Two-Dimensional Mapping for Wireless OFDMA Systems

Yehuda Ben-Shimol, Itzik Kitroser, and Yefim Dinitz

***Abstract*—The recent Orthogonal Frequency Division Multiple Access (OFDMA) transmission technique is gaining popularity as a preferred technology in the Broadband Wireless Access (BWA) emerging standards. In standards 802.16-2004 and 802.16e, the basic allocation units are comprised of sub-channels and OFDMA time symbols; each sub-channel is a group of sub-carriers, so that all the sub-channels are considered equally adequate to all users. We study the naturally arising new approach of *two-dimensional mapping* of incoming requests into the matrix that represents the system resources, where each allocation is of an *arbitrary multi-rectangular shape* (to the best of our knowledge, this approach has not been discussed elsewhere). We define a cost model and constraints related to practical OFDMA systems, which depend on the spatial shape of the two-dimensional allocation; the main objective function is the spatial efficiency. We show that the arising problem, even in its simplest form, is NP-hard [1]. We present run-time efficient heuristic solutions for various mapping problems, taking into account the above QoS and OFDMA related constraints. In particular, a novel solution for two-dimensional mapping under priority constraints is suggested. Extensive simulations with parameters of real systems were used to investigate the performance of the proposed solutions in terms of throughput, delay and system load. The results show that high throughput can be achieved with relatively simple mapping algorithms. We believe that the proposed two-dimensional mapping approach is prospective, due to its fitness to modern standards.**

***Index Terms*—IEEE802.16, mapping, OFDMA, QoS, resource allocation, scheduling, WiMAX.**

I. Introduction

THE OFDMA transmission technique is gaining popularity as a preferred technology in the Broadband Wireless Access (BWA) emerging standards. The demand for high data rates, combined with physical requirements such as supporting near- and non-line-of-sight operation, multipath mitigation and operating with fading channels, requires a technology that can efficiently give an appropriate answer. Multi-carrier modulation techniques, specifically OFDM, can efficiently handle multipath propagation and increase robustness against frequency selective fading or narrow-band interference. OFDMA combines the TDMA and FDMA schemes. The time domain is segmented into (groups of) OFDMA symbols and each symbol is segmented into sub-carriers. The number of sub-carriers which are allocated to a single transmitter vary according to the transmitter's needs. The transmission rate on those carriers is set to meet the transmitter's needs and capabilities.

Various schemes of allocating sub-carriers in hybrid TDMA/FDMA and OFDMA systems are given in [2]–[6]. [2], [3] present schemes of adaptive allocation of sub-carriers to different users simultaneously as extension of an OFDM system for frequency selective fading environment based on instantaneous channel parameters per user. In [4] a fair resource allocation scheme for OFDMA system is presented with a general goal of maximizing the throughput subject to the total power constraint regarding the QoS requirements of users. An optimization of long term average performance of subcarrier allocation is discussed in [6].

Other approaches of performing multiple access in OFDMA systems and improving their performance by achieving frequency diversity gain for dispersive fading channels use clustered OFDM techniques. In such schemes, the sub-carriers domain is divided into many groups of contiguous sub-carriers called clusters, and each transmitter accesses several clusters. The allocations schemes with granularity of one sub-carrier can be viewed as a specific case of a clustered schemes with one sub-carrier per cluster. A variant of the clustered OFDM techniques is used in the OFDM and OFDMA modes of [7], [8] where the sub-carriers (from different clusters) are grouped into a logical entity called a *sub-channel*. All sub-channels are *equally adequate* for all transmitters; this is possible since each sub-channel is composed of sub-carriers scattered over the entire frequency band. The system model which is studied is the OFDMA PHY mode defined in the standards: [7], [8] for the OFDMA PHY mode and [9] for the BS3 mode. This mode of operation is considered now by WiMAX vendors as the most relevant solution for practical wireless access systems.

Under the above framework, the resources are represented by a two-dimensional matrix, whose dimensions are sub-channels vs. time symbols; all matrix elements are then equally adequate for the purpose of allocation. We study the naturally arising new approach of *two-dimensional mapping* of incoming requests into the above matrix, where each allocation is of an *arbitrary multi-rectangular shape*. This approach directly corresponds to the PUSC and FUSC modes of [7], [8], [10] and BS3 mode of [9], which are considered now as most promising. To the best of our knowledge, this approach has not been discussed elsewhere.

The IEEE 802.16 standard ([7], [8], [10]), is one of the most popular standards for near future fixed and mobile wireless access technologies. The OFDMA mode in this standard is considered to be the baseline technology for future mobile technologies. Our study uses the model defined in this standard as a baseline and is aligned to it both from the problem definition and from the performance evaluation aspects.

Most publications on resource allocation in OFDMA systems [2]–[6] assumed that sub-carriers are allocated to transmitters according to channel conditions of each transmitter and its power and rate requirements. *This decision is done for each OFDMA time symbol in an on-line manner*. Therefore, the allocation complexity increases when the channel is rapidly changing (such as in mobile environment) since the list of suitable sub-carriers per transmitter may change between

Manuscript received September 21, 2005; revised April 30, 2006.

The authors are with the Department of Communication Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel (e-mail: benshimo@bgumail.bgu.ac.il).

Digital Object Identifier 10.1109/TBC.2006.879937

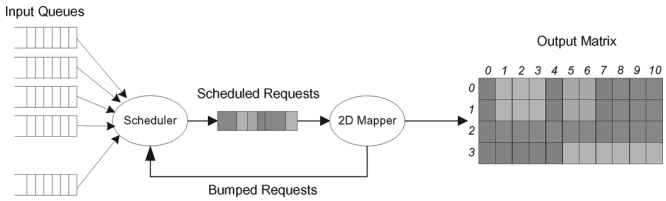

Fig. 1. Example of scheduling and mapping interrelations.

scheduling iterations. Moreover, a closed loop feedback mechanism is required to report on any such change. In addition, the possibility that each user has a subset of suitable sub-carriers for the allocation introduces additional constraints when other transmitters have similar subsets, since then, the overlapping sub-carriers must be shared between the transmitters.

