Parag Mogre, Matthias Hollick, Stefan Dimitrov, Ralf Steinmetz:
Incorporating Spatial Reuse into Algorithms for Bandwidth Management and Scheduling in IEEE 802.16j Relay Networks. In: Proceedings of the 34th IEEE Conference on Local Computer Networks (IEEE LCN 2009), p. to appear, IEEE, October 2009.

Incorporating Spatial Reuse into Algorithms for Bandwidth Management and Scheduling in IEEE 802.16j Relay Networks

Parag S. Mogre, Matthias Hollick, Stefan Dimitrov, and Ralf Steinmetz
Email: {pmogre,mhollick,sdimitrov,rst}@kom.tu-darmstadt.de
Multimedia Communications Lab (KOM), Technische Universität Darmstadt,
Merckstr. 25, 64283 Darmstadt, Germany

***Abstract*—The IEEE 802.16 Standard recently introduced an additional relay mode of operation permitting coverage extension of the Point-to-Multipoint (PMP) based WiMAX networks. Relays can further aid in increasing the achievable data rates for Subscriber Stations (SS) in the network. Although bandwidth management and scheduling have been studied in some detail in traditional IEEE 802.16 PMP networks, the corresponding study for relay networks is far from complete. A trivial extension to the scheduling and bandwidth allocation algorithms used in the PMP mode for the relay networks leads to inefficient bandwidth utilization and allocation. Of particular interest here is the possibility of spatial reuse of allocated blocks of bandwidth which is permissible in relay networks. This paper investigates and designs bandwidth allocation and scheduling algorithms for IEEE 802.16 based relay networks, considering the support for spatial reuse to additionally improve the throughput while at the same time supporting quality of service (QoS). We research the issues which need to be considered when designing bandwidth allocation algorithms for relay networks. Finally, as a proof of concept we provide a thorough simulation study to investigate the performance of the designed algorithms. The results validate the design choices we presented, and also provide insights into areas for further research in WiMAX relay networks.**

I. Introduction

Relay nodes promise to increase the range/coverage as well as the capacity/bandwidth efficiency of wireless networks [1]. As a result, contemporary wireless network technologies such as IEEE 802.16 are currently being amended with relay functionalities.

As of today, QoS-aware bandwidth management/allocation and scheduling have been extensively studied for traditional wireless networks. For IEEE 802.16 based relay networks—to the best of our knowledge—an investigation of these issues is still missing. In this paper, we look on how to design bandwidth management schemes as well as their implementation using scheduling algorithms for the case of relay networks, which enable spatial reuse while supporting QoS.

The contributions of this paper are as follows.

- We discuss strategies and design issues for bandwidth management in IEEE 802.16 relay networks.
- We design QoS-aware bandwidth management algorithms. In particular, we investigate algorithms for building collision sets, which allow for efficient realization of spatial reuse in relay networks.
- We design scheduling algorithms together with admission control strategies suitable for relay networks.
- As a proof of concept, we design and implement the above algorithms and strategies for IEEE 802.16j relay networks. We investigate the performance of the schemes by means of a simulation study.

II. Background and Related Work

We next provide a brief introduction to the relay mode of IEEE 802.16 with an emphasis on the frame format and frame division between base station (BS) and relay station (RS). In contrast to purely PHY-layer relays, the 802.16j extension specifies a non-transparent, multihop relay mode operating on PHY/MAC layer. Then we discuss related work in the field of bandwidth management.

A. System

We assume the relay network to follow the specification of the IEEE 802.16 standard (see [2]), which defines a TDMA-based, connection oriented MAC. We assume that a relay network comprises one BS and one or multiple RSs and Subscriber Stations (SS) (see Fig. 1 for a simple relay topology). Appropriate frame division techniques allow for separate, non-interfering downlink and uplink subframes for either centralized or distributed scheduling as discussed below. All stations are assumed to have successfully entered the network and maintain (basic) management connections.

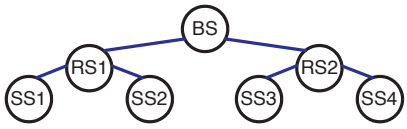


Fig. 1. Simple relay topology

B. Multihop Relaying in IEEE 802.16j

To enable the RS to manage certain parts of the bandwidth, a modification to the frame division of the 802.16 standard is necessary. In [3] and [4] the authors propose two strategies for dividing the frame between BS and RS; a performance analysis of both can be found in [5].

Strategy one, which has been designed for centralized scheduling, is shown in Fig. 2. The BS acts as coordinator and employs a centralized schedule, which covers the relays as well. As a result, the RS is dynamically assigned designated parts of the Downlink (DL) and Uplink (UL) subframes by

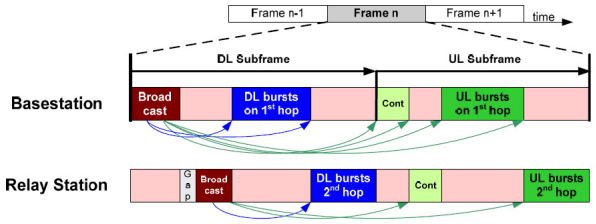


Fig. 2. Frame division with centralized scheduling (Source: [3])

the BS. A second strategy for frame division is also presented in the latter work which enables distributed scheduling. Here the relay frame is embedded into the UL subframe. BS and RS communicate using the regular UL/DL subframes, while communication between RS and SS are carried out in the dedicated relay subframe. From the perspective of a SS, the RS appears as a BS. Hence, the relay subframe should start at a fixed offset to allow for periodicity; moreover, it should not be inserted at the end of the UL-subframe to allow for a variable size. For the remainder of this work, we assume strategy one for the frame division. Another frame division strategy proposed by the authors in [6] uses spatial reuse beyond three tiers in the relay network. This approach is, however, not flexible and not applicable for smaller trees and it is not considered here.

