This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

Bandwidth Balancing in Multi-Channel IEEE 802.16 Wireless Mesh networks

Claudio Cicconetti*†, Ian F. Akyildiz*, and Luciano Lenzini†
* School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332–0250
† Dipartimento di Ingegneria dell'Informazione
University of Pisa, Via Diotisalvi, 2 – 56010 Pisa, Italy

***Abstract*— In wireless mesh networks, the end-to-end throughput of traffic flows depends on the path length, i.e. the higher the number of hops, the lower becomes the throughput. In this paper, a Fair End-to-end Bandwidth Allocation (FEBA) algorithm is introduced to solve this problem. FEBA is implemented at the Medium Access Control (MAC) layer of single-radio, multiple channels IEEE 802.16 mesh nodes, operated in a distributed coordinated scheduling mode. FEBA negotiates bandwidth among neighbors to assign a fair share to each end-to-end traffic flow. This is carried out in two steps. First, bandwidth is requested and granted in a round-robin fashion where heavily loaded links are provided with a proportionally higher amount of service than the lightly loaded links at each round. Second, at each output link, packets from different traffic flows are buffered in separate queues which are served by the Deficit Round Robin (DRR) scheduling algorithm. If multiple channels are available, all of them are shared evenly in order to increase the network capacity due to frequency reuse. The performance of FEBA is evaluated by extensive simulations and is shown to provide fairness by balancing the bandwidth among traffic flows.**

I. Introduction

Wireless Mesh Networks (WMNs) are emerging as a key technology for next generation wireless networking. Due to their several advantages compared to other wireless networks, WMNs are undergoing a very fast development progress and inspiring numerous applications. The WMN architecture, in general, consists of two tiers [1]: *backhaul* and *access* tiers where the *backhaul* tier consists of *wireless mesh routers* which create a multi-hop ad hoc network and provide Internet or intra-WMN connections to *wireless mesh clients* in the *access* tier. Wireless mesh routers are fixed devices with unlimited energy, high computational and communication capabilities.

Recently, some research has been conducted to use the well-known IEEE 802.11 technology for the backhaul tier which has performance problems in its current form [2]. Indeed, in the existing IEEE 802.11 technology there are few available channels [3], the transmission range is very limited [4] unless expensive external amplified antennae are employed, the Medium Access Control (MAC) protocol achieves low performance for multi-hop traffic flows [5]. In particular, the fairness among traffic flows traversing a different number of hops is severely affected. Specifically, the available network capacity, accordingly the system throughput, decreases with the increasing number of hops because (i) some nodes experience backoff more often than others due to the "hidden terminal" problem, and (ii) flows with a longer path length have more contentions for medium access than the flows originated closer to their destination. More contentions result in higher probability for collisions and losses. Several solutions are suggested in the literature to solve this problem. These solutions use a MAC protocol based on Time Division Multiple Access (TDMA) [6] [7] [8] with highly simplifying assumptions which make them impractical for actual network deployments. The working group IEEE 802.11s is established to investigate these research problems [9]. However, the research is still in the infancy phase [10].

An alternative to IEEE 802.11 is the IEEE 802.16 standard [11] which is specifically designed for the backhaul tier of WMNs and includes a TDMA MAC protocol operating in *mesh mode* where nodes coordinate among themselves to transmit packets in a multi-hop manner. There are two coordination modes: *centralized* and *distributed*. In the *centralized* mode, the Base Station (BS) is responsible for defining the schedule of transmissions in the entire network. In the *distributed* mode, transmissions are scheduled in a fully distributed fashion without requiring any interaction with the BS. The distributed mode is more flexible and responsive than the centralized mode, since decisions are taken locally by nodes according to their current traffic load and physical channel status. In this study we consider the distributed mode alone.

In the distributed mode, the IEEE 802.16 standard specifies a MAC protocol to coordinate the transmission of control messages in a collision-free manner. This is modeled and its performance is evaluated in [12]. On the other hand, the bandwidth allocation problem in the distributed mode is left unsolved by the IEEE 802.16 standard so far except providing some control messages that may be used for this purpose, such as bandwidth requests and grants.

In this paper we propose a Fair End-to-end Bandwidth Allocation (FEBA) algorithm for IEEE 802.16 nodes to negotiate bandwidth in a multi-channel environment. FEBA is aimed at providing a fair bandwidth allocation, in terms of throughput, to end-to-end traffic flows regardless of their path length. This is done by assigning bandwidth requests and grants at each neighbor in a round-robin fashion, with an amount of service proportional to the number of traffic flows going to or coming from the neighbor, respectively. This procedure

0743-166X/07/$25.00 ©2007 IEEE

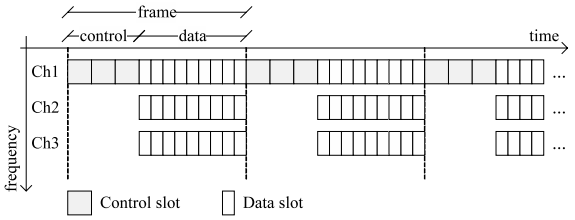

Fig. 1. Example of frame structure, with three channels.

is carried out locally by each node in the network without end-to-end signaling. The effectiveness of FEBA is evaluated through packet-level simulation of traffic flows with infinite bandwidth demands. Furthermore, we assess the performance of finite-load traffic flows with respect to several network configurations and system parameters.

The remainder of the paper is organized as follows. In Section II we report an overview of the IEEE 802.16 mesh MAC protocol with distributed coordinated scheduling. In Section III we describe the FEBA algorithm which is then evaluated by simulation in Section IV. We conclude the paper in Section V.

II. IEEE 802.16 MESH

In this section we discuss the aspects of the IEEE 802.16 MAC protocol relevant to the mesh mode with distributed scheduling. As already introduced, data transmission is coordinated among nodes in a fully distributed manner. Hereafter, we adopt the IEEE 802.16 terminology which defines two nodes that can communicate between each other as *neighbors*. In IEEE 802.16 a logical *link* is set up between any two neighbors by means of a link establishment procedure. Instead nodes that are not in the transmission range of one another but have a common neighbor are denoted as two-hop neighbors.