Our work is concerned with the case where the basic allocated resource unit is a combination of a sub-channel and a time unit, and where all sub-channels are *equally adequate* for all transmitters. Hence, all sub-channel are equally adequate for all users; in this case, only average SNR measurements per transmitter are needed to determine its maximum available rate. This mode of operation is considered now by WiMAX vendors as the most relevant solution for practical wireless access systems. This shifts the focus of the problem from on-line selecting of sub-carriers in order to achieve some optimization goal (e.g., max throughput, min power, etc) to selecting slots in a two-dimensional allocation resource matrix. Since the allocated slots are equally adequate for all selected transmitters, there is no need to use bit-selection algorithms to allocate the sub-carriers. This establish a foundation for new algorithmic approaches that deal directly with a two-dimensional mapping problem.

We look at the problem under similar power and rate constraints as in the single sub-carrier allocation problems, but do not limit the availability of specific slots to specific transmitters. In addition, the allocated slots of each request can form an arbitrary multi-rectangular shape, this means that each allocation can be assigned to multiple sub-channels simultaneously. To our best knowledge similar resources allocation approach was not discussed elsewhere for OFDMA systems.

The scheduling task in the studied system is to share the available resources (i.e., time-sub channel slots in a two-dimensional table) among requirements of flows. Each flow may contain packets of variable length. In our model, each flow's requirement is translated directly to required allocation units (slots) using the average SNR measurements per flow. The task of the mapper is to determine number of slots per each served flow. As shown in Fig. 1, the mapping block receives input from the scheduler. We do not force a particular scheduling discipline, and therefore, the scheduling block can use any type of QoS scheduling algorithm. We assume that each allocation has a cost. The number of preambles transmitted before the data on the used sub-channels or size (in bytes) to describe the allocation (for example, UL/DL MAP message entry in [8]) are such costs. In this paper we look on the preamble case as the cost model and therefore, the cost is associated with the spatial shape of the allocation.

In the two-dimensional case, each user's request must be assigned a two-dimensional multi-rectangular allocation structure, which is a subset of the general allocation table (henceforth table). After performing the scheduling task, which results with a number of requested slots D_i per flow i, each served flow, (i.e. $D_i \neq 0$) must be mapped into a set of specific two-dimensional rectangles. Therefore in addition to the scheduling task, a two-dimensional mapping is applied. The scheduler and 2D mapper are illustrated in Fig. 1.

Our study presents novel solutions to a wide set of mapping problems that are related to the prospective OFDMA system. The main approach relies on *raster* based scanning of the allocation matrix, starting from top-left slot and continuing along the time index. We present run-time efficient heuristic solutions for various mapping problems, taking into account QoS and OFDMA related constraints. In particular, a novel solution for two-dimensional mapping under priority constraints is suggested. To the best knowledge of the authors, the raster based mapping approach and its variations were never discussed elsewhere in the context of OFDMA systems. Extensive simulations with parameters of real systems were used to investigate the performance of the proposed solutions in terms of throughput, delay and system load. In addition, a comparison to known approaches is given throughout the text. The results show that high throughput can be achieved with relatively simple mapping algorithms. We believe that the proposed two-dimensional mapping approach is prospective, due to its fitness to modern standards.

In Section II we describe the system model used in our study and simulations. In Section III we give a formal description of the problem. In Section IV we examine the mapping problem for unconstrained scenarios, show theoretical performance boundary for the given cost model and show different approaches to solve the mapping task. In Section V we examine the mapping problem under scenarios with constraints and present three new algorithms. Finally, conclusions and open problems for future research are given in Section VI.

II. System Model

We consider an OFDMA system based on [7], [8] consisted of a Base Stations (BS) and a group of Subscriber Stations (SS). The BS transmits information to all the SSs in its range using a broadcast channel called Downlink channel (DL). All the SSs share the same Uplink channel (UL) to transmit back to the BS. The BS receives requests from SSs to access the UL channel and uses a scheduling algorithm to decide how to grant access in the UL channel. Such decisions are published by the BS by using a special message called a Map message.

Our model allows fragmentation of data across multiple frames, although no additional overhead due to fragmentation is assumed. The system uses adaptive modulation scheme with a Rayleigh fading channel. The modulation and coding rate used are: QPSK-1/2, 16QAM-1/2, 64QAM-2/3 and 64QAM-3/4. To follow the behavior of the system to a high degree, each Forward Error Correction (FEC) block (i.e. one sub-channel for the duration of one OFDMA symbol) contained 48 symbols (i.e., 6 bytes in QPSK-1/2). In each scheduling cycle the request of a connection is normalized to slots according to the instantaneous SNR value that is derived from the channel model. The traffic of each SS is modeled using the 4IPP traffic model (see [11]) scaled to 128 kbps for each connection.

The basic time unit for making the scheduling and mapping decisions is a frame. Between any two consecutive frames the scheduler receives a list of allocation requests and applies the scheduling function on the requests list, including old requests that are still waiting for service. The scheduled requests serve as the input for the mapping algorithm. At the mapping phase a request that cannot be inserted due to lack of allocation space is sent back to the scheduler.

The tested system uses specific values for frame duration which directly implies on the number of symbols in the UL and DL channels. Short frames are more suitable for dynamic changing channels and low delays, but has high relative overhead cost (e.g., preambles, MAP message representation or packing overhead). Long frames have smaller relative overhead cost, but suffer from slow response to changing channels and longer delays (due to longer times between consecutive scheduling and mapping decisions). It is a common practice to make an intelligent compromise and select a frame size which fits the specific system requirements. We tested the performance of the proposed mapping algorithms with medium size frames, having values of 25 OFDMA symbols for both UL and DL channels which represents a realistic frame size for practical 802.16 systems operating with non-mobile users. We assume 32 sub-channels (30 data and 2 ranging), taken explicitly from [8] for 2K FFT size and the FUSC mode.

III. Problem Definition

In the following sections we present basic notations and terms needed for the definition and analysis of the problem.

