On the Trade-off between Energy and Multicast Efficiency in 802.16e-like Mobile Networks*

Reuven Cohen† Liran Katzir‡ Romeo Rizzi§

Abstract

In this paper we define a new problem that has not been addressed in the past: the trade-off between energy efficiency and throughput for multicast services in 802.16e or similar mobile networks. In such networks, the mobile host can reduce its energy consumption by entering the sleep mode when it is not supposed to receive or transmit information. For unicast applications the trade-off between delay and energy efficiency has been extensively researched. However, for mobile hosts running multicast (usually push-based) applications, it is much more difficult to determine when data should be transmitted by the base-station and when each host should enter the sleep mode. In order to maximize the channel throughput while limiting energy consumption, a group of hosts needing similar data items should be active during the same time intervals. We define this as an optimization problem, and present several algorithms for it. We show that the most efficient solution is the one that employs cross-layer optimization by dividing the hosts into groups according to the quality of their downlink PHY channels.

Keywords: Mobile communication, Scheduling, Multicast channels, Energy management.

1 Introduction

An important goal of the IEEE 802.16e standard is to reduce the power consumption of the mobile hosts by introducing the sleep mode operation [14]. When there is no data awaiting transmission from the base-station to the mobile host or vice versa, the mobile host can move to sleep mode where energy consumption is minimized. How long the host can stay in this mode is negotiated

*A preliminary version of this paper was presented in Infocom'06. This journal version extends the Infocom version by addressing variable-size items, by presenting the new AMC-simple model, by presenting new algorithms for the AMC model and by extending the simulation section.

†Dept. of Computer Science, Technion, Israel

‡Dept. of Computer Science, Technion, Israel

§Università degli Studi di Udine, Facoltà di Ingegneria - Dipartimento di Matematica e Informatica, Via delle Scienze, 208, I-33100 Udine, Italy

Digital Object Indentifier 10.1109/TMC.2007.70729

between the host and the base-station through the exchange of MOB-SLP-REQ (from hosts) and MOB-SLP-RSP (to hosts) messages. This time duration depends on the applications executed by the host: real-time applications require the host to return to active mode after a short interval of several milliseconds, while non-real-time applications allow the host to stay in this mode much longer.

While HTTP – the "engine" of the Web – is a pull-based unicast application, the lion's share of mobile network bandwidth is likely to be employed by push-based multicast services[6, 22], for the following reasons:

1. The downlink channel in such networks is a broadcast physical channel, to which all mobile hosts can listen at the same time. Therefore, for the price of one transmission, the base-station can transfer the same data items to many hosts.

2. A significant portion of the data needed by individual users will probably be location-dependent. This implies that other users from the same broadcast domain ("cell") will most likely need the same data at the same time.

According to [6], mobile users in Japan who subscribe to KDDI's EZChannel multimedia service already receive content that is pushed into their terminals. In addition, a number of operators in Europe have launched sports information services that push short video clips of game highlights.

For unicast applications (service flows), the trade-off between delay and energy efficiency is quite well understood. However, for mobile hosts running multicast push-based applications, it is much more difficult to determine when data should be transmitted by the base-station and when each individual host should enter the sleep mode. Since many hosts are likely to need the same data items, and transmitting such data items many times may drastically degrade throughput, these hosts have to wake up at the same time.

In this paper we propose a scheme for sending multicast data to mobile hosts while addressing the trade-off between throughput and energy efficiency. Following 802.16e terminology, and regardless of the physical layer technology (SC, OFDM or OFDMA), we assume that the downlink channel is divided into fixed size time frames. Every frame starts with a preamble, followed by a region of