The time is partitioned into frames of fixed duration. Each frame consists of a control sub-frame and a data sub-frame, as illustrated in Fig. 1. Control sub-frames are partitioned into slots of fixed duration (hereafter, *control slots*), which are accessed by nodes based on the distributed election procedure specified by the standard. This ensures that, in a steady state, each node gets the opportunity to transmit control messages in a regular, though not periodical, manner. An IEEE 802.16 mesh network can employ up to 16 multiple non-interfering channels for data transmission to increase the available transmission capacity for nearby nodes which cannot exploit spatial reuse. However, control messages are transmitted by all nodes in the network in the same channel, e.g., channel Ch1 in Fig. 1. Data sub-frames consist of a fixed number of data mini-slots (hereafter, *slots*). The amount of bytes conveyed by a slot depends on the Modulation and Coding Scheme (MCS) used by the sender to transmit data to the receiver. Every node dynamically adapts the MCS from neighbor to neighbor based on measurements of the received signal quality at the physical layer.

Nodes use Mesh Distributed Schedule (MSH-DSCH) messages for bandwidth negotiation. In fact, data transmission is coordinated by means of a three-way handshake procedure: (i) a node, namely the *requester*, asks a neighbor node, namely the *granter*, to allocate some bandwidth; (ii) the granter advertises a set of slots as 'granted' to the requester; (iii) the requester confirms that it will actually use that set of slots (or part thereof) to transmit data.

More specifically, MSH-DSCH messages contain a list of information elements (IEs), classified by the IEEE 802.16 standard into four types. A *request IE* indicates that the requester has data addressed to the granter awaiting transmission, i.e. backlog. The granter reserves bandwidth for the requester using *grant IEs*, each containing a range of slots over a range of frames in a given channel. A grant is thus expressed as a triple $\langle$slot range, frame range, channel$\rangle$, e.g. $\langle[3, 8], [4, 5], 1\rangle$ represents the slots numbered from 3 to 8 in the data sub-frame of the fourth and fifth frame since the grant is issued, in channel 1. The same set of parameters is also used in *confirmation IEs*, which are used by the requester to complete the three-way handshake procedure. Finally, *availability IEs* can be used to report slots that cannot be used by the requester to transmit or receive data.

We now discuss how the messages above can be exploited to avoid collision. First of all, we assume that a node is not able to decode data which is received on a channel from a neighbor node if either (i) the receiving node is transmitting on the same channel, or (ii) another neighbor of the receiving node is transmitting on the same channel. This is called the "protocol-model" in [13]. Therefore nodes need to keep track of all the combinations $\langle$slot, frame, channel$\rangle$ that cannot be granted to the requester because any of the following conditions is true [1]: (i) the granter transmits/receives in $\langle$slot, frame$\rangle$; (ii) the requester transmits/receives in $\langle$slot, frame$\rangle$; (iii) one of the requester's neighbors transmits in $\langle$slot, frame, channel$\rangle$. Conditions (i) and (ii) are because nodes have a single radio, thus they can either receive from or transmit to a single channel at a given time, while the last one results from the "protocol-model" assumption.

A two-way handshake, i.e. without the confirmation step, is sufficient as long as all nodes are in the same transmission range. However, this is not always the case in a WMN. Let us consider the example given in Fig. 2(a), where a solid line means that there is a link between two nodes. Assume that nodes A and C send bandwidth requests to nodes B and D, respectively, at approximately the same time. Node B grants an interval of slots $x = \langle$slot range, frame range, channel$\rangle$. However D is not aware of this grant, since it is not a neighbor of B. Thus, it might allocate the same interval of slots x, which cannot be used by C for data transmission. This, in fact, would result in B not being able to decode data transmitted by A. Thus, C refrains from confirming x to D. We call this event *grant withdrawal*.

We propose a procedure, namely *re-granting*, to resolve the grant withdrawal which is not specified by the IEEE

[1] This can be done efficiently via a set of *grant bitmaps* to indicate whether each of the conditions is true. Grant bitmaps are updated whenever a node receives/transmits an MSH-DSCH message.

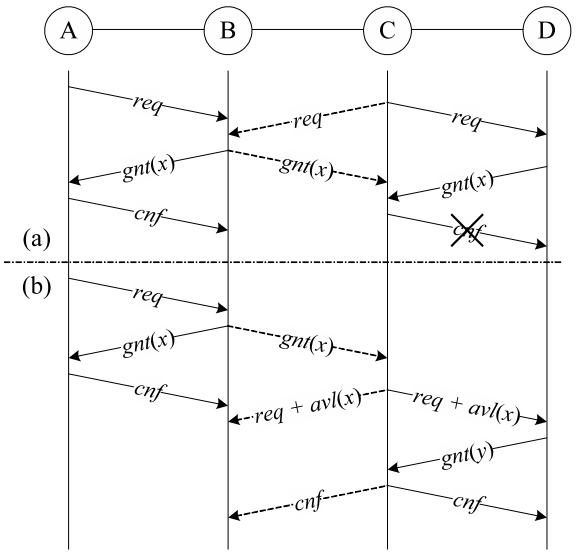

Fig. 2. Two examples of three-way bandwidth negotiation. Control transmissions overheard by a node, but not directly addressed to that node, are represented as dashed lines.

802.16. In our procedure, the granter sends an unsolicited new grant to the requester that compensates the slots that have not been confirmed. A straightforward alternative to re-granting consists of the requester sending a copy of the failed bandwidth request again, which would however increase the bandwidth negotiation overhead.

Withdrawing grants negatively affects the system performance since it wastes capacity in the control slots and increases the per-hop latency. Therefore, it should be avoided whenever possible. In some cases, this can be achieved through the notification of availabilities. Let us consider the example given in Fig. 2(b). Unlike node D, node C knows that the interval of slots x is not available for transmission by itself. Thus, it can notify this unavailability to node D (represented as $avl(x)$ in Fig. 2), which will then refrain from granting that interval to C. Unfortunately, using availabilities does not avoid re-granting in all cases. For example, in Fig. 2(a), node C receives the grant after it has the chance to notify node D.