A. Model and Notations

We define the two-dimensional allocation matrix as allocation table (or *Table*) $T(p,q)$ with $length(T)=p$ and $width(T)=q$ where each item within T is called a *Slot*. Grouping slots from table T in a rectangular spatial shape is called a *Rectangle* which is defined by its dimensions. A $Preamble$ of a rectangle R is denoted by $Preamble(R)$ and consists of its left most slots. The *Effective Area* of a rectangle $(EArea)$ is its area without the area occupied by the preamble. For a rectangle R, $EArea(R) = Area(R) - Area(Preamble(R))$. A Request is a number D of slots to be mapped to table T and an allocation is a set of non overlapping rectangles: $Alloc = \{R_1, \ldots, R_k\}$. The Area of an allocation is defined by: $Area(Alloc) = \sum_{R_i \in Alloc} Area(R_i)$ and the Effective Area (EArea) of an allocation is the sum of effective areas of its rectangles, $EArea(Alloc) = \sum_{R_i \in Alloc} EArea(R_i)$. The cost of an allocation is the number of slots used as a preambles: $cost(Alloc) = \sum_{R_i \in Alloc} Preamble(R_i)$. The $width$ of an allocation $Alloc = \{R_1, \ldots, R_k\}$ is the maximum sum (over the x coordinate) of the widths of its rectangles that has a common x coordinate.

Finally, a Mapping is the process that maps a request D to an allocation $Alloc(D)$, such that $EArea(Alloc(D)) \geq D$.

A Constraint is a limiting rule which is related to a specific request and must be enforced when mapping the request to an allocation. For a request D and an allocation $Alloc(D)$, the following constraints will be considered for the problems defined in Section V. $Max\_Width(D)$: A value M such that $width(Alloc(D)) \leq M$. $Late\_Time(D)$: A value t such that $\max_{R_i \in Alloc(D)}\{X(R_i) + length(R_i)\} \leq t$. $Granularity(D)$: An integer value G that represents the amount of consecutive slots that are used as an allocation unit, that is $\forall R_i \in Alloc(D) : length(R_i) \equiv (1 \mod G)$. $Priority(D)$: An integer value Pr that defines the priority of D where the value 1 defines the highest priority.

B. Basic Problem Definition

The scheduling task in the studied system is to share the available resource (in our case, slots in a two-dimensional table) per requirements of given flows. In practical network scenarios, traffic is bursty and heterogeneous due to mixture of applications, each with its own quality-of-service (QoS) requirements. In such cases the scheduler has to consider both traffic dynamics and QoS requirements to achieve fairness among flows while simultaneously maximizing overall system utilization. We assume that each allocation has a cost, and usually refer the preambles as such cost. Therefore, We associate a cost with the spatial shape of the allocation.

Given an allocation $Alloc(D)$ for a request D, the utilization of an allocation is defined by:

$$\eta(Alloc(D)) = \frac{EArea(Alloc(D))}{Area(Alloc(D))} \tag{1}$$

In the two-dimensional case, each flow's request must be assigned a two-dimensional rectangular allocation structure, which is a subset of the general *allocation table*. After performing the scheduling task, each served flow must be mapped into a specific two-dimensional allocation. Therefore in addition to the scheduling task, a two-dimensional mapping should be applied.

Given a two-dimensional table $T(p,q)$, and a list of requests $L = \{D_1, \ldots, D_n\}$, an allocation $Alloc(D_i) \subseteq T$ is an eligible mapping of request D_i in T if $Alloc(D_i)$ is a Mapping and $\forall i \neq j$, $Alloc(D_i) \cap Alloc(D_j) = \emptyset$.

Given table $T(p,q)$, a set of requests L and a set of allocations $M(T,L) = \{Alloc(D_{k_1}), \ldots, Alloc(D_{k_l})\}$, we define an unused slot $s \notin M(T,L)$. Therefore we can define the efficiency of the Mapping M as the ratio between the effective area allocated by M and the size of T:

$$E(M) = \frac{\sum_{Alloc(D_i) \in M} EArea(Alloc(D_i))}{p \times q} \tag{2}$$

Equation (2) is used to compare between different mapping algorithms under high traffic load scenarios. Due to the two-dimensional nature of the problem, slots can be skipped during a mapping process, thus leaving unused slots in the table. Therefore, the efficiency is defined as a ratio of how many slots in the table were used for *effective* allocations to unused slots and preambles.

The formal definition of the problem follows:

Definition 3.1: Basic Two-Dimensional Allocation Problem (B2DAP): Given a table $T(p,q)$ and a list of requests $L = \{D_1, \ldots, D_l\}$, find set of eligible allocations $M(T,L) = \{Alloc(D_1), \ldots, Alloc(D_l)\}$ that maximizes $E(M)$.

The mapper may fail to map all the requests provided by the scheduler and returns a feedback of such requests which then may be scheduled in the subsequence frame. Therefore, the mapper must be aware of QoS context of requests in order to make rejection decision based on QoS requirements of requests (for example, rejecting request with low priority).

We extend the basic problem to handle constraints using the following definition:

Definition 3.2: Constraints Aware Two-Dimensional Allocation Problem (C2DAP): Given a table $T(p,q)$ and a list of requests $L = \{D_1, \ldots, D_l\}$ where each request may have constraints attached to it, find set of eligible allocations $M(T,L) = \{Alloc(D_1), \ldots, Alloc(D_l)\}$ that maximize $E(M)$ while satisfying all the constraints.

Algorithmic solutions to the *B2DAP* and the *C2DAP* are given in Sections IV and V, respectively. We note that in Section V we present specific solution to the *Priority* constraint, and solutions for the other constraints sets can be found in the extended version of this paper.

In our model, the time unit for making the scheduling and mapping decisions is a frame, therefore the scheduling and mapping tasks are done in an off-line style, which enables the separation between the two tasks. The separation between the mapping algorithm and the scheduling algorithm makes the overall allocation process more flexible. As will be shown later (Section IV-A-2), mapping the packets according to the order determined by the scheduler does not always provide an efficient solution due to the two-dimensional nature of the problem. Therefore, the two-dimensional mapper and the scheduler have a bi-directional connection (see Fig. 1) and backlogging packets may lead to a better solution. In addition, during the mapping process and due to mapping constraints, slots may be left empty and not be allocated to any request. Therefore, for some series of requests, not all the requests can be mapped and will be bumped. The bumped requests are indicated as such by the mapper and will be returned to the scheduler to be served in future scheduling iterations.