C. Spatial Reuse

Spatial reuse can increase the capacity of the network in addition to increasing its coverage. If we assume a single frequency relay network, the depth of the relay tree needs to be large enough, however, to facilitate spatial reuse. Again appropriate frame division strategies have to be employed and implemented to allow for spatial reuse. We build on the centralized frame division algorithm from [3] and [4], since it provides for maximum flexibility of the schedule within each frame. We extend the frame structure with one additional dimension to support OFDMA and each downlink and uplink burst can be allocated to more than one station to permit spatial reuse.

D. Bandwidth Management and Scheduling

Bandwidth management, admission control (AC) and scheduling are well studied for the point-to-multipoint mode of the 802.16 standard (See e.g. [7], [8], [9] and [10]). According to the standard, the AC decision are performed by the BS. Each SS manages its service flows by using 802.16's DSA (add), DSC (change) and DSD (delete) control messages. The BS is responsible for admitting and granting flows respecting the corresponding QoS classes (if possible). The BS then periodically (each frame) calculates and disseminates the schedule using the DL/UL-map.

For single hop networks, there exist appropriate QoS-aware scheduling mechanisms. However, for relay networks with spatial reuse, such mechanisms are not well studied.

We propose to build on [7], which describes an efficient two-level QoS-aware scheduling algorithm tailored for IEEE 802.16 networks. The first layer uses a Deficit Fair Priority Queue (DFPQ)[1] to decide which QoS class should be granted next. UGS bandwidth is allocated already during admission, for the remaining classes, the priorities are as follow: $rtPS > nrtPS > BE$. $Downlink$ flows are prioritized over $Uplink$ flows.

The second layer of the scheduling algorithm from [7] uses different types of priority queues to match service flow type: Earliest Deadline First (EDF) for the rtPS queues, Weighted Fair Queue (WFQ) for the nrtPS queues and Round Robin (RR) for the BE queues. However, in its current form, the algorithms from [7] cannot be directly applied to relay networks since multihop operation as well as spatial reuse are not supported.

Admission control and scheduling in relay networks are also investigated in [8]. The authors propose algorithms for scheduling multihop transmissions based on the deadline of the packets. The algorithms base on an even-odd labelling scheme that operates on the links of the scheduling tree. As a result, it cannot easily be implemented in realistic networks following the IEEE 802.16j specification. Still, the basic idea of [8] is of interest for our work, which aims to optimally exploit the network topology using spatial reuse.

Other related work for admission control and scheduling within IEEE 802.16 networks can be found in Refs. [9], [10] and [11]. Since this work is not directly beneficial for our intended scenario, we only describe it briefly: [9] studies admission control algorithms for the PMP mode as well as strategies for bandwidth estimation of the rtPS and nrtPS flows for minimizing delays and avoiding unneeded bandwidth reservations. [10] discusses an AC strategy for distributed scheduling with a long term planning of the network resources. To cut down the control information overhead, in [11] the possibility of minimizing the control information overhead by using aggregation and concatenation of the connections between the relay station and the base station is discussed.

III. Algorithms for Bandwidth Management and Scheduling in Relay Networks

We next investigate and design bandwidth allocation and scheduling algorithms for IEEE 802.16 based relay networks. Design goals are to maximize the throughput in the relay network by supporting spatial reuse while supporting strict QoS requirements. This necessitates mechanisms to allow for proper reuse of the wireless channel, i.e. avoidance of interference by appropriate bandwidth allocation to slots. Moreover, to optimize the achievable delay, the algorithms needs to ensure the scheduling of the individual slot reservations, such that

[1]DFPQ is based on the Deficit Weighted Round Robin (DWRR) algorithm. At each step the algorithm selects the next queue and allocates some slots for the bandwidth requests on the top of the queue. The number of bits to be granted from the selected queue is based on the sum of the maximum admitted bandwidth for all service flows, belonging to that queue. This way the algorithm gives a chance of transmission and prevents the starvation of the low priority flows.

grants for one hop of a connection precede the grants for the following hop.

The fundament for our bandwidth management algorithms is to obtain—based on the topology of the relay network—the set of links that can be successfully scheduled simultaneously. We next develop algorithms to obtain this set by obtaining the collision sets (*collSets*) in the relay network. The designed collision set builder (CSB) analyzes the network topology and builds on algorithms to identify so-called activation sets (*actSets*) and interference graphs (*intfrGraph*), which contain the links that can be simultaneously activated or cause interference, respectively.

Subsequently we design Admission Control (AC) schemes to determine, if new service flows can be admitted depending on the current network load. Finally, we design scheduling schemes to implement the above strategies and to grant adequate transmission slots for fulfilling the requested QoS. An overview of the developed system architecture is shown in Fig. 3.

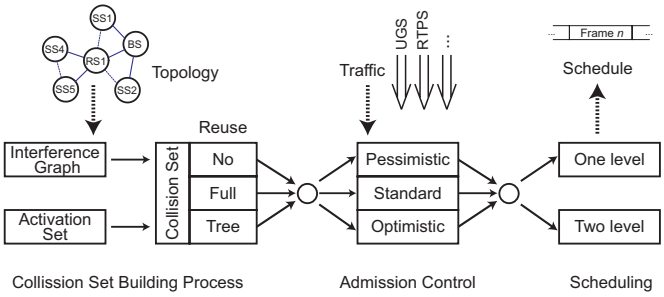


Fig. 3. System overview

A. Interference Graph, Activation Sets and Collision Sets

We define the interference graphs as well as the activation and collision sets as follows.