III. Fair End-to-end Bandwidth Allocation

We assume that the network topology is fixed, i.e. nodes are not added to or removed from the network. Moreover, even though the quality perceived by nodes on different links may change over time, links are considered stable enough not to be torn down. This assumption is motivated by the fact that IEEE 802.16 nodes are intended to be employed as wireless mesh routers in the backhaul tier of a highly reliable WMN, which ensures stable routes. Finally, we assume that each node has a single radio interface, which can dynamically switch to one channel at a time, among several available channels. Note that the latter is not a requirement. In fact, FEBA does not rely on the use of multiple channels, which can however boost the performance in terms of the overall achievable throughput.

MAC Service Data Units (SDUs) received from upper layers are fed into one of the packet schedulers (one per link) based on its next-hop node. Each packet scheduler selects the packets to be transmitted, encapsulates them into MAC Protocol Data Units (PDUs), and passes the latter to the physical layer. If needed, it can fragment an SDU into multiple PDUs to avoid capacity wastage. As confirmed by performance evaluation in Section IV, the packet scheduler is a critical component to balance the load of multiple traffic flows directed to the same neighbor. In particular, the transmission buffers for different traffic flows should be kept separate at each packet scheduler. Among several existing packet scheduling algorithms in the literature we chose the Deficit Round Robin (DRR) [14] algorithm because it achieves fair queueing for variable length packets, can operate at $\mathcal{O}(1)$ complexity, and its implementation is easy.

A. Motivation

The "Fairness" is a desirable property for any MAC protocol employed as the backhaul tier of a WMN. Although this notion is well-defined in single-hop networks, it is not clear in case of multi-hop networks. The packets that have the same source and/or the same destination can be treated as a single aggregate, which leads to the concept of *traffic flow*. In the literature, the definition of a traffic flow typically depends on the target application of the proposed solution. For instance, in [5] the authors developed a fairness reference model for WMN nodes that act as Transit Access Points (TAPs) of traffic to/from the Internet. Since each TAP is assumed to correspond to a single residence/public hot spot, all micro-flows originated at a TAP are treated as a single aggregate. Instead, in this work we do not consider any specific assumption on where the ingress/egress points of the WMN backhaul are located, if any. Therefore, we argue that any flow of packets identified by a ⟨source, destination⟩ ought to obtain the same treatment from the network, and employ the following definition:

Definition 1 (Traffic flow): A traffic flow is a stream of IP datagrams from a source to a destination node. Hence, a traffic flow may aggregate multiple flows of data packets, e.g., multiple TCP connections, provided that they all have the same source and destination nodes.

According to the definition above, flows that traverse a high number of hops should not be penalized with respect to others that have a shorter path, i.e. no spatial bias should exist.

Moreover, as it is known, in wireless networks the transmission rate of links can vary over time. A typical problem is thus whether the fairness should be measured in terms of the amount of bandwidth, i.e. bytes, or time that is consumed by a traffic flow [15]. In the former case, nodes that perceive worse channel conditions than others will consume more resources to transmit their fair share of bandwidth. We consider this case to be the most relevant to our work, since nodes are assumed to be fixed, hence transmission rates are bound to remain the same for long periods.

As a summary, we aim at providing traffic flows, as defined

above, with max-min fair[2] access to the network resources, in terms of throughput, regardless of their spatial bias. This is done by defining a bandwidth allocation algorithm, i.e. FEBA, that uses the set of messages defined in the IEEE 802.16 mesh MAC protocol with distributed coordinated scheduling, and does not rely on end-to-end signaling. The basic idea is that each node assigns bandwidth requests and grants in a round-robin manner, where the amount of bandwidth allocated, in bytes, is proportional to the number of traffic flows that will use it. The data structures and procedures of FEBA are described in the remainder of this section while its effectiveness is evaluated through simulations in Section IV.

B. Bandwidth Request/Grant Data Structures

In the following we define a traffic flow from node i to node j to be *active* if there are SDUs originated at node i directed to node j. Since in IEEE 802.16 there is no end-to-end signaling for data transmission, each node x along the path from i to j has to keep track of the set of active flows. Specifically, as soon as an SDU is received by x, it adds $\langle i, j\rangle$ to the set of active flows, if not already present, based on the source and destination IP addresses. Node x instead removes $\langle i, j\rangle$ from the set of active flows after it has not received SDUs that belong to that flow for a timeout period. We chose the latter to be equal to the default TCP Maximum Segment Lifetime (MSL), i.e. two minutes. Keeping track of the flows that are currently active increases the spatial and temporal computational complexity of the procedure, which is a classical problem of flow-based architectures. However, we do not consider this to be an issue for a real implementation of IEEE 802.16 in the backhaul of a WMN because of (i) the high expected computational capabilities of wireless mesh routers, and (ii) the relatively small number of nodes, which is more likely to be in the order of tens than in that of thousands.

A generic node X maintains a virtual *requesting* queue towards any of its neighbors, say Y, containing the values of the following state variables [3]:

- req_Y^{out}: number of bytes that X has notified to Y by means of request IEs;
- cnf_Y^{out}: number of bytes that X has confirmed to Y;
- $blog_Y^{out}$: number of bytes awaiting transmission at X towards Y for which no bandwidth requests have been issued.

If $blog_Y^{out} = 0$ the requesting queue is considered to be inactive, since the node does not need to be granted bandwidth by Y. The difference $req_Y^{out} - cnf_Y^{out}$ is the amount of pending bytes, i.e. bytes for which X awaits confirmation from Y. We enforce this amount to be smaller than a threshold, $pending_{max}$. Exceeding this threshold is an indication that the requester demands cannot be fulfilled by the granter at this time. We choose the value of $pending_{max}$ such that a single node is able to keep the pipe full, i.e. to utilize the entire channel capacity.

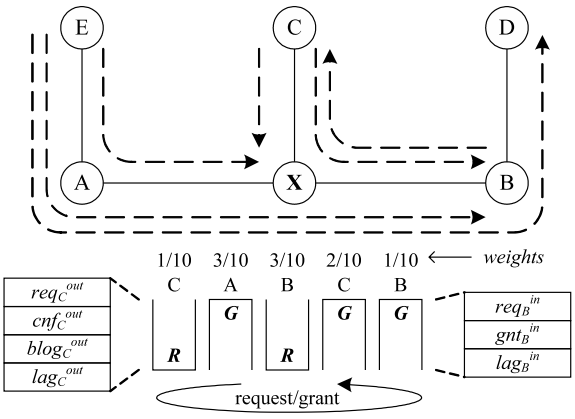

Fig. 3. Bandwidth request/grant data structure at node X.