B2DAP can be shown to be NP-hard using a reduction from the *BP-SIF* problem and its corresponding decision problem D(*BP-SIF*), which were shown to be NP-hard in [12].

IV. Allocations Without Constraints

The upper bound for the efficiency of any mapping algorithm under the given cost model (i.e., using a preamble) is $(p-1)/p$ for a table with length p. This means that the size of the requests affects the efficiency of the algorithm. For a table $T(p,q)$ the best performance (i.e., highest efficiency) will be received for requests of sizes that are multiple of $(p-1)$ with total sum of $q(p-1)$, since all the requests will be mapped with exactly one preamble per table row. (2).

A. Raster Algorithms

We provide a family of scheduling algorithms to solve the mapping problem. The algorithms are based on a Raster scanning approach, in which the table T is filled row by row, from left to right and from the top to bottom while allocating requests according to a specific criteria.

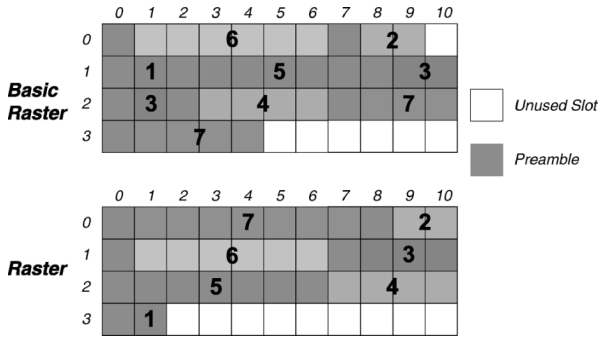

Fig. 2. Example of the Raster and Basic Raster algorithms for the request series: 6,2,1,5,3,4,7 and $T(11,4)$.

1) Basic Raster Algorithm: The Basic Raster algorithm represents a naive approach for solving the *B2DAP* problem and is used as a performance reference base line for the mapping algorithms presented ahead. The algorithm receives a table $T(p,q)$ and a set of requests $L = (D_1, \ldots, D_n$. Table T is scanned row by row, from left to right and from top to bottom and the requests are processed according to the received order. Allocations are done by assigning slots for a request in the scan order. Each assignment begins with a preamble. If the number of slots to be assigned is higher than that of non-scanned slots in the current row, the assignment will continue in the next row, starting with a new preamble. The final allocation for a request D_i is $Alloc(D_i)$. If the number of the required rows per allocation D_i at the current location is greater than the number of remaining rows, then the request is returned to the scheduler by setting $Alloc(D_i) = 0$.

2) Raster Algorithm: The Basic Raster algorithm is simple, but not always efficient since it adds costs that could have been avoided. This efficiency problem is derived from the following observations: (1) Each new row of an allocation adds a cost (i.e., a preamble). This can lead into scenarios in which a different selection of requests allocated during the iteration stage of the algorithm prevents the fragmentation of an allocation between two rows, thus not adding a cost. (2) It is better to fragment a longer allocation than a shorter one, since it results with better utilization factor η. The main reason for this is that the relative cost of a preamble is higher for short requests than for longer ones. The efficiency expression (2) considers both the overhead due to preambles and unused slots. Therefore each η value contributes to the overall efficiency, and each short request will contribute more overhead to the total efficiency calculation. One way to improve the Basic Raster algorithm is to use a greedy approach to select requests to be served, while avoiding splitting allocations between rows as much as possible. The Raster algorithm modifies the Basic Raster algorithm as follows: **Step 1**: Sort requests in descending order of their size. **Step 2**: In each iteration, select the longest request that can be allocated without fragmentation. **Step 3**: If no request can be found in Step 2, select the longest request that can be inserted.

The modification above tries to answer the two problems identified in the Basic Raster algorithm. Fig. 2 presents an example of the Raster vs. the Basic Raster algorithms for the same table and set of requests. As can be seen in the figure,

the effect of sorting and selecting the best request at a time can prevent unnecessary preambles or unused slots in the middle of the allocation. We note here, that the greedy approach prefers specific requests that answer a local greedy criterion, and can potentially lead a starvation of some requests which can be bumped by the *Raster* algorithm. This problem is solved when the priority constraint is used (see Section V-B) by assuring that the decision to insert a request considers its priority in addition to overall efficiency goal.

For a given table $T(p,q)$, we can see that the value of p can affect the performance of Raster based algorithms, since it determines how many preambles will be required and their relative cost in terms of slots in T. Increasing p, affects the performance in two directions: a) more requests can be inserted without fragmentation and therefore, the efficiency increases both for Basic Raster and Raster algorithms, and b) the number of slots in T increases, and therefore, more requests can be served. It is also easy clear that the value of q affects only the number of requests than can be served.

B. Fixed Mapping Algorithm

An alternative approach for two-dimensional mapping has been presented in [13]. The table T is pre-segmented into a set of M fixed rectangles of different heights and widths. During the rest of this section, we will refer to the fixed rectangles as containers. Usually Pre-allocation is done using an expected distribution of requests and available rates, creating various containers to achieve efficient use of power and bandwidth. Mapping in a fixed segmentation approach is coupling requests with the best container using a size (capacity) criterion.

The fixed segmentation approach is simple and reduces the mapping problem to a simple matching algorithm. The criterion for selecting pre-allocated container R_j for request D_i such that $length(R_j) \times (width(R_j) - 1) \geq D_i$ assumes that the left most slots of R_j are the preambles costs and therefore are not considered as slots that can be allocated.