The ***intfrGraph*** contains the links of the topology as vertexes. Every edge in this graph signifies interference between the two links in the topology, which is represented by the connected vertexes. In our sample topology, a part of the uplink *intfrGraph* is defined by the edges from vertexes SS1 → RS1, SS2 → RS1 and RS2 → BS to vertex RS1 → BS. The CSB algorithm uses the *intfrGraph*, together with the *actSets* to build the collision sets of the topology.

actSets contain all possible sets of links which can be simultaneously used for transmissions. In our sample topology, the transmissions BS → RS2 and RS1 → SS1 can take place simultaneously. The scheduler builds on the identified *actSets*, by assigning the same slots in a frame for multiple non-interfering links.

collSets contain all possible sets of links, for which no two links can reuse the same slot simultaneously. For the scheduler, the sum of the given grants for all links in a *collSet* should never exceed the number of slots available in the subframe. In our simple topology links RS1 → SS1, RS1 → SS2 and BS → RS1 form one *collSet*. The AC and the scheduling utilize the *collSets* in order to determine the maximum amount of data that can be transmitted in the topology for the different links for every frame or part of the frame.

B. Collision Set Building Process

Algorithms to obtain the *actSets*, *intfrGraph* and *collSets* form the basis of our scheme and are detailed in the following for the cases of no reuse, full reuse or tree-based reuse permitted.

1) CSB with No Reuse Permitted (NR): The basic bandwidth allocation scheme of IEEE 802.16 does not permit reuse. We implement this scheme to act as a baseline for our algorithms, the corresponding CSB treats every pair of links as interfering. The number of *actSets* equals the total number of links, each *actSet* contains one link only. The *intfrGraph* is fully connected and exactly one *collSet* holding all links is defined. The *actSets* are ordered such that the scheduler respects the order of hops in multihop connections (1^{st} hop subframe before 2^{nd} hop subframe, etc.).

2) CSB with Full Reuse Permitted (FR): CSB-FR has been designed to provide maximum throughput by optimizing the spatial reuse of slots. The *actSets* are built by recursively adding non-interfering links. The result sets are filtered for duplicates and any subsets are removed. A con of CSB-FR is that a correct ordering of the grants along multiple hops cannot be achieved for the entire network. This can lead to increased delays and a waste of granted resources in case that queues of the 2^{nd}, 3^{rd}, etc. hop do not yet hold data after a transmission commenced. Once the queues are populated, this effect should be minimal[2].

Algorithm 1 shows the pseudocode for obtaining the activation sets with full reuse permitted; Algorithm 2 shows the pseudocode for building the corresponding interference graph and Algorithm 3 shows the pseudocode for obtaining the collision set.

For our algorithms, we define the following variables.

- $Nodes = (node_1, ..., node_n)$: the set of nodes.
- $Links = (link_1, ..., link_n)$: the set of links.
- $actSets = (actSet_1, ..., actSet_n)$: the activation sets.
- $intfrGraph$: the interference graph.
- $collSets = (collSet_1, ..., collSet_n)$: the collision sets.

Table I shows the *actSet* and Table II shows the *collSets* obtained using the above algorithms for full reuse.

3) CSB with Tree-based Reuse Permitted (TBR): To minimize the amount of grants, for which there is no data to be transmitted, we propose to build *actSet* based on the link tier. The sets contain only links with the same tier of the tree, counted as number of hops from the BS. CSB-TBR does not maximize the reuse, but optimize efficiency and delay, since it allows to order the *actSets* such that grants are given first for the 1^{st} hop, next for the 2^{nd} hop, etc.

[2]According to draft 802.16j [2], only the service flow endpoint can request bandwidth for a flow. With non-ordered grants, this might lead to data getting trapped in a queue, after the endpoint stops sending data and requesting bandwidth for it. Allowing RSs to generate requests for the remaining data solves this issue.

Algorithm 1 Building *actSets* for Full Reuse

```
actSets = newList()                                   ▷ Create list of activation sets
for link_i ∈ Links do                                 ▷ Start building activation sets with each link
    addedLinks = newList()                            ▷ Create list with links in activation set
    addedLinks.add(link_i)
    restLinks = newList()                             ▷ Create list with all other links
    for link_j ∈ Links do
        if link_i != link_j then
            restLinks.add(link_j)
        end if
    end for
    removeInterferingLinks(link, restLinks)
    addNextLinkToSet(addedLinks, restLinks)
end for
function ADDNEXTLINKTOSET(addedLinks, restLinks)
                                                      ▷ Adds next possible link to the activation set
    if restLinks.isEmpty() then
                                                      ▷ If no more links to be added, this is a activation set
        if !actSets.contains(addedLinks) then
                                                      ▷ Add to the list, only if not present already
            actSets.add(addedLinks)
        end if
    else
        for link_k ∈ restLinks do                     ▷ Add each of the remaining links
            addedLinks.add(link_k)
            restLinksNew = restLinks
            removeInterferingLinks(link_k, restLinksNew)
                                                      ▷ Remove all interfering links
            addNextLinkToSet(addedLinks, restLinksNew)
                                                      ▷ Add next possible link to the activation set
            addedLinks.remove(link_k)
        end for
    end if
end function
```

Algorithm 4 shows the pseudocode for obtaining the activation sets with tree-based reuse permitted, Table III shows the activation sets for the topology in Fig. 1 in the tree-based reuse scenario. Algorithm 5 shows the pseudocode for building the corresponding interference graph.

