorithms have a similar response to increasing congestion.

Considering the three cases, we can confirm the focus of the algorithms: THR tries to give more importance to priority “1”, neglecting priority “2” and “3”; PMP wants to put the two more important priority traffic classes on a par; GCAD has the same focus as PMP but it gives the best performance due to the presence of the defragmentation process.

The defragmentation process gives the algorithm the capability of accepting a higher number of requests and amount of bandwidth. This is visible in figures 9, 10 and 11. In figure 9 the only algorithm which has refused calls is THR, while figure 10 shows that GCAD is able to obtain a higher number of requests of priority “2” and this is confirmed by the packet loss depicted in figure 7. In figure 9, PMP did not refuse calls with higher priority, but it reached high packet loss values in a congested network, as PMP accepts all requests but gives them small amounts of bandwidth. Both PMP and GCAD behaviour depicted in figure 11 are similar.
In this way, evaluating the packet loss and the average number of refused calls, we can conclude that the introduction of a defragmentation process, allows the bandwidth to be managed in a more optimized way. The elimination of little availability gaps, gives the granter the possibility to create contiguous allocations, with the right size, to admit new calls.

Another way of viewing the capability of each algorithm to allow good results in terms of successfully transmitted packets, is to analyze the throughput performance. The throughput trends are depicted in figures 12, 13 and 14.

Figure 12 shows that our algorithm gives the best performance to sources with priority “1”. THR behaviour is also good, and this is because it preserves a bandwidth portion for sources with a higher priority. Instead, PMP performance, as depicted in figure 12, is characterized by a degradation due to bandwidth reserved for other kinds of traffic. Figures 13 and 14, relating to priority equal to “2” and “3” respectively, also confirm the quality of our proposal. Another point in favor of the GCAD algorithm is due to the greedy choice, in fact, if there is a sufficient number of minislots, it accepts each kind of request, and only in a second moment starts the preemption process if, and only if, necessary.

Figure 15. Average end-to-end delay: sources with priority equal to “1”.

Figure 16. Average end-to-end delay: sources with priority equal to “2”.

Figure 17. Average end-to-end delay: sources with priority equal to “3”.

In this way, evaluating the packet loss and the average number of refused calls, we can conclude that the introduction of a defragmentation process, allows the bandwidth to be managed in a more optimized way. The elimination of little availability gaps, gives the granter the possibility to create contiguous allocations, with the right size, to admit new calls.

Another way of viewing the capability of each algorithm to allow good results in terms of successfully transmitted packets, is to analyze the throughput performance. The throughput trends are depicted in figures 12, 13 and 14.
In figures 15, 16 and 17 average end-to-end packet delays are depicted. Figure 15 considers the priority “1” case. Observing the depicted trends, it is possible to see how the only algorithm, which respects the delay constraint, in each network condition, is the GCAD algorithm (it is necessary to remember that the QoS constraint for data flow with priority value equal to “1” is an end-to-end delay value less than 0.04 s). PMP and THR do not respect the QoS delay constraint in a scenario with 18, 21 and 24 sources.

Also in the priority “2” case the THR algorithm overflows the delay threshold (in this case the end-to-end delay constraint is less than 0.08 s). Instead in the priority “3” case, there are no quality thresholds. The GCAD algorithm does not give the best performance in each case and this is due to the presence of the defragmentation process. On one hand it allows the optimization of bandwidth management, but on the other it pays for this with an imperfect delay behaviour.

Finally in figures 18 and 19 we depict the jitter trends related to priority “1” and “2” cases.

The GCAD algorithm, in traffic with priority equal to “1”, gives the best results. Its jitter trend is regular and the values are not great even in a congested network. This delay jitter characteristic is very important in a real-time application. Instead, by observing figure 19, we can see that the delay jitter trend is more irregular, this is surely due to the defragmentation process. In fact, it can introduce variable delays, as it causes a new bandwidth bargaining process of connections involved in defragmentation.

VII. CONCLUSIONS

In this paper we have presented a new call admission control algorithm for 802.16 distributed mesh networks. The algorithm is characterized by an initial greedy choice, by a preemption and a defragmentation process. The proposed algorithm has been tested in a scenario of 25 mesh nodes with a max number of 24 sources. The performance of the proposed GCAD algorithm is evaluated in terms of throughput, average end-to-end delay, average delay jitter, number of refused requests and packet loss percentage. The GCAD performances are compared with those of another two CAC algorithms in literature. The GCAD algorithm gives the best performance due to the presence of a defragmentation process. It allows an optimized management of minislot allocation.

REFERENCES


Floriano De Rango graduated in Computer Science Engineering in October 2000, and a Ph.D. in Electronics and Communications engineering in January 2005, both at the University of Calabria, Italy. From January 2000 to October 2000 he had a scholarship in the Telecom Research LAB C.S.E.L.T. in Turin. From March 2004 to November 2004 he was visiting Researcher at the University of California at Los Angeles (UCLA). From November 2004 until September 2007 he was a Research Fellow in the D.E.I.S. Department, University of Calabria where he is now Assistant Professor. He was recipient of the Young Researcher Award in 2007. He served as reviewer and TPC member for many International Conferences such as IEEE VTC, ICC, WCNC, Globecom, Med Hoc Net, SPECTS, WirelessCOM, WinSys and reviewer for many journals such as IEEE Communication Letters, JSAC, IEEE Trans. on Vehicular Technology, Computer Communication, Eurasip JWCN, WINET etc. His interests include Satellite networks, IP QoS architectures, Adaptive Wireless Networks, Ad Hoc Networks and Pervasive Computing. He has co-authored more than 130 papers among International Journal and Conferences Proceedings.

Andrea Malfitano was born in Neuchatel (Switzerland) in 1977 and graduated in Computer Science Engineering. He currently is a Ph.D student at University of Calabria and collaborates with the telecommunication group of the DEIS Dept. His main research interests are about topics related to physical and MAC (Medium Access Control) levels of IEEE 802.16 protocol stack. In particular he carries out research into signal impairment effects due to environment, transmission techniques of the IEEE 802.16 protocol, transmission channel modeling, scheduling issues, mechanisms to support QoS and call admission control topics.

Salvatore Marano, graduated in Electronics Engineering at the University of Rome in 1973. In 1974 he joined the Fondazione Ugo Bordoni. Between 1976 and 1977 he worked at the ITT Laboratory in Leeds, United Kingdom. Since 1979 he has been an Associate Professor at the University of Calabria, Italy. He was reviewer for many journals such as IEEE Communication Letters, JSAC, IEEE Trans. on Vehicular Technology, IEEE Transaction on Wireless Comm., European Transaction on Telecommunication Journal. His research interests include performance evaluation in mobile communication systems, satellite systems and 3g/4G networks. He has published more than 160 papers in international conferences and journals.