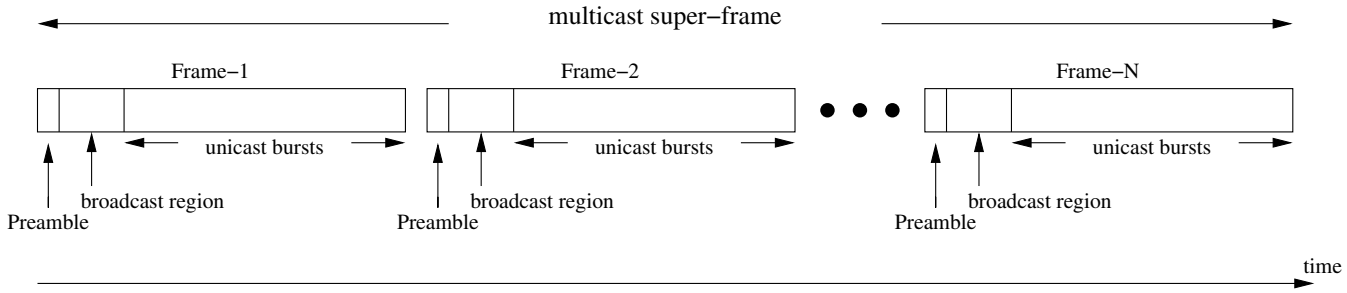


Figure 1: The concept of "multicast super-frames"

broadcast messages, and then a sequence of "unicast downlink bursts." Each burst has a different modulation/coding combination. The bursts are transmitted in order of decreasing robustness. For the sake of efficient multicast, we consider every N consecutive downlink frames as a single "multicast super-frame" (not to be confused with the standard AAS super-frame), as depicted in Figure 1.

Before a multicast super-frame begins, the base-station makes a scheduling decision for the broadcast regions of all the frames in this super-frame. The trade-off between energy efficiency and throughput for multicast transmission can be demonstrated in the following simple example. In order to allow each host to be active only one-half of the time, the base-station needs to transmit the multicast information only in the first (or in the last) $N/2$ frames of the super-frame. Of course, unicast information for the considered host should also be transmitted in the same specific frames. However, since there is no correlation between the unicast transmission to different hosts, unicast scheduling is much easier than multicast scheduling, and is therefore ignored throughout the paper.

If two hosts h_1 and h_2 need to get the same piece of information but each of them wakes up during different frames, the information must be transmitted by the base-station twice, resulting in throughput degradation. This degradation is avoided if all the hosts are active all the time, because then every data item can be transmitted only once. Another option is to define the same wake-up interval for hosts that need the same broadcast items. Therefore, the optimization problem is to determine *what data should be transmitted in the multicast region of every frame and when every host should become active.*

The proposed multicast super-frame concept is applicable both to FDD (Frequency Division

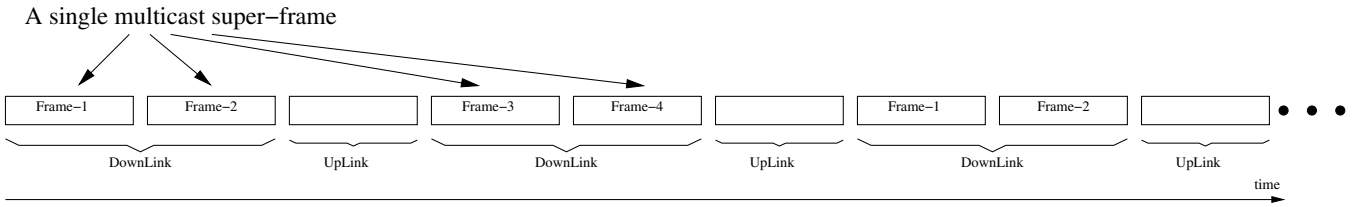


Figure 2: A multicast super-frame in a TDD system

Multiplexing) and to TDD (Time Division Multiplexing). Figure 1 can be considered as a TDD example, because all the downlink frames are transmitted consecutively. In contrast, in an FDD scenario, a downlink super-frame can be interrupted once or more by uplink frames. This is depicted in the example in Figure 2. In this figure, each multicast super-frame consists of 4 frames. Since the time allocated to downlink transmissions is only sufficient for 2 frames, each multicast super-frame is interrupted by one uplink frame. Without loss of generality, we consider an FDD system for the rest of this paper.