C. Admission Control (AC)

We designed three AC strategies to complement the above algorithms. The bandwidth manager executes the AC algorithm on base of all stored DSA-REQ messages, the available subframe size and the set of the already admitted service flows. All schemes check whether the BW-requirements of all service flows for all hops fit into the subframes for every collision domain; thus the performance of the AC schemes directly depends on the quality of the CSB.

- **Pessimistic Admission Control (PAC)** checks service flows using the *maximum* BW-requirements: $bw_j = max_j$, which assures that the network is never overloaded.
- **Standard Admission Control (SAC)** checks service flows using the *average* BW-requirements: $bw_j = (min_j + max_j)/2$, which results in a saturated network in average.
- **Optimistic Admission Control (OAC)** checks service flows using the *minimum* BW-requirements: $bw_j = min_j$, which allows the network to be over-saturated.

Where bw_j is the bandwidth request of flow j with min_j and max_j bandwidth requirements.

For the AC decision the following condition must be applied for the service flows for the set of links for every *collSet* (or for all the links in the topology for the no reuse case):

$$\sum_{i=1}^{num_{links}} \sum_{j=1}^{num_{flows}} partof(i, path_j).bw_j \leq size_{subframe}$$

TABLE I
actSets FOR FULL REUSE FOR TOPOLOGY IN FIG. 1

FR-*actSet*	**Downlink** (**Uplink** is vice versa)
1	BS → RS1, RS2 → SS3
2	BS → RS1, RS2 → SS4
3	BS → RS2, RS1 → SS1
4	BS → RS2, RS1 → SS2
5	RS1 → SS1, RS2 → SS3
6	RS1 → SS1, RS2 → SS4
7	RS1 → SS2, RS2 → SS3
8	RS1 → SS2, RS2 → SS4

TABLE II
collSets FOR FULL REUSE FOR TOPOLOGY IN FIG. 1

FR-*collSet*	**Downlink** (**Uplink** is vice versa)
1	BS → RS1, BS → RS2
2	BS → RS1, RS1 → SS1, RS1 → SS2
3	BS → RS2, RS2 → SS3, RS2 → SS4

Algorithm 2 Building *intfrGraph* for Full Reuse

```
intfrGraph = newList()                                ▷ Create the intfrGraph
for link_i ∈ Links do                                 ▷ Add all links to the intfrGraph
    for link_j ∈ Links do
        if interferes(link_i, link_j) then
                                                      ▷ Add a link in the intfrGraph, if the two links interfere
            intfrGraph.add(link_i, link_j)
        end if
    end for
end for
```

Algorithm 3 Building *collSets* using CSB-FR

```
collSets = newList()                                  ▷ Create the list of collision domains
for link_i ∈ Links do                                 ▷ Start building collSets with each link
    addedLinks = newList()
                                                      ▷ Create a list with links in the collision domain
    addedLinks.add(link_i)
    restLinks = newList()                             ▷ Create a list with all other links
    for link_j ∈ Links do
        if link_i != link_j then
            restLinks.add(link_j)
        end if
    end for
    addNextLinkTocollSet(addedLinks, restLinks)
end for
function ADDNEXTLINKTOCOLLSET(addedLinks, restLinks)
                                                      ▷ Adds next possible link to the collision domain
    if restLinks.isEmpty() then
                                                      ▷ If no more links to be added, this is a activation set
        if !collSets.contains(addedLinks) then
                                                      ▷ Add to the list, only if not present already
            collSets.add(addedLinks)
        end if
    else
        for link_k ∈ restLinks do                     ▷ Add each of the remaining links
            addedLinks.add(link_k)
            restLinksNew = restLinks
            removeReusingLinks(link_k, restLinksNew)
                                                      ▷ Remove all possibly reusing links
            addNextLinkToSet(addedLinks, restLinksNew)
                                                      ▷ Add next possible link to the collision domain
            addedLinks.remove(link_k)
        end for
    end if
end function
```

Algorithm 4 Building *actSets* for Tree-based Reuse

```
actSets = newList()                                  ▷ Create the list of activation sets
linksPerLevel = assignLevels(Links)
                                                     ▷ Assign a tree level to each link
downlinkLinksPerLevel = getDownlinkLinks(linksPerLevel)
                                                     ▷ Get only the downlink links
uplinkLinksPerLevel = getUplinkLinks(linksPerLevel)
                                                     ▷ Get only the uplink links
for link_i ∈ Links do                                ▷ Start building activation sets with each link
    addedLinks = newList()                           ▷ Create a list with links in the activation set
    addedLinks.add(link_i)
    restLinks = newList()                            ▷ Create a list with all other links
    for level ∈ levels do                            ▷ Cycle starting from the lowest level
        addNextLinkToSet(emptySet, downlinkLinksPerLevel
        (level))                                     ▷ Add the possible activation sets for that level
    end for
    for level ∈ levels.reverse() do                  ▷ Cycle from the highest level
        addNextLinkToSet(emptySet, uplinkLinksPerLevel
        (level))                                     ▷ Add the possible activation sets for that level
    end for
end for
```

Algorithm 5 Building *intfrGraph* for Tree-based Reuse

```
linksPerLevel = assignLevels(Links)
                                                     ▷ Assign a tree level to each link
intfrGraph = newList()                               ▷ Create the intfrGraph
for link_i ∈ Links do                                ▷ Add all links to the intfrGraph
    for link_j ∈ Links do
        if interferes(link_i, link_j)orlevel_link_i ! = level_link_j then
▷ Add a link to intfrGraph, if the two links interfere or have different tree levels
            intfrGraph.add(link_i, link_j)
        end if
    end for
end for
```

Where num_{links} is the # of links in the *collSet*, num_{flows} is the # of service flows and $partof(i, path_j)$ is a function, which returns 1 if link i is part of the path of flow j. $size_{subframe}$ is the size of the subframe, in which the service flows have to be scheduled.