Additionally, X maintains a virtual *granting* queue for any of its neighbors, say Y, containing the values of the following state variables:

- req_Y^{in}: number of bytes that Y has notified to X by means of request IEs;
- gnt_Y^{in}: number of bytes that X has granted to Y.

If $req_Y^{in} = gnt_Y^{in}$ the granting queue is considered to be inactive, since X granted the bytes requested by Y. Both request and grant inactive queues are removed from the set of active queues (called *active list* hereafter).

To explain the request/grant data structures, in Fig. 3 we give a simple network example with six nodes ($A-E$, X) and six traffic flows ($E \rightarrow B$, $E \rightarrow D$, $E \rightarrow X$, $C \rightarrow X$, $C \rightarrow B$, $B \rightarrow C$). An edge between two nodes means that they are neighbors. Active flows are represented by dashed arrows. We also illustrate the data structures used for bandwidth negotiation at node X (lag_i will be described at the end of this section). Requesting (granting) queues are labeled as ***R*** (***G***).

Each active queue, both requesting and granting, is assigned a *weight* (ϕ), which is used by the bandwidth request/grant procedure below. The weight ϕ_i of any queue i is computed so that the amount of service is proportional to the number of traffic flows under service:

$$\phi_i = \frac{\sum_{j \in \mathcal{A}} I_i(j)}{|\mathcal{A}|}, \tag{1}$$

where $\mathcal{A}$ is the set of all active traffic flows served by this node, j is an active flow, and $I_i(j)$ is an indicator function which equals 1 if j is under service at queue i, 0 otherwise [4]. Since each traffic flow is under service at exactly one queue, $\sum_i \phi_i = 1$. The bandwidth request/grant procedure is applied

[2] In general, a bandwidth allocation to different entities is said to be max-min fair if it is not possible to increase any bandwidth share without decreasing another bandwidth share which is already smaller than that.

[3] In IEEE 802.16 bandwidth request and grants are expressed in units of slots. Since the state variables are in terms of bytes, we assume that nodes convert between bytes and slots, depending on the current MCS employed, whenever needed.

[4] By assigning different values to the indicator function it would be possible to allow some traffic flows to have a higher priority than others. However, this is not analyzed in this paper.

to both requesting and granting active queues which are served in a round-robin fashion: at each round queue i is entitled to serve $\phi_i F_{RR}$ bytes, where F_{RR} is a system parameter, called *target round duration*. Specifically, when defining the content of the MSH-DSCH message, each granting queue i is entitled to grant up to $\phi_i F_{RR}$ bytes to neighbor i, while each requesting queue i is entitled to request up to $\phi_i F_{RR}$ bytes from neighbor i. If the number of bytes requested from (granted to) neighbor i is smaller than $\phi_i F_{RR}$ the queue is removed from the active list after service.

Since in IEEE 802.16 the channel used by the requester is selected by the granter on a per-grant basis, slots are allocated in the time and frequency domains independently. However, not all data slots should have the same chance to be assigned by the granter. On the one hand, the granter should not schedule data slots "too early", i.e. before the requester has the chance to confirm them. For example, assume that the granter schedules in frame t a grant starting on frame $t+g$, but the next frame when the requester will have an opportunity to confirm that grant is after frame $t+g$. This would result in other nodes in the neighborhood refraining from granting those slots, while the requester will not use them because they are not unconfirmed. Thus, the granted bandwidth would be wasted. On the other hand, scheduling data slots "too late" would entail longer response times of the three-way handshake procedure, and hence greater per-hop packet transmission delays.

Definition 2 (Grant horizon): The grant horizon between a granter gnt and a requester req is defined as the range of frames where gnt is allowed to grant slots for data to be transmitted by req.

The value of the grant horizon depends on the interval between two consecutive turns to transmit an MSH-DSCH message (hereafter *control latency*) of both the requester and the granter. However, this interval is not the same for all nodes [12]: nodes in higher density zones of the network access control slots less frequently than other nodes in lower density zones. Therefore, in FEBA, each node uses a different grant horizon on a neighbor by neighbor basis, depending on the neighbor's control latency. Since the latter is not known *a priori* we estimate it through:

$$h_i^+ = \alpha \cdot h_i^- + (1-\alpha) \cdot h_i^{sampled}, \qquad (2)$$

where $h_i^{sampled}$ is the last sampled control latency, h_i^+ and h_i^- are the new and old estimations, respectively, and α and $1-\alpha$ are used to weigh the old and new estimations respectively. Based on our extensive simulations, the value for $\alpha = 0.1$ produces the most accurate estimations .

The grant horizon, at time t, in units of frames, can be expressed as $[t + h_{req}, t + h_{req} + h_{gnt}]$, where h_{req} and h_{gnt} are the control latency values of the requester and the granter, respectively, rounded up to the next multiple of the frame duration. Note that the higher is the estimated control latency of the granter, the larger the grant horizon becomes. In this way, nodes accessing control slots less frequently are allowed to assign grants over larger frame windows than those with smaller values of h_{gnt}. With regard to the *re-granting* introduced in Section II, we define the *re-grant horizon* as $[t+h_{req}+h_{gnt}+1, t+2h_{req}+h_{gnt}+1]$. This way "unsolicited" granting does not impair "regular" granting, since the sets of slots where they are performed are disjoint.

grant (i)
$lag_i^{in} \leftarrow \min\{lag_i^{in} + \phi_i F_{RR}, req_i^{in} - gnt_i^{in}\}$
$granted \leftarrow \textbf{fit}(i, lag_i^{in}, h_{neigh}, h_{neigh} + h_{self})$
$gnt_i^{in} \leftarrow gnt_i^{in} + granted$
$lag_i^{in} \leftarrow lag_i^{in} - granted$
if $(lag_i^{in} > lag_{max})$ terminate
if $(req_i^{in} = gnt_i^{in})$ remove $\langle i, in\rangle$ from the active list

request (i)
$lag_i^{out} \leftarrow \min\{lag_i^{out} + \phi_i F_{RR}, blog_i^{out}\}$
if $(req_i^{out} - cnf_i^{out} > pending_{max})$
    if $(lag_i^{out} > lag_{max})$ terminate
else
    req (lag_i^{out})
    $req_i^{out} \leftarrow req_i^{out} + needed$
    $lag_i^{out} \leftarrow 0$
    $blog_i^{out} \leftarrow blog_i^{out} - needed$
    if $(blog_i^{out} = 0)$ remove $\langle i, out\rangle$ from the active list

Fig. 4. Pseudo-code of the request and grant procedures for queue i.