In a push-based system with a broadcast physical channel, it is also important to determine what data items should be received by each host. A practical solution is to ask clients to subscribe to content channels such as sports, news, or traffic reports [3, 7, 18]. These will be used to create client profiles, which can then be used by the base-station to produce a list of data items that should be broadcast to each host. This list is based on various considerations, which are out of the scope of the present paper. For instance, it can be decided that the base-station should send to each node the most recent data items created by every content channel the host is subscribed to. Alternatively, if the system employs the concept of cyclic data broadcast (also known as "broadcast disks"[1], or "data carousel"), the base-station may need to transmit to each node data items that have already been transmitted in the past but have not been stored at the host buffers.

If the bandwidth allocated to the push-service is not sufficient for the transmission of all the data items in the base-station list, the scheduler logic at the base-station has to determine which items should get priority. To this end, we assume that each data item i is associated with a merit attribute $m(h, i)$, which indicates the profit host h gains from receiving item i during the next multicast super-frame. A private case is when $m(h, i)$ is 1 if h is subscribed to the content channel

of item i, and 0 otherwise. A more general case is when $m(h,i) \in [0,1]$ indicates the probability that host h will actually use this item. The exact way to determine $m()$, which depends on the multicast model (push-based vs. pull-based), the caching capabilities of each host, error recovery, and other system-dependent parameters, is beyond the scope of this paper.

The *normalized throughput* of the scheduling algorithm during a given multicast super-frame is defined as

$$\sum_{h,i} \big(m(h,i) \cdot r(h,i)\big) / \sum_{h,i} m(h,i), \tag{1}$$

where $r(h,i) = 1$ if host h receives data item i during this broadcast, and 0 otherwise. Host h is said to receive data item i if this item is broadcast when h is not in sleep mode.

Consider first the case where all the hosts are always active and therefore receive all the broadcast information. We say then that the system contains $C = 1$ logical broadcast channels, with which all the hosts are associated. If the data items are of fixed size, and the broadcast region of all the frames in a multicast super-frame can accommodate L such items, the base-station could easily determine the L data items that should be broadcast in each frame of this channel in order to maximize the profit. It simply needs to compute the value of $\sum_h m(h,i)$ for every item i, to sort the items according to this merit attribute in descending order, and to broadcast the first L items in the sorted list in any order. This greedy algorithm provides an optimal schedule. When the items are not of equal size, the problem is know as the KNAPSACK problem, and it is NP-complete.

Items that are not broadcast due to lack of bandwidth, or are received by some hosts with errors, can be scheduled in one of the succeeding multicast super-frames, depending on their merit at that time. Obviously, for the hosts that correctly received the item the merit should be reduced to 0, whereas for hosts that have not received the item the merit can remain unchanged. If a host needs a data item that is not in its local cache because it was not recently broadcast, because a transmission error occurred during the last broadcast, or because of the pruning policy of its local cache, it can explicitly request this item using unicast communication.

Figure 3(a) depicts the case where the system contains $C = 2$ logical broadcast channels, allowing every host to be active only half of the time, and Figure 3(b) depicts the case where the system