It can be easily shown that by (pre)sorting the pre-allocated containers by their $EArea$ value will result with time complexity which depends only on the matching step, that is, for each of the n requests we search for the best of M containers. The time complexity for the matching will result with $O(n \log M)$. The main disadvantage of the algorithm is the fact that an a-priory knowledge of the distribution of L is required such that the overhead is minimized, or at least is known in advance. In the general case, we see that for given request D_i when coupled with container R_j, either $EArea(R_j) = D_i + \varepsilon_i$ or no appropriate R_j can be found. In the first case, the efficiency of the allocation is defined as

$$\eta_{fixed} = \frac{D_i}{D_i + width(R_j) + \varepsilon_i} \tag{3}$$

η_{fixed} decrease as ε_i increases, that is, the efficiency drops with the increase in inconsistency (i.e., ε_i) between the pre-defined rectangles and requests. We have selected to limit the maximum slots per request to $p-1$ in a $T(p,q)$ table, since it will result with the highest utilization value per allocation. While this method is appealing due to its low complexity, the mapping can be inefficient when the prediction of the fixed allocation is not accurate.

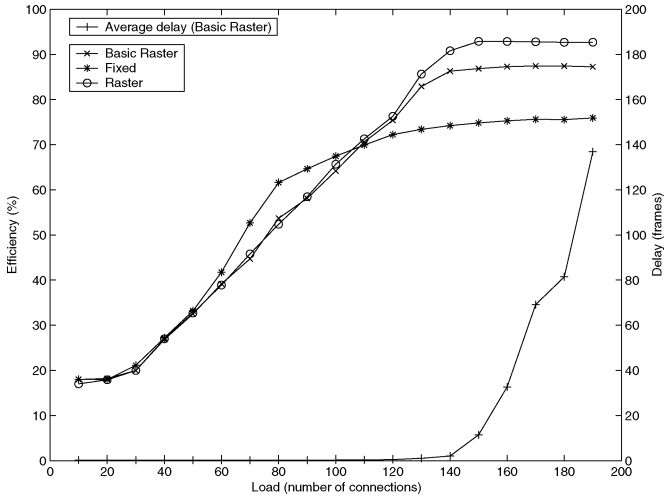

Fig. 3. Mapping efficiency of *Basic Raster*, *Raster* and *Fixed* mapping algorithms and average delay of the *Raster* algorithm as a function of system load for $T(25,30)$.

Fig. 3 shows the efficiency of the *Basic Raster*, *Raster* and *Fixed* mapping algorithms and the average packets delay for different changing load conditions. The average delay starts to increase at load of around 140 connections and exceeds the 10 frames delay at load of around 155–160 connections. The point of exceeding the delay threshold defines the system operating point as the balance of maximal tolerated delay and efficiency. As seen in Fig. 3, the performance of the mapping algorithms converges to its maximal value at load of about 150 connection, under average delay of about 5 frames. In addition, the difference between the efficiencies of the raster based algorithms and the fixed mapping algorithm, expresses the misalignment of the fixed approach prediction chosen in our implementation and the dynamic nature of the IP traffic.

V. Allocations Under Constraints

When dealing with requests without constraints, the Raster algorithm seems to provide a simple and reasonable solution to the *B2DAP* problem. But when requests have a set of constraints attached to them, the Raster algorithm must be modified in order to satisfy these constraints. In this section we will show algorithms that solve the *C2DAP* problem.

In this paper, we examined the $Granularity$, Max_Width and $Priority$ while in the full work, we also examined the $Late\_Time$ constraint. These constraints are grouped into the following classes: **Minimal Set**: $\{Granularity, Max\_Width\}$ **Advanced Set**: $\{Granularity, Max\_Width, Priority\}$ The *Minimal Set* class, defines a minimal set of constraints that are attached to each request. This set describes the granularity constraint and enforcing an upper bound on the number of rows that can be used for an allocation. For the algorithms defined ahead, the *Minimal Set* class will always be used. The *Advanced Set* adds a priority tag to a request The $Priority$ constraints are optional.

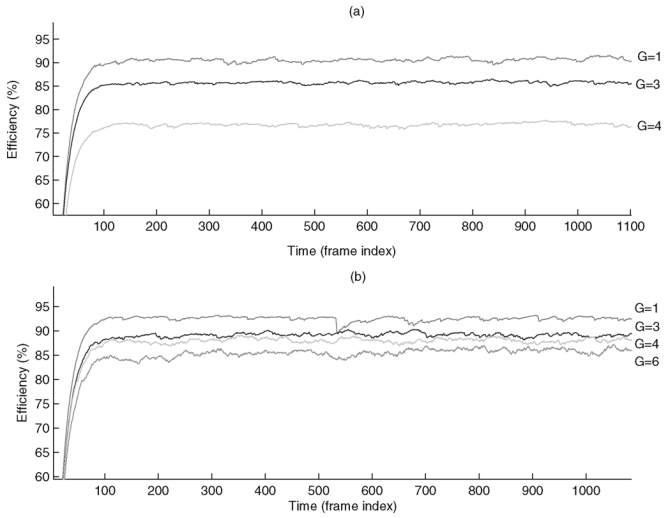

Fig. 4. *MSR* efficiency as a function of granularity and size of T (a) $\mathrm{T} = (20 \times 30)$; (b) $\mathrm{T} = (25 \times 30)$.

A. Minimal Set Raster (MSR)

MSR algorithm is a modified Raster algorithm that satisfies the *Minimal Set* constraints. Each request D_i is aligned to be a multiple of the granularity parameter. Each rectangle R in the allocation $Alloc(D_i)$ has width of one row and length which is a multiple of G (excluding a preamble).

Selecting requests to be served is similar to the Raster algorithm with the following additional properties: **Property 1**: If $Alloc_{rows}(D_i) = Max\_Width + 1$, the request is backlogged (i.e., not selected for mapping in a certain selection iteration). **Property 2**: If $Alloc_{rows}(D_i) > Max\_Width + 1$, the request is dropped. **Property 3**: The algorithm tries always to select requests that do not leave less than $G + 1$ (i.e., granularity size plus a preamble) unused slots in their last row, thus trying to minimize the overall number of unused slots.