Finally, there are cases when a queue i, though in the active list, is not eligible for service. More specifically, a granting queue i is not eligible for service when all the slots in the grant horizon are busy, i.e. it is not possible to grant any slots to neighbor i. On the other hand, a requesting queue i is not eligible for service when $req_i^{out} - cnf_i^{out} > pending_{max}$, i.e. the requester demands cannot be satisfied. Ineligible queues are not removed from the active list. However, we store in the variable lag_i the number of bytes that queue i could not consume while it was ineligible. In a subsequent turn when queue i eventually becomes eligible, it will receive an extra service equal to lag_i bytes. To prevent queue i from not being served at all, lag_i is bounded by a threshold, lag_{max} which is set to 2 F_{RR}. Note that the notion of "lagging" queue is known in the scheduling literature [16] where it is used for fair-queuing in wireless networks. However, those results cannot be directly applied to our problem in IEEE in 802.16 mesh networks.

C. Bandwidth Request/Grant Procedure

The pseudo-code to grant/request bandwidth to/from node i is reported in Fig. 4. The procedures reported are performed in a round-robin manner over all the queues in the active list until any of the following conditions becomes true: (i) the active list becomes empty; (ii) there is not enough space left in the control slot to add another IE to the MSH-DSCH message; (iii) the lag of the queue under service exceeds lag_{max}.

With regard to bandwidth request, the sub-function **req** in

Fig. 4 simply adds a request IE to the MSH-DSCH message. On the other hand, as far as bandwidth granting is concerned, the sub-function **fit**, which adds grant IEs to the MSH-DSCH message, is more complex. In fact it has to find a set of contiguous slots in the specified grant horizon such that the requester can transmit $pending$ bytes. This might result in multiple grant IEs added to the MSH-DSCH message. Since this problem of "fitting" the amount of bytes into the grant horizon is constrained by previous grants, minimizing the number of IEs required in the MSH-DSCH message at a reasonable computational complexity is a challenging task. However, since control slots have fixed durations, sub-optimal allocations do not affect the performance, as long as they do not entail an earlier termination of the request/grant procedure due to the exhaustion of the control slot capacity.

Therefore, we apply the following algorithm: (i) randomly select a channel; (ii) find the first available slot in the first frame of the grant horizon; (iii) if no slots are available, move to the next channel; (iv) if all channels have been searched, move to the next available frame of the grant horizon. First, slots are visited in temporal order, so as to reduce the bandwidth negotiation latency. Second, nodes two hops away cannot listen to each other concerning the MSH-DSCH messages. Thus, they may grant the same data slots to neighboring nodes which are left unconfirmed if they are in the same channel. Randomly selecting the channel reduces the occurrence probability of such an event. Finally, granting as many contiguous slots as possible in the same $\langle$frame, channel$\rangle$ reduces the number of grant IEs in the MSH-DSCH message.

A simple example of this procedure is illustrated in Fig. 5, where the dashed line represents the visiting order of slots, and crossed boxes represent slots that cannot be granted because they have already been allocated for data transmission by the granter or the granter's neighbors. Assume that the granter needs to assign three slots for the transmission of its i-th neighbor (i.e. the argument lag_i^{in} of **fit** in Fig. 4 is equal to the number of bytes that can be transmitted in three slots with the MCS currently employed by i). It first randomly selects one channel between the two available ones, say channel 1. Then, it visits the grant horizon from the earliest frame, i.e. x in Fig. 5. Slots 3 and 4 in channel 1 are available, thus a two-slot grant is issued. Since there is still one slot to be granted, the granter continues searching for available slots. Note that slot 4 in channel 2 cannot be granted anymore, since it overlaps in time with slot 4 in channel 1. Thus, slot 1 of frame y in channel 1 is granted instead, which completes the procedure. In the example the following grant IEs are thus added to MSH-DSCH: $\langle[3,4],[x,x],1\rangle$ and $\langle[1,1],[y,y],1\rangle$.

IV. Performance Evaluation

We first define the settings under which the performance evaluation is carried out in Section IV-A, and then introduce the traffic models and metrics in Section IV-B. Simulation results are presented last, from Section IV-C to Section IV-E.

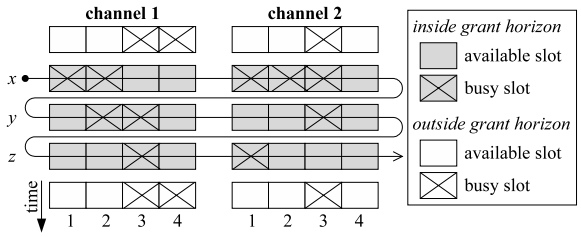

Fig. 5. Example of grant allocation of data slots within the grant horizon.

A. Simulation Environment

The parameters used in simulations are those specified in the IEEE 802.16 standard as system profile *profP3_10* [11]. Specifically, the channel bandwidth is 10 MHz, with a frame duration of 4 ms, including both control and data sub-frames. The *XmtHoldoffExponent* has been set to 0. Unless stated otherwise, there are four control slots per frame, and the nodes employ the 16-QAM-1/2 MCS, which yields a raw physical bandwidth of about 13 Mb/s per channel. We do not take into account channel errors, which allows us to focus specifically on the systems performance at the MAC layer.

We have implemented the IEEE 802.16 mesh mode with coordinated scheduling in the *ns2* network simulator [17]. Each traffic flow had a separate 100 kB buffer. The simulation output evaluation has been carried out using the method of independent replications [18]. For each scenario we ran 20 independent replications, 120 s each, with a 30 s initial warm-up period. Confidence intervals are not drawn whenever negligible.