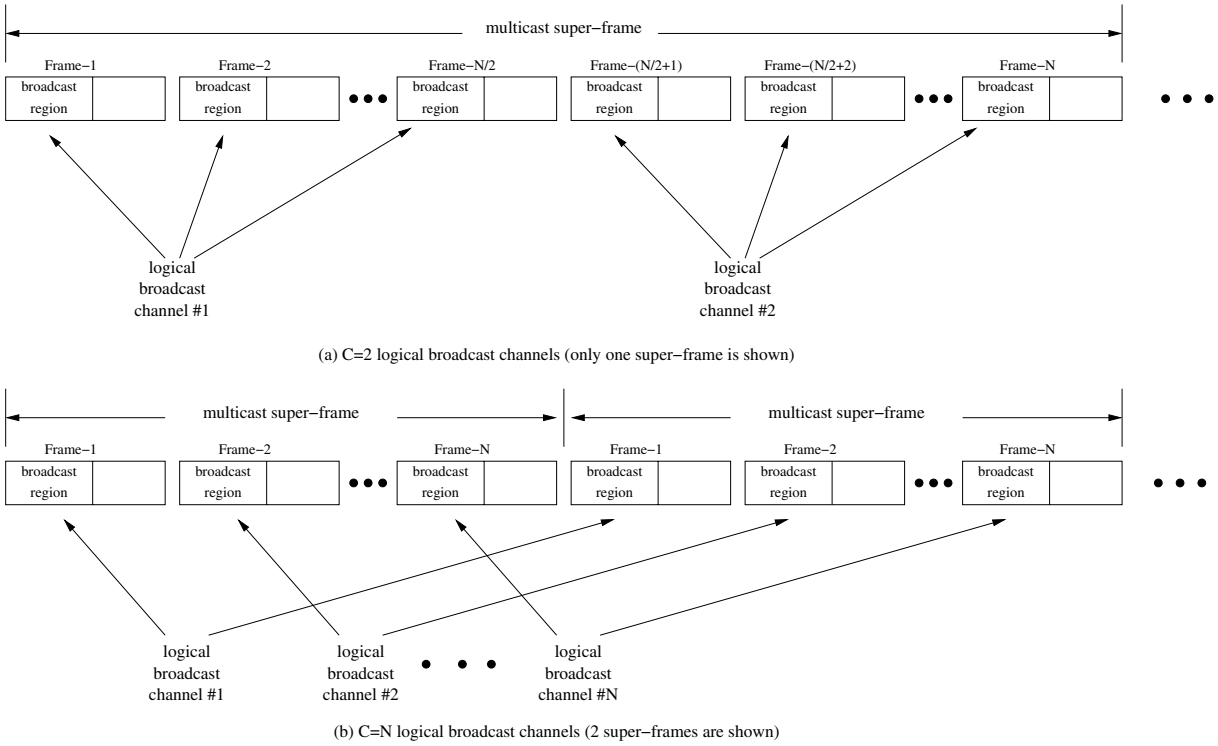


Figure 3: The Concept of "Logical Broadcast Channels"

contains $C = N$ logical broadcast channels, allowing every host to be active only $1/N$th of the time.

There are three approaches for maintaining the association between each host and a logical broadcast channel. The first is the static approach, where the base-station determines in advance to which logical broadcast channel every host should listen, either using some random algorithm that load balances the number of hosts associated with each channel, or using a more intelligent algorithm that takes into account the content channels every host is subscribed to. We call this the "static model," because the association between every host and a logical broadcast channel is fixed. In the static model the base-station multicast scheduling algorithm is easy because the greedy algorithm described earlier for the case where $C = 1$ can be executed for each logical broadcast channel independently.

Another approach is to change the association between hosts and logical broadcast channels dynamically. This would be done during the scheduling process executed by the base-station before the beginning of each multicast super-frame, in accordance with the actual data awaiting transmis-

sion. Such a solution is referred to as the "dynamic model." Its main drawback is that it imposes significant signaling overhead because, before a multicast super-frame begins, the base-station needs to inform each host which logical broadcast channel it should listen to. Nonetheless, such a solution might increase broadcast bandwidth utilization because it can minimize the number of items that have to be transmitted more than once. We consider this model mainly as a benchmark for the static model.

The third model, referred to as the "adaptive modulation and coding (AMC) model," is based on cross-layer optimization. It uses PHY layer information to determine the quality of the downlink channel of each mobile user. It then divides the hosts into broadcast groups in accordance with this parameter, such that hosts with good PHY layer conditions are not be associated with the same logical broadcast channel as hosts with bad ones. The main advantage of this approach is that the most robust and inefficient modulation for broadcast (QPSK) need not be used for hosts with high quality downlink channel. Rather, such hosts can receive the broadcast information using more efficient coding schemes like 16-QAM or even 64-QAM. While adaptive modulation is a well-known technique for increasing the throughput of unicast transmissions, it has not been used for multicast or broadcast because it is assumed that at least one of the destination nodes will be very likely to have a bad downlink channel and should thus receive the data using the most robust encoding. However, when the base-station has the ability to control the association between hosts and their broadcast groups, the transmission of broadcast data using efficient modulation/coding scheme increases the throughput significantly.