Property 1 assumes that the required number of rows for an allocation may be reduced (by one row) when starting the mapping of column with smaller index and therefore backlogging a request is done (assuming that there will be an opportunity to insert this request). Property 3 defines a selection criterion that tries to minimize the overall overhead by tying not to leave unused slots at the end of a row. In order to satisfy the granularity constraint, the algorithm performs the following steps: Pad each request to be a multiple of G. If current row contains less than $G + 1$ unused slots, then a $\mathrm{preamble} + \mathrm{multiple\ of}\ G$ cannot be inserted and the slots are not allocated. If an allocation is fragmented in the current row, then in the current row only multiples of G can be allocated, therefore a reminder is calculated as: $Reminder \equiv ((p - col) \bmod G)$, where *col* is the index of the current column in T. $Reminder$ represents the number of slots that are not allocated in the current row.

Fig. 4 shows the effect of the granularity constraint on the efficiency of the mapping algorithm for two values of the time slot number: 20 time slots in 4(a) and 25 time slots in 4(b). As can be seen, in both cases, due to alignment requirement of requests to the units of defined granularity, the overall efficiency of the mapping algorithm decreases when the granularity unit value increases, since the average number of overhead slots increases. In addition, we can see the correlation of the granularity value and the length of T. As is seen in Fig. 4(a), the efficiency decreases strongly (as compared to Fig. 4(b)) since the granularity values are divisors of the length of T (excluding the preamble) in the lower figure, while in the upper figure they are not, and therefore, additional overhead is added as unallocated slots in some rows.

B. Priority Raster

This section deals with the case where some requests have a precedence (and hence higher priority) on other requests. The scheduler performs the scheduling task, in which several requests are scheduled within the same frame. Since the *Raster* based mapper can bump some of the requests due to the addition of preambles and unused slots, the priority of the requests must be kept such that a request with a given priority may be mapped if and only if all requests of higher priority are already mapped.

Definition 5.1: Priority Preservation: A Priority Preservation property requires that at the end of a mapping process, a request with priority P is mapped to table T if and only if all requests with priority higher than P are mapped in T.

Note that the Priority Preservation property may reduce efficiency in cases where a large request of high priority cannot be mapped, due to lack of space, the unused allocation space cannot be used for smaller requests of lower priority. We keep the Priority Preservation constraint, since the mapper may not be aware of the scheduling policy employed by the scheduler.

The strict priority constraint defines a strict order between the requests, thus we define that each request D_i is associated with priority class $p_i \in \{1, \ldots, k\}$ forming k available priority classes. The priority label of the requests defines an order between them where 1 is the highest and k is the lowest.

The priority constraint may contradict the efficiency considerations of the mapping algorithm. Taking for example a table of $T(5, 2)$, and request list {(2,1),(2,1),(1,2),(1,2)} of (size,priority) pairs. Than *Raster* mapping according to priority will map first the requests of size 2 and than 1 request of size 1, while the regular *MSR* algorithm will manage to map all the requests (in each row one request of size 2 and one request of size 1) and with smaller cost.

In order to perform raster based mapping with priority constraints, we will use bounds defining the worst and best case cost per allocation. The best case cost bound is used for a preprocessing calculation in which the mapper will return to the scheduler all the requests that cannot be mapped. The worst case bound is used to define a *Safety Zone*, that is, a list of requests and priority classes which are *guaranteed* to be allocated even if maximal cost is added per allocation. Having a safety zone enables us to perform mapping using only efficiency consideration and ignoring the priority considerations.

Given a table $T(p, q)$ and a requests list $L = \{D_1, \ldots, D_n\}$, where each request D_i is associated with priority class $p_i \in \{1, \ldots, k\}$, we define P_{up} and P_{low} by:

Definition 5.2: Upper Bound P_{up}: Defines a priority class P_{up} such that for each priority class $p > P_{up}$, all requests from the class p cannot be allocated, while keeping the priority preservation property.

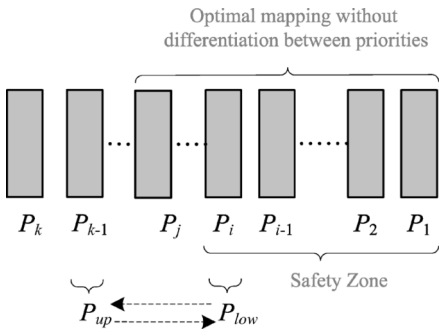

Fig. 5. Illustration of priority classes and priority bounds.

Definition 5.3: Lower Bound P_{low}: Defines a priority class P_{low} such that for each priority class $p \leq P_{low}$, requests from class p are guaranteed to be allocated, while keeping the priority preservation property.

The P_{up} bound defines an upper bound for an allocation scenario and represents the case of minimal overhead for each allocation, that is, always assumes that each allocation starts at the beginning of a row. The P_{low} bound defines a lower bound for an allocation scenario and represents the case of maximal overhead for each allocation. The values of P_{low} and P_{up} are calculated using their definitions presented below.

We assume that the request list L is segmented according to the priority classes, $L = L_1 \cup \ldots \cup L_k$ were each request D_i belongs to the group L_{p_i}. We define by A the set of allocated requests, and for any $D_i \in A$, by $\Gamma(D_i)$ the actual cost of the request after allocation: the requested slots (D_i) plus the actual overhead. For each request D_i, we define $\Gamma_{max}(D_i)$ as its maximum requirement $(\text{request} + \text{maximum overhead})$ from T : $\Gamma_{max}(D_i) = D_i + \lceil D_i/\beta \rceil + 1$, where β is the number of slots which can be allocated in an empty row. For given priority value j, we calculate the sum SL_j by: $SL_j = \sum_{D_i \in \cup_{r=1}^{j} L_r \setminus A} \Gamma_{max}(D_i) + \sum_{D_i \in A} \Gamma(D_i)$. Then, P_{low} is calculated as follows: $P_{low} = \max\{j | SL_j \leq (p \times q)\}$. For each request D_i, we define $\Gamma_{min}(D_i)$ as its minimum requirement $(\text{request} + \text{minimum overhead})$ from $T : \Gamma_{min}(D_i) = D_i + \lceil D_i/\beta \rceil$. For given priority value j, we calculate the sum SU_j by: $SU_j = \sum_{D_i \in \cup_{r=1}^{j} L_r \setminus A} \Gamma_{min}(D_i) + \sum_{D_i \in A} \Gamma(D_i)$. Then, P_{up} is calculated as follows: $P_{up} = \max\{j | SU_j \leq (p \times q)\}$.