B. Performance Metrics and Workload Characterization

Several indices are considered for the purpose of assessing the system performance. First, the end-to-end throughput (or *throughput*) of a traffic flow is the number of bits received by the destination node per second, without any MAC overhead. Second, the *MAC throughput* of a node is the number of bits received by the MAC layer of that node per second, regardless of the traffic flow to which the received data belong. The end-to-end delay (or *delay*) of a packet is the time interval between the arrival time of this packet at the network layer of the sender node, and the time when this packet is completely delivered to the network layer at the destination node. Finally, the fairness index is defined as the ratio $\left(\sum_{i=1}^{n} x_i\right)^2 / \left(n \sum_{i=1}^{n} x_i^2\right)$, where n denotes the number of traffic flows, and x_i the throughput of the i-th traffic flow. In the simulated scenarios the fairness objectives enumerated in Section III-A are reached if bandwidth is evenly partitioned among traffic flows, i.e. the fairness index is close to 1.

We consider three types of traffic: *asymptotic*, *Internet*, and *telnet*. If a node has an *asymptotic* traffic flow towards a destination, there is a constant bit-rate stream of 1000 bytes packets generated by the source node to the destination at a rate equal to the raw channel bandwidth. This emulates infinite bandwidth demands. The *Internet* traffic model instead has been proposed in [19] as the reference traffic model for

characterizing the performance of IEEE 802.16 networks. An Internet source generates 192 bytes packets, based on the super-imposition of four Interrupted Poisson Processes (IPPs), whose parameters are given in [19]. Finally, *telnet* traffic is generated via the *Telnet* ns2 application and uses the *NewReno* flavor of TCP, with the default ns2 configuration parameters.

C. Chain Topology

Here we compare FEBA to a *Greedy* approach where each node requests and grants as much bandwidth as possible at each turn to transmit an MSH-DSCH message. Additionally, we compare the DRR algorithm for packet scheduling to a First-In-First-Out (FIFO) scheduler where all SDUs output at the same link are enqueued into the same buffer. For this purpose we simulate a network with an increasing number of nodes, from 2 to 10, arranged in a *chain* topology. Each node has an asymptotic traffic flow directed to the chain end-point node. The raw channel bandwidth for data transmission is about 6.7 Mb/s, since nodes employ the QPSK-1/2 MCS.

We first consider a single-channel case. In Fig. 6 we show the sum of the end-to-end and MAC throughput of all traffic flows. As can be seen, the MAC throughput increases with the number of nodes, because the higher the number of nodes, the higher the amount of data that can be transmitted in the network, due to spatial reuse. In fact, the MAC throughput is constant when the number of nodes is smaller than or equal to four, where there is no spatial reuse. The MAC throughput with Greedy is significantly higher than that with FEBA, since the latter is designed to trade channel utilization for fairness among traffic flows. Moreover, the FEBA/DRR achieves an increasingly lower MAC throughput than the FEBA/FIFO when the number of nodes increases. This is because FEBA/FIFO favors traffic flows with a shorter number of hops.

Let us now consider the end-to-end throughput (see Fig. 6), which steeply decreases as the number of nodes increases regardless of the scheme adopted. This is because an increasing fraction of the channel capacity is employed to relay packets at intermediate nodes. For instance, with three nodes the end-to-end throughput achieved by FEBA/DRR is about 2/3 of the available raw bandwidth: 1/3 is consumed by the traffic flow that is one hop from the destination, and 2/3 are consumed by the other one that has a length of two hops. This way they both achieve the same end-to-end throughput. Moreover, FEBA/DRR achieves the highest end-to-end throughput compared to other schemes which are demonstrated in Fig. 6. In fact, with Greedy, most wireless resources are employed to transmit data to neighbors. This creates many bottlenecks along the network, which obstructs those traffic flows that are farther from the destination node, hence severely degrades their throughput performance. In fact, a large amount of data transmitted over the wireless channel is dropped by intermediate nodes due to buffer overflow. This phenomenon has been observed in the context of IEEE 802.11 networks, and has been analyzed (e.g.) in [5]. Such an undesirable situation is prevented by FEBA, which has a flow-based architecture.

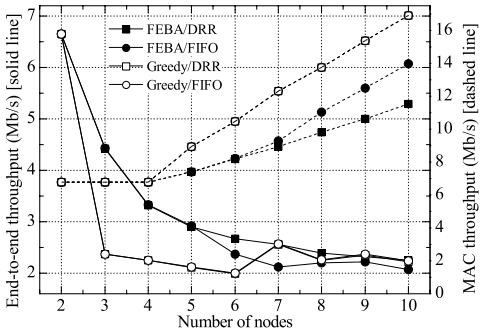

Fig. 6. *Chain topology*. Sum of the end-to-end/MAC throughput of all traffic flows, in solid and dashed lines, respectively.

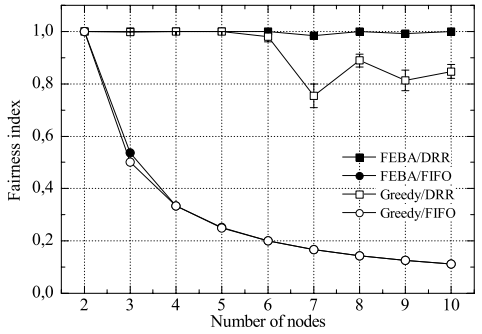

Fig. 7. *Chain topology*. Fairness index of the end-to-end throughput.

However, bandwidth allocation alone is effective only when the number of nodes is smaller than six. This limit is overcome by coupling FEBA with the DRR packet scheduling algorithm, which in fact over-performs FEBA/FIFO in the full range of nodes considered in this scenario.

The results discussed above are confirmed in Fig. 7, which shows the fairness index computed over time windows of 100 ms. As can be seen, FEBA/DRR is the only combination to achieve almost perfect fairness among all the traffic flows in all cases. On the other hand, Greedy/DRR is only fair while the number of nodes is relatively small, i.e. below seven. In Fig. 7 the end-to-end throughput of Greedy/DRR has a sharp increase, which is due to the almost complete starvation of traffic flows far away from the destination node, and hence a decrease in fairness. Finally, the packet scheduler FIFO exhibits very poor fairness even with three nodes, regardless of the bandwidth allocation algorithm. This is because SDUs that need to be relayed are enqueued at the same buffer that stores local SDUs, which are generated at a rate much higher than the arrival rate from neighbors.