D. Scheduling

We develop a relay-aware *simple* as well as a *priority-based* (two-level) scheduling scheme to implement the obtained bandwidth allocation in an IEEE 802.16 compatible fashion. Fig. 4 shows the flow of the scheduling algorithms inside the bandwidth manager, starting from the received bandwidth requests and ending with issuing the schedule (how many and which slots to grant for the individual flows) to all RSs and SSs using the DL/UL-map.

1) Simple Scheduling (SIS): The scheduler determines, which BW-requests can be granted. The algorithm initially grants UGS and Basic (control) flows their fixed amount of slots; for all other service flows the minimum of the admitted or requested data is granted, unless the node has requested less in this frame. In case of a faulty AC, when even the minimum admitted bandwidth is not feasible, all grants are decreased, starting from the BE flows, followed by nrtPS, ertPS, rtPS and UGS.

After granting of the minimum admitted bandwidth, some slots in the subframe might be available, unless the CSB has accepted too many flows. These available slots are assigned proportionally to the flows with requests exceeding the admitted bandwidth (grants up to the maximum admitted bandwidth; if still slots are available grants up to max_j). To ensure the feasibility of the schedule, the constraints for all the links in every *collSet* are taken into account:

$$\sum_{i=1}^{num_{links}} \sum_{j=1}^{num_{flows}} partof(i, path_j).grant_j \leq size_{subframe}$$

With num_{links} being the number of links in the *collSet*, num_{flows} being the number of service flows, $grant_j, grant_i$ being the grant for flow j or link i and $size_{subframe}$ being the size of the subframe, in which the service flows are scheduled.

The algorithm has to determine how to distribute the grants among the activation sets, so that all grants can fit within the subframe. A set of inequalities containing one inequality for every link is solved:

$$\sum_{j=1}^{num_{act\_sets}} partof(i, set_j).X_j \geq grant_i$$

Where num_{act_sets} is the number of *actSets*, i is the link, X_j is the unknown grant for the activation set j, $partof(i, set_j)$ is a function, which returns 1 if the link i is in the activation set j.

In addition an inequality for the subframe size is added to the system:

$$\sum_{j=1}^{num_{reuse\_sets}} X_j \leq size_{subframe}$$

For the simple topology from Fig. 1, when using CSB-FR, the system of inequalities is as follows:

$$1.X_1 + 1.X_2 + 0.X_3 + 0.X_4 + 0.X_5 + 0.X_6 + 0.X_7 + 0.X_8 \geq grant_1$$
$$0.X_1 + 0.X_2 + 1.X_3 + 1.X_4 + 0.X_5 + 0.X_6 + 0.X_7 + 0.X_8 \geq grant_2$$
$$0.X_1 + 0.X_2 + 1.X_3 + 0.X_4 + 1.X_5 + 1.X_6 + 0.X_7 + 0.X_8 \geq grant_3$$
$$0.X_1 + 0.X_2 + 0.X_3 + 1.X_4 + 0.X_5 + 0.X_6 + 1.X_7 + 1.X_8 \geq grant_4$$
$$1.X_1 + 0.X_2 + 0.X_3 + 0.X_4 + 1.X_5 + 0.X_6 + 1.X_7 + 0.X_8 \geq grant_5$$
$$0.X_1 + 1.X_2 + 0.X_3 + 0.X_4 + 0.X_5 + 1.X_6 + 0.X_7 + 1.X_8 \geq grant_6$$
$$1.X_1 + 1.X_2 + 1.X_3 + 1.X_4 + 1.X_5 + 1.X_6 + 1.X_7 + 1.X_8 \leq size_{subframe}$$

In this system each row represents a link from the topology and each column represents an *actSet*. For example, the 1^{st} row represents the link BS → RS1, which is included only in the first two FR-*actSets* from Table I and therefore only these columns contain the value 1 as a coefficient. The 1^{st} column represents the second FR-*actSet* from Table I. It includes the links BS → RS1 and RS2 → SS4. Therefore, the coefficients for the 1^{st} column for rows 1 and 5 (link RS2 → SS4) have a value of 1. The simplex method, is used to solve the system

TABLE III
actSets FOR TREE-BASED REUSE FOR TOPOLOGY IN FIG. 1

TBR-*actSet*	**Downlink** (**Uplink** is vice versa)
1	BS → RS1
2	BS → RS2
3	RS1 → SS1, RS2 → SS3
4	RS1 → SS1, RS2 → SS4
5	RS1 → SS2, RS2 → SS3
6	RS1 → SS2, RS2 → SS4

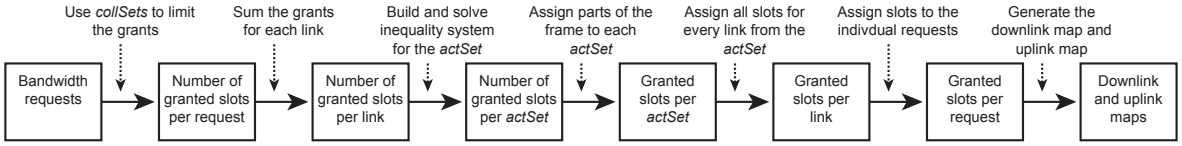


Fig. 4. Operational flow of the scheduling algorithms

of inequalities, using the following objecting function:

$$min \sum_{j=1}^{num_{reuse\_sets}} grant_j$$

The grants for each *actSet* are rounded down to an integer number of slots. Next, a packing algorithm assigns the grants to the individual service flows. It obtains the total number of slots granted to each flow and assigns specific subchannels and slots to it. It cycles through the *actSets* in the order they were created by the CSB. Every *actSet* is given the next part of the subframe, and that part is granted to the first waiting requests for all the links in the *actSet*.