As can be noted from the above definitions, that the upper bound P_{up} and lower bound P_{low} are dynamically updated during the allocation process, where P_{low} can be increased (i.e. by using lower cost for an allocation) and the upper P_{up} can be decreased (i.e. by using higher cost for an allocation). Fig. 5 illustrates the priority classes and priority bounds.

Let $\text{TDM}_{\text{OPT}}(L)$ represent an optimal mapping algorithm without *Priority Preservation* property and let $\text{TDMP}_{\text{OPT}}(L)$ represent an optimal mapping algorithm with *Priority Preservation* property. Let $E(ALG)$ define the efficiency of an arbitrary two-dimensional mapping algorithm ALG.

Theorem 5.1: Given a table $T(p,q)$ and a list of requests $L = \{D_1, \ldots, D_n\}$, let P_{low} represent the lower bound for L and let $L' = L_1 \cup \ldots \cup L_{P_{low}}$. Then, $E(\text{TDM}_{\text{OPT}}(L')) = E(\text{TDMP}_{\text{OPT}}(L'))$.

From Theorem 5.1 we can see that all the requests in L' (i.e., the safety zone) may be mapped without relating to the priority constraint. We use the upper bound P_{up} as the bound for setting the requests list L and to bump requests from lower priority classes, which cannot be mapped while satisfying the Priority Preservation property.

We now present a heuristic algorithms based on the property of Theorem 5.1 for two-dimensional mapping with *Priority Preservation* which performs well both from efficiency and runtime perspectives. In these algorithms, the values of P_{low} and P_{up} are changed dynamically. Their values may change w.r.t. the currently defined part of the allocation, since in that part, the actual overhead may be somewhere between its lower and upper bounds.

a) Strict Priority Raster (SPR): Is the simplest approach for priority handling. The algorithm groups the requests according to their priority class and apply the *MSR* algorithm for each group. (Note, the algorithm assumes that *MSR* algorithm will tag allocated slots in T such that they will be recognized in successive iterations). It is easy to see that the *SPR* algorithm keeps the *Priority Preservation* property. The *SPR* algorithm is primarily concerned with *Priority Preservation*, rather than efficiency. Therefore it will suffer from low efficiency compared to any algorithm that can consider both.

b) Heuristic Priority Raster (HPR): Is an algorithm which uses the property of Theorem 5.1 for mapping while keeping the *Priority Preservation* property. The HPR algorithm starts by calculating the lower bound P_{low}. The main iteration is similar to the *MSR* algorithm and it is performed on the current L' group, which is the group of requests of priority classes smaller than P_{low}. After an allocation of each request, the P_{low} bound is updated. The calculation of P_{low} finds full priority classes matching the criterion. When the algorithm finishes all requests from priority class P_{low}, the list L' is empty. However, it may happen that L is still not empty, and there are still available slots in T. Then, the algorithm calls the *SPR* algorithm for allocations of the leftover requests, in the order of their priority. The algorithm stops when L is empty or there is no more room in T.

The HPR algorithm satisfies the *Priority Preservation* property, since it allocates requests from L' that define a *safety zone* and hence ensures that all the requests will be mapped. In each mapping algorithm, the selection of a request to be mapped is done according some criteria, such as defined by the *MSR* algorithm and which are aimed to improve efficiency. According to Theorem 5.1, since the domain of the requests is from the safety zone, the priority constraint can be ignored during the mapping process for these requests, while still satisfying the *Priority Preservation* property.

Let us explain why using the *SPR* algorithm after that may be relevant. According to the definition of P_{low}, when all the requests in priority class less or equal to P_{low} are successfully allocated, $\sum_{D_j \in P_{low}+1} \Gamma_{max}(D_j)$ exceeds the number of available slots. However, this does not contradict a possibility to allocate *a part* of the requests of priority $P_{low} + 1$ (and may be even of lower priorities, if the total actual cost of allocation of all requests in $L_{P_{low}+1}$ would be strictly less than the remaining place in T).

The HPR algorithm deals with the case of finding the P_{low} bound on the boundary of full priority classes. In some cases, this degrades performance. An example for such case is as follows: given a list of {request, priority} pairs: {{2,1}, {2,2}, {2,3}, {2,4}, {1,5}, {1,5}, {1,5}, {1,5}} for $T(5,4)$. According to the definition, $P_{low} = 4$, since adding all the requests from priority class 5 will exceed the table size. After each mapping of the first four requests, P_{low} cannot be increased and the mapping order will be {2, 2, 2, 2, 1, 1, 1} while the optimal mapping is {2, 1, 2, 1, 2, 1, 2, 1} with no unused slots and all requests mapped.

Notice that the HPR algorithm may be improved by considering also requests from class $(P_{low}+1)$, for allocation, at every iteration. For the above example, considering requests from the priority class $(P_{low}+1)$ results with the mapping presented under the optimal title. However, recall that $SL_{P_{low}+1}$ exceeds $(p \times q)$, which means that not all requests from priority class $(P_{low}+1)$ are guaranteed to be inserted. Therefore, the HPR algorithm is updated to HPR^* by extending the group L' by the *safe part* of the class $(P_{low}+1)$, as follows:

$$L' = \bigcup_{j=1}^{P_{low}} L_j + \Big\{ D_i | D_i \in L_{P_{low}+1} \cap \Big(\Big(SL_{P_{low}} + \sum \Gamma_{max}(D_i) \Big) \le (p \times q) \Big) \Big\} \quad (4)$$

We extend Theorem 5.1 for the new case as follows:

Theorem 5.2: Given a table $T(p,q)$ and a list of requests $L = \{D_1, \ldots, D_n\}$, let P_{low} represent the lower bound for L and let L' be as defined in (4). Then, $E(\mathrm{TDM}_{\mathrm{OPT}}(L')) = E(\mathrm{TDMP}_{\mathrm{OPT}}(L'))$.