The rest of the paper is organized as follows. In Section 2 we discuss related work, mainly in the context of scheduling algorithms for broadcast channels. In Section 3 we discuss the computational complexity of finding an optimal scheduling for the various models. In Section 4 we present algorithms for the static and dynamic models. In Section 5 we present algorithms for the AMC model and simulation results for all the models. Finally, Section 6 concludes the paper.

2 Related Work

This paper addresses the trade-off between energy and throughput for multicast in mobile networks. While many papers have addressed the trade-off between throughput/delay and energy efficiency for unicast applications in broadcast systems (e.g. [9, 23, 24]), to the best of our knowledge no paper has addressed this problem in the context of multicast applications.

A dynamic system for data delivery to mobile users is proposed and discussed in several papers (see, for example, [3] and references therein). In [4], a system where not all the required data items are available at the broadcasting station is discussed. A set of mechanisms that coordinate the process of broadcast scheduling with the process of locating and retrieving the data items to be broadcast is therefore proposed. Reference [18] presents a framework for information dissemination to mobile clients using a model called Publish-Subscribe. It also reviews some of the "push" architectures for mobile hosts.

In [5] and [12] the hosts need to inform the broadcasting station before each time frame which data items are needed. It is also assumed that the broadcasting station receives this information asynchronously and processes the requests in the order they arrive. In [10], a model similar to the one considered here is discussed in the context of a satellite-based content distribution network: the base-station uses a merit matrix in order to determine what information to broadcast at every time slot.

In [21] the authors distinguish between two models: pull-based and push-based. In the former, the receiving parties inform the broadcasting node of their exact requirements. This is similar to the models discussed in [5] and [12]. In the latter model, the target is to minimize the average delay of the receiving parties, whose exact requirements cannot be sent to the broadcast station. In [13], this idea is extended to multiple channels, and it is assumed that the hosts tune their receivers to a random channel.

In [1] a system called "broadcast disk" or "data carousel" is proposed, where the broadcasting station repeatedly broadcasts all of the data items. In this system, two models are considered. In the first model the hosts do not have a cache and cannot request special delivery of a missing data item. Hence, the base-station must broadcast the same data items periodically, and the main problem

is to determine how often every data item should be broadcast in order to minimize the response time – the time elapsed between the user's request to view a data item and the next broadcast of this item. The second model addressed in [1] is the case where the hosts have caching capabilities. Hence, in addition to developing a scheduling strategy, a cache pruning policy must be developed as well. Unlike in a regular caching system where the least important objects are usually pruned first, here the most important objects are pruned first because these objects are likely to be broadcast more often. In [8] a similar system is considered, but the issue of broadcast reliability is addressed. A solution called "digital fountain" is proposed, where the broadcasting station transmits a stream of distinct encoded packets. Reference [15] addresses the cache replacement issue at the client, but in the context of broadcast mobile networks.

3 Computational Complexity of Broadcast Scheduling over Multiple Channels

We consider three different models for the association between hosts and logical broadcast channels: the static model, the dynamic model, and the adaptive modulation/coding (AMC) model. For each model we define an optimization problem as follows:

SMBC-S: Scheduling over Multiple Broadcast Channels – the static model: Assuming a fixed association between hosts and logical broadcast channels, what items should the base-station transmit on each channel in order to maximize the broadcast profit?

SMBC-D: Scheduling over Multiple Broadcast Channels – the dynamic model: What items should the base-station transmit on each broadcast channel, and to which broadcast channel should every host listen, in order to maximize the profit of the broadcast?

SMBC-AMC: Scheduling over Multiple Broadcast Channels – the AMC model: What items should the base-station transmit on each broadcast channel, and to which broadcast channel should every host listen, in order to maximize the profit of the broadcast, assuming that the quality of the downlink channel associated with every host is known to the base-station?

Throughout the paper we consider only the profit metric to measure the effectiveness of the various models and algorithms. The "profit of the broadcast" is defined as $\sum_{h,i} m(h,i) \cdot r(h,i)$.

Here $m(