2) Priority-based Two-level Scheduling (TLS): The TLS algorithm maintains different queues for the requests from different QoS classes (rtPS, ertPS, nrtPS and BE). The minimum admitted for each flow, as well as all flows from type UGS or Basic are granted initially, and the corresponding requests are removed from the queues. TLS maintains a history of grants for each connection queue and for each service type. In level one of the algorithm a queue is selected; level two selects the first request from that queue, which is granted if possible. If infeasible, the service flow is removed from the queue for the current frame. The selection of a queue is based on the lowest satisfaction index, calculated as follows:

$$s_{type} = \frac{w_{type} \cdot \sum_{i=1}^{history} grant_{i,type}}{history.min_{type}}$$

Where s_{type} is the satisfaction index and w_{type} is the priority weight for the queue, $history$ is the number of frames stored as history, $grant_{i,type}$ is the grant for the frame i from the history, given to flows from this queue and min_{type} is the min. admitted bandwidth for all flows belonging to the queue for one frame.

The priorities of the queues are weighted as follows: $w_{rtPS} > w_{ertPS} > w_{nrtPS} > w_{BE}$. The queues for the flows of type rtPS and ertPS are sorted based on the deadline of the packets, starting with the earliest deadline: $deadline_{request} = time_{request} + max\_delay_{service\_flow}$, where $deadline_{request}$ is the deadline for the request, $time_{request}$ is the generation time of the request and $max\_delay_{service\_flow}$ is the maximum allowed end-to-end delay for the packets from the service flow.

The queues for the flows of type nrtPS and BE are sorted based on the lowest satisfaction index within the queue, calculated as follows:

$$s_{flow} = \frac{\sum_{i=1}^{history} grant_{i,flow}}{history.min_{flow}}$$

Where $grant_{i,flow}$ is the grant for frame i from the history, given to this flow and min_{flow} is the min. admitted bandwidth for the flow for one frame.

Analog to the SIS algorithm, TLS finally obtains the grants for every selected service flow, calculates the grants per link of the topology, determines the grants for every *actSet* and creates the DL/UL-maps.

IV. Performance Evaluation

In this section we analyze the performance of the developed algorithms. We investigate the trade-offs of the different schemes w.r.t. QoS provisioning, throughput and network utilization, while incorporating spatial reuse in relay networks. Existing work mainly focusses on trivial relay topologies, i.e., one-hop relaying. We have chosen the non-trivial random topology shown in Fig. 5. In particular, this topology enables us to study the performance of our algorithms in a topology where the BS and RSs serve SSs that are between one and three hops away from the BS.

A. Experimental Design

In the setting of an IEEE 802.16j relay network, we investigate combinations of the following algorithms and parameters.

- **Collision Set Builder (CSB)** strategy: no reuse (*NR*), tree-based (*TBR*), or full reuse (*FR*).
- **Admission Control (AC)** strategy: pessimistic (*PAC*), standard (*SAC*), or optimistic AC (*OAC*).
- **Scheduling (S)** strategy: simple (*SIS*) or two-layer scheduling (*TLS*).
- **Workload (WL)**: moderate (*MWL*) or high workload (*HWL*).

We follow a full factorial experimental design with 15 replications for each experiment. The results are presented as average values including the 95% confidence intervals. For

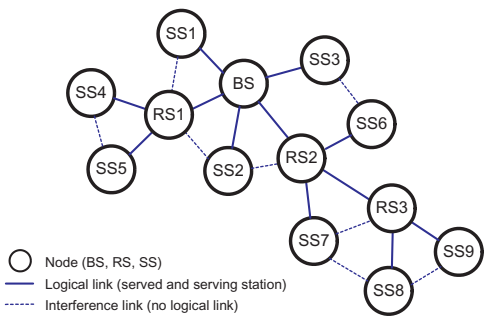


Fig. 5. Simulation topology

TABLE IV
IEEE 802.16J SIMULATION PARAMETERS

Parameter	Downlink	Uplink
Frame duration	$20ms$	
Frames per second	50	
Number of frames	3000	
Channel bandwidth	ETSI, $14MHz$	
Access scheme	OFDMA	
Symbol duration	$128\mu s + 8\mu s = 136\mu s$	
Symbols per frame	147	
Fraction of subframe	60%	40%
Symbols per subframe	90	57
Symbols per slot	2	3
Slots per subframe	45	19
Number of subcarriers	48	
Number of subchannels	32	
Bits per slot (QPSK 1/2)	96	144
Allocation matrix	45×32	19×32
Bits per frame (QPSK 1/2)	138240	87552
Bits per second (QPSK 1/2)	6912000	4377600
Bits per second total (QPSK 1/2)	11289600	

TABLE V
WORKLOAD/NETWORK TRAFFIC PARAMETERS

Type	QoS Class	Bandwidth in [bytes/frame] Minimum	Maximum	Average
CBR	UGS	variable, uniformly distributed in 250 – 750	250 – 750	250 – 750
VBR	rtPS	variable, uniformly distributed in 250 – 500	500 – 750	375 – 625
Gaming	rtPS	6	15	6
VoIP	rtPS	7	7	7
VBR	nrtPS	variable, uniformly distributed in 250 – 500	500 – 750	375 – 625
FTP	nrtPS	20	50	35
nrtPS	nrtPS	5	10	8
Real trace	nrtPS	200	550	425
VBR	BE	0	variable, unif. dist. in 0 – 750	0 – 375
HTTP	BE	0	35	18

each experiment, we simulated 3000 frames, i.e. a duration of $60s$. Table IV shows the PHY and MAC parameters of the employed IEEE 802.16 technology; Table V shows the workload/traffic parameters used in the performed simulation study, where we aimed at creating a realistic traffic mix including all service classes of IEEE 802.16.