The selection of requests in the HPR^* algorithm is done according to the efficiency criterion. In each mapping iteration, the *best* request is selected according to the efficiency criterion, as defined by the *MSR* algorithm. Each such request is then verified that it belongs either to a priority class $\le P_{low}$, or to priority class $L_{P_{low}+1}$ according to (4). More directly, the latter condition is that the allocation of this request does not decrease the value of P_{low}, according to the definitions of SL_j and P_{low}. Otherwise, the request will be rejected, and a next possible request is then examined.

Since HPR^* improves HPR, we examine, by a simulation, the performance of the HPR^* algorithm only. Fig. 6 shows the performance of the HPR^* algorithm with comparison to *SPR* and *MSR* mapping algorithms with 8 priority classes. As can be seen, the heuristic priority approach improves the strict priority one, in *Raster* based algorithms, while satisfying the Priority Preservation constraint. The dynamic range for the operational point of 160 connections (see Fig. 3) is about 3.2% between our best case (*MSR*) and worst case (*SPR*). We see that the efficiency of HPR is higher than that of *SPR* by about 1% which is approximately 31% of the dynamic range.

Another approach of analysis is conditional: based on the assumption that the *MSR* algorithm provides a good approximation to the best performance that may be achieved by any algorithm, when there is no priority constraints, the improvement made by the HPR^* heuristic is quite substantial. Note that *MSR* should give a far not realistic upper bound, in the case of priority preservation, since it ignores the priorities.

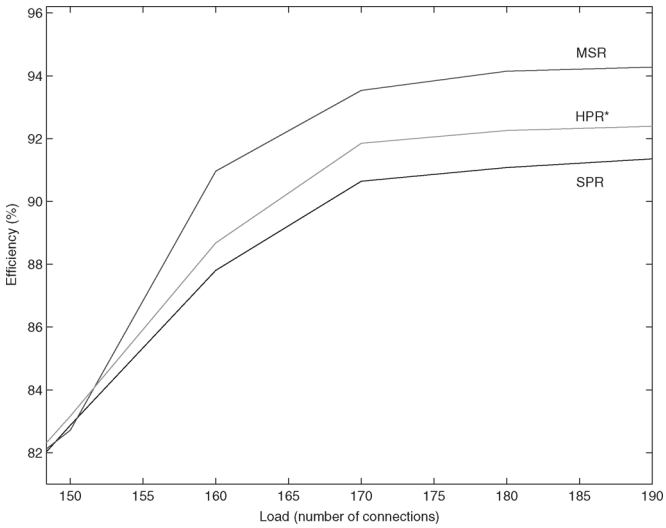

Fig. 6. Performance of *MSR*, *SPR* and HPR^* with 8 priority classes and as a function of system load in $T(25, 30)$.

VI. Conclusions

The IEEE 802.16 standard and especially its OFDMA mode, is one of the most popular technologies for fixed and mobile broadband wireless access. This paper provides a framework for defining the resource allocation approaches and presnts open problems for real OFDMA system parameters and constraints. The problem of two-dimensional scheduling and mapping in OFDMA systems where all sub-channels are equally adequate for all transmitters, is studied and we have presented a general set of OFDMA constraints, which are implied both from the physical characteristics of the OFDMA access scheme and from general QoS parameters which are relevant to OFDMA systems due to mapping restrictions. A new cost model which affects the cost of an allocation according to its spatial shape (i.e. by adding a preamble as left most slot in each allocation) was presented. Heuristics were proposed to solve the general problem without and with constraints relevant for QoS requirements for network applications. The algorithmic approaches from which the solutions were derived is the *Raster* approach. The *Fixed* Mapping approach was used due to its consideration as a practical solution that is used commercially and was proposed to prospective standardization groups. All the algorithms were evaluated using extensive simulations on a proprietary simulator implementing the OFDMA mode of the IEEE 802.16 standard ([7], [8]).

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W. H. Freeman and Co., 1979.

[2] C. Y. Wong, R. S. Cheng, K. B. Letaief, and R. D. Murch, "Multiuser OFDM with adaptive subcarrier, bit, and power allocation," *IEEE Journal on Selected Areas in Communication*, vol. 17, October 1999.

[3] R. Grunheid and H. Rohling, "Adaptive modulation and multiple access for the ofdm transmission technique," *Wireless Personal Communications*, vol. 13, pp. 5–13, May 2000.

[4] M. Ergen, S. Coleri, and P. Varaiya, "QoS aware adaptive resource allocation techniques for fair scheduling in ofdma based broadband wireless access systems," *IEEE Trans. Broadcasting*, vol. 49, December 2003.

[5] I. Koutsopoulos and L. Tassiulas, "Channel-state adaptive techniques for throughput enhancement in wireless broadband networks," in *Proc. IEEE INFOCOM*, Anchorage, AK, April 2001, vol. 2, pp. 757–766.

[6] T. Javidi, "Rate stable resource allocation in OFDM systems: From waterfilling to queue-balancing," in *Allerton Conference on Communication, Control, and Computing*, September 2004.

[7] *802.16a: IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems—Amendment 2: Medium Access Control Modifications and Additional Physical Layer Specifications for 211 GHz*, IEEE 802.16a, April 2003.

[8] *802.16-2004: IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE 802.16-2004, October 2004.

[9] *Digital Video Broadcasting (DVB); Interaction Channel for Digital Terrestrial Television (RCT) Incorporating Multiple Access OFDM*, ETSI EN 301 958 V1.1.1, March 2002.

[10] *802.16e: IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands*, IEEE P802.16e/D11, September 2005.

[11] C. R. Baugh and J. Huang, Traffic Model for 802.16 tg3 mac/phy Simulations IEEE 802.16 work-in-progress document no. 802.16.3c-01/30r1, March 2001.

[12] N. Naaman and R. Rom, "Analysis of packet scheduling with fragmentation," in *Proceedings of Infocom'02*, New York, June 2002, pp. 824–831.

[13] S. Kapoor and J. Li, Initial contribution on a system meeting mbwa characteristics Contribution to IEEE 802.20 Working Group on Mobile Broadband Wireless Access, IEEE C802.20-03/16, March 2003.