In regard to FEBA/DRR in all investigated experiments we verified that the fairness index is almost 1.

We now repeat the same scenario above, with nodes employing 16-QAM-1/2 and 64-QAM-2/3 MCSs and with two channels. The sum of the end-to-end throughput of all flows is given in Fig. 8. In the single-channel case the throughput is proportional to the MCS efficiency: the QPSK-1/2 throughput is about half of the 16-QAM-1/2 one, which in turn is about

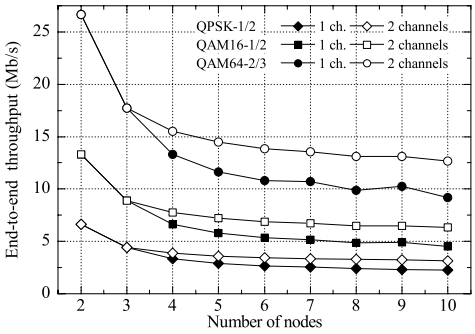

Fig. 8. *Chain topology*. Sum of the end-to-end throughput of all nodes.

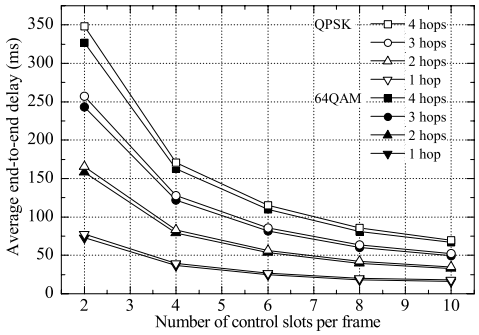

Fig. 9. *Chain topology* with five-nodes. Average end-to-end delay of telnet traffic.

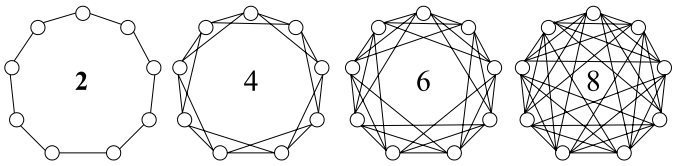

Fig. 10. Connectivity graphs of the networks in Section IV-D

half of that with 64-QAM-2/3. On the other hand, adding a second channel does *not* double the end-to-end throughput. This is because all nodes have a single radio, thus data transmissions between different pairs of nodes at the same time, on different channels, can only occur between disjoint pairs. For example, with three nodes only, it is not possible to exploit the second channel because the middle node always takes part in the communication. Adding further channels, in this case, does not improve the system performance significantly.

We now evaluate the performance of telnet traffic, in terms of the average delay, in a chain topology of five nodes. Specifically, each node has a telnet traffic flow towards the chain end-point node. Throughput measures are not relevant in this scenario because the entire offered load is carried by the network. Since the average rate of a telnet source is small, i.e. about 14 kb/s, the delay is severely affected by the distance from the source to the destination, i.e. the number of hops. In fact, for each hop, nodes have to complete a three-way handshake for bandwidth negotiation, whose duration depends on the network density, the system parameter XmtHoldoffExponent, the frame duration, and the number of control slots per frame [12]. In Fig. 9 we show the average delay for 2 to 10 control slots, which corresponds to an overhead from 0.08 to 0.41, expressed as the ratio between the control sub-frame and frame durations. For any distance, the delays decrease almost proportionally to the number of control slots per frame. Instead employing a more robust MCS, e.g. 64-QAM-2/3 instead of QPSK-1/2, does not improve performance significantly, in terms of the average delay. In fact, this only reduces the time to transmit PDUs, which is a small fraction of the overall delay.

D. Densely Populated Network

In this scenario we evaluate the performance improvement, in terms of throughput, due to the use of multiple channels in a densely populated network. Specifically, we consider a network of 9 nodes, with an increasing number of neighbors, from 2, i.e. the connectivity graph is a *ring*, to 8, i.e. the connectivity graph is completely connected (*clique*), as shown in Fig. 10. Each node has an asymptotic traffic flow towards one of its neighbors.

In Fig. 11 we show the sum of the throughput of all traffic flows, with one to four available channels. As can be seen, all curves decrease when the network density increases, because the spatial reuse decreases. In fact, the more dense is a network, the higher is the number of nodes that compete for granting bandwidth in interfering links. However, in this scenario, employing multiple channels greatly improves the network throughput, since it allows nodes to exploit frequency reuse. This is especially true with a completely connected network, i.e. 8 neighbors/node, where every node receives all the MSH-DSCH messages advertised by neighbors. Therefore, all grants are always confirmed without re-granting.

E. Star Topology

We now evaluate the isolation among traffic flows. To this aim, we set up a network with 19 nodes, arranged in a topology shaped as a six-pointed star, as illustrated in Fig. 12. We denote the middle node as the *root* node. Let a traffic flow entering (leaving) the root be a *downlink* (*uplink*) flow. Each root's neighbor has a bi-directional Internet traffic flow towards the

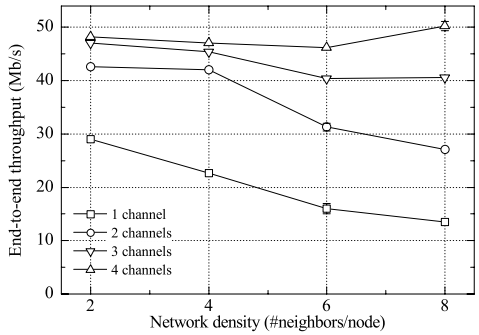

Fig. 11. *Densely populated network.* Sum of the end-to-end throughput of all traffic flows.

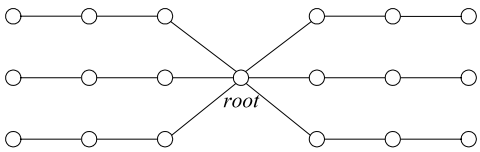

Fig. 12. Connectivity graph of the network in Section IV-E.