B. Analysis of Results

The effectiveness and efficiency of the developed algorithms are evaluated as follows. We have checked the established *collSets* and *actSets* of the CSB algorithms, which operated according to the specifications. Using the metrics of throughput and network utilization the efficiency of the combination of CSB, AC and scheduling is analyzed. We emphasize on the effects of the scheduling/granting algorithms, since these perform the aggregation of multiple requests and have to deal with spatial reuse while—at the same time—these have to fulfill the specifications of IEEE 802.16, i.e., grants are issued on a station/link level.

1) Network Utilization and Throughput: Fig. 6 shows the achievable network utilization given in number of admitted flows for the different AC schemes. We show the results for the highly loaded network, because in the moderately loaded scenario the choice of the AC scheme is not decisive, since all flows can be admitted. In the HWL scenario, we see an increase in admitted flows over PAC for SAC and a further increase for OAC. We also notice a significant difference for the number of admitted flows for the different reuse schemes. With TBR and FR significantly more flows can be admitted, due to spatial reuse.

Fig. 7 shows the obtained throughput per frame (a) as well as per link (b) for the most demanding traffic setup (OAC, HWL). The effects of the different reuse schemes are clearly visible. FR is able to sustain a much higher data rate for both scheduling schemes compared to TBR and NR. For TBR, we notice a difference between SIS and TBS in favor of the SIS scheme, while for NR the different scheduling schemes are roughly on par again.

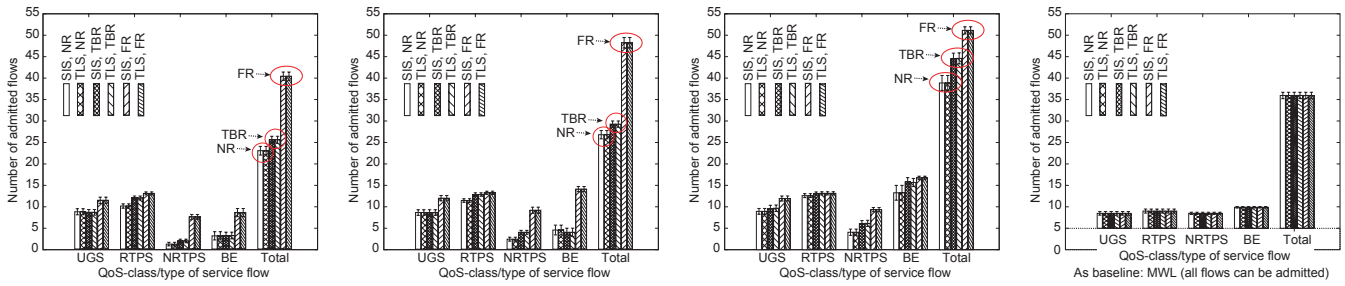
Fig. 7(b) clearly indicates the benefits of the FR scheme. The most stressed links (i.e. the links from the BS to the one-hop RSs as well as the links between RSs) achieve a much better throughput. The data rate for the individual SSs is distributed in a fair manner.

Fig. 7(c) illustrates the percentage of Grant messages that do not lead to a reservation that can be used by the system. For all schemes the obtained values of in average below 1% for FR and 0.5% for TBR of non-successful grants can be considered to be very good. The results confirm that TBR can mitigate unsuccessful grants that are caused by misordering of the individual hops in the topology.

2) Quality of Service: Fig. 8 shows the obtained average delay for our experiment. We clearly see that the TBR scheme (see Fig. 8(b)) can sustain the low delay of the NR scheme (see Fig. 8(a)), despite the fact that reuse takes place. The delay of the FR scheme (see Fig.8(c)) increases with increasing number of hops, which is a result of the non-optimal ordering of the grants. Fig. 8(d) shows that even for a saturated network, the delays for the QoS classes UGS, RTPS, NRTPS are contained in the specified delay-envelope. The (unbounded) delay of the BE class in contrast increases due to full queues and only minimal service for this class.

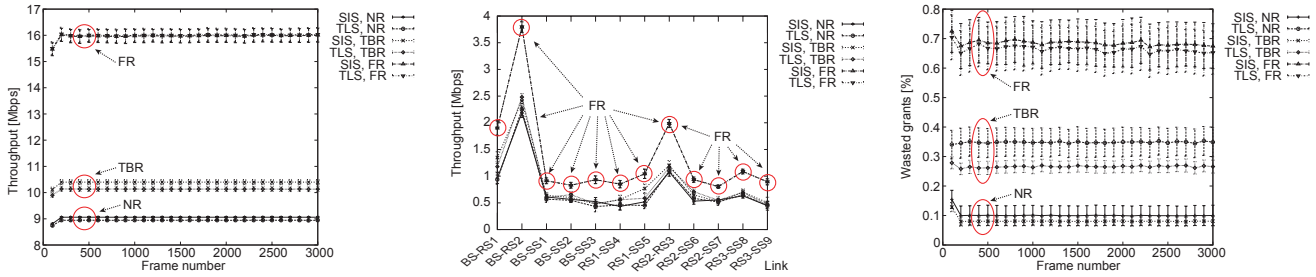
V. CONCLUSION

We have investigated and designed QoS-aware bandwidth allocation and scheduling algorithms for IEEE 802.16 based relay networks, which support for spatial reuse to additionally improve the throughput of the network. By systematically evaluating various design choices, implementing proof-of-concept algorithms and by means of a thorough simulation study, we analysed the trade-offs involved in bandwidth management for IEEE 802.16 relay networks. Our insights and algorithms allow for future research in WiMAX relay networks.