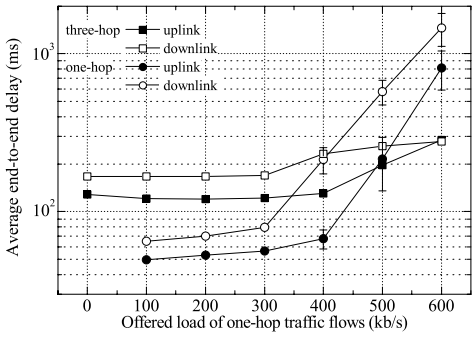

Fig. 13. *Star topology.* Average end-to-end delay.

root, at the offered load reported in the x-axis of Fig. 12. Each three-hop root's neighbor has a bi-directional Internet traffic flow at 100 kb/s towards the root.

In Fig. 13 we show the average delay of one-hop and three-hop traffic flows, in the uplink and downlink direction, respectively. We start from a scenario in which there are three-hop flows only and increase the network load slowly by first injecting one-hop flows at an offered load of 100 kb/s and reaching 600 kb/s.

While the network is lightly loaded, i.e. the offered load of one-hop traffic flows is below 400 kb/s, the average delay of three-hop traffic flows is almost constant, but significantly higher than that of one-hop flows. This is due to the three-way bandwidth negotiation at each hop. Both for one- and three-hop flows, the uplink delays are below the downlink delays, which is due to the root node being heavily loaded, with respect to the other nodes in the network. When the network becomes overloaded, i.e. the offered load of one-hop flows is above 400 kb/s, then the links between the root and its neighbors become bottlenecks, which slows down both the one-hop and the three-hop flows and causes longer delays, as shown in Fig. 13. Note that the one-hop delays increase steeply when the offered load is higher than 400 kb/s, which is caused by the channel being occupied, i.e. the bandwidth exhausted, and queues being created accordingly.

V. Conclusion

In this paper we have presented FEBA, a distributed algorithm for bandwidth balancing in multi-channel IEEE 802.16 WMNs. FEBA is specifically tailored to solve the problem of unfairness among traffic flows with different path length, which otherwise affects WMNs. Specifically, bandwidth requesting and granting is carried out in a round-robin fashion, where the amount of service at each round is proportional to the number of incoming or outgoing flows at each neighbor. Additionally, outgoing packets that belong to different flows are served using the DRR scheduling algorithm.

We have shown, via extensive simulations, that both mechanisms are necessary to provide fairness to end-to-end traffic flows, with respect to alternative approaches which assign bandwidth in a greedy manner and/or employ a FIFO algorithm to schedule packets. In regard to scenarios with multiple channels, we have shown that frequency diversity is exploited by FEBA to increase the network capacity, hence the achievable throughput. The effect of multi-channel is especially significant in densely populated networks, with low spatial reuse.

Acknowledgment

The authors would like to thank Prof. Eylem Ekici for his insightful comments on an earlier draft of this paper.

References

[1] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Computer Networks (Elsevier)*, vol. 47, pp. 445–487, Mar. 2005.

[2] P. Bahl, A. Adya, J. Padhye, and A. Wolman, "Reconsidering wireless systems with multiple radios," *ACM SIGCOMM Computer Communications Review*, vol. 34, pp. 39–46, Oct. 2004.

[3] A. Raniwala and T. Chiueh, "Architecture and algorithms for and IEEE 802.11-based multi-channel wireless mesh network," in *Proc. IEEE Infocom*, Miami, USA, Mar. 2005, pp. 2223–2234.

[4] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "IEEE 802.11b ad hoc networks: Performance measurements," *Cluster Computing (Springer)*, vol. 8, pp. 135–145, July 2005.

[5] V. Gambiroza, B. Sadeghi, and E. W. Knightly, "End-to-end performance and fairness in multihop wireless backhaul networks," in *Proc. ACM MobiCom*, Philadelphia, USA, Sept. 2004, pp. 287–301.

[6] T. Salonidis and L. Tassiular, "Distributed on-line schedule adaptation for balanced slot allocation in wireless ad hoc networks," in *Proc. IEEE IWQoS*, Montreal, Canada, June 2004, pp. 20–29.

[7] Y. Xue, B. Li, and K. Nahrstedt, "Optimal resource allocation in wireless ad hoc networks: A price-based approach," *IEEE Trans. Mobile Comput.*, vol. 5, pp. 347–364, Apr. 2006.

[8] L. Bui, A. Eryilmaz, R. Srikant, and X. Wu, "Joint asynchronous congestion control and distributed scheduling for multi-hop wireless networks," in *Proc. IEEE Infocom*, Barcelona, Spain, Apr. 2006.

[9] *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications. ESS Mesh Networking*, IEEE Std. 802.11s, 2006.

[10] R. Zhao, B. Walke, and G. R. Hiertz, "An efficient IEEE 802.11 ESS mesh network supporting quality-of-service," *IEEE J. Select. Areas Commun.*, vol. 24, pp. 2005–2017, Nov. 2006.

[11] *Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Std. 802.16, 2004.

[12] M. Cao, W. Ma, Q. Zhang, X. Wang, and W. Zhu, "Modelling and performance analysis of the distributed scheduler in IEEE 802.16 mesh mode," in *Proc. ACM MobiHoc*, Urbana-Champaign, USA, May 2005, pp. 78–89.

[13] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inform. Theory*, vol. 46, pp. 388–404, Mar. 2000.

[14] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," *IEEE/ACM Trans. Networking*, vol. 4, pp. 375–385, June 1996.

[15] Q. Dong, S. Banerjee, and B. Liu, "Throughput optimization and fair bandwidth allocation in multi-hop wireless LANs," in *Proc. IEEE Infocom*, Barcelona, Spain, Apr. 2006.

[16] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Trans. Networking*, vol. 7, pp. 473–489, Aug. 1999.

[17] Network simulator 2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[18] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*. McGraw-Hill, 2000.

[19] C. R. Baugh, J. Huang, R. Schwartz, and D. Trinkwon, "Traffic model for 802.16 TG3 MAC/PHY simulations," IEEE 802.16 Broadband Wireless Access Working Group, Tech. Rep., Mar. 2001.