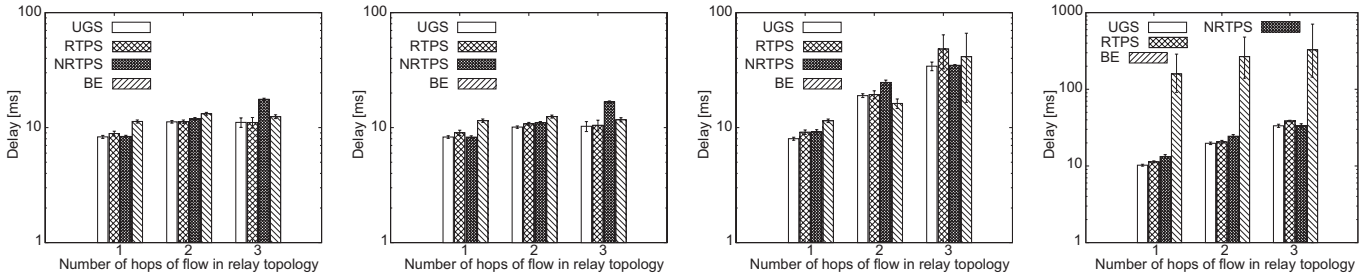
(a) Pessimistic Admission Control (b) Standard Admission Control (c) Optimistic Admission Control (d) Optimistic Admission Control (Moderate Workload)

Fig. 6. Number of admitted flows to measure the utilization of the relay network. We show the effects of admission control for all combinations of scheduling (SIS/TLS) and reuse (NR,TBR,FR) algorithms for the high workload scenario (HWL) in (a) - (c). As a baseline we show the moderate workload scenario (MWL) in combination with optimistic admission control in (d).



(a) Per-frame Throughput (b) Per-link Throughput (c) Per-frame Unused Grants

Fig. 7. Throughput per frame (a) and per link (b); number of unused Grants per frame (c) are shown for the combination of optimistic admission control (OAC) and high workload (HWL) for all combinations of scheduling (SIS/TS) and reuse (NR,TBR,FR).



(a) No Reuse (b) Tree-based Reuse (c) Full Reuse (d) Full Reuse (High Workload)

Fig. 8. Per-hop delays for the four different QoS classes (UGS, RTPS, NRTPS, BE) for the standard admission control (SAC) for moderate workload (MWL) are shown in (a) - (c). In (d) we show full reuse in in combination with SAC and HWL.

REFERENCES

[1] B. Walke, S. Mangold, and L. Berlemann, *IEEE 802 Wireless Systems: Protocols, Multi-Hop Mesh/Relaying, Performance and Spectrum Coexistence*. Wiley, 2006.

[2] "Baseline document for draft standard for local and metropolitan area networks part 16: Air interface for fixed and mobile broadband wireless access systems multihop relay specification," 2007.

[3] C. Hoymann and K. Klagges, "MAC Frame Concepts to Support Multihop Communication in IEEE 802.16 Networks," in *Proc. of 16th WWRF '06, Shanghai, China*, April 2006.

[4] C. Hoymann, K. Klagges, and M. Schinnenburg, "Multihop Communication in Relay Enhanced IEEE 802.16 Networks," in *Proc. of 17th IEEE PIMRC'06*, September 2006, pp. 1–4.

[5] K. Klagges and C. Hoymann, "Performance Analysis of the Subframe Concept in the IEEE 802.16 Network," in *Proc. of 13th EWC'07*, April 2007, p. 6.

[6] Z. Tao, A. Li, K. Teo, and J. Zhang, "Frame Structure Design for IEEE 802.16j Mobile Multihop Relay (MMR) Networks," in *Proc. of IEEE Globecom'07*, November 2007, pp. 4301–4306.

[7] J. Chen, W. Jiao, and H. Wang, "A Service Flow Management Strategy for IEEE 802.16 Broadband Wireless Access Systems in TDD Mode," in *Proc. of IEEE ICC'05*, vol. 5, May 2005, pp. 3422–3426.

[8] D. Ghosh, A. Gupta, and P. Mohapatra, "Admission Control and Interference-Aware Scheduling in Multi-hop WiMAX Networks," in *Proc. of IEEE MASS'07*, October 2007, pp. 1–9.

[9] S. Chandra and A. Sahoo, "An Efficient Call Admission Control for IEEE 802.16 Networks," in *Proc. of 15th IEEE LANMAN'07*, June 2007, pp. 188–193.

[10] F. Hou, P. Ho, J. Cai, X. Shen, C. Hsieh, and A. Chen, "A Novel Distributed Connection Admission Control Scheme for IEEE 802.16 Networks," in *Proc. of 10th ACM MSWIM'07*, 2007, pp. 173–180.

[11] Z. Tao, K. Teo, and J. Zhang, "Aggregation and Concatenation in IEEE 802.16j Mobile Multihop Relay (MMR) Networks," in *Proc. of IEEE Mobile WiMAX Symposium,07*, March 2007, pp. 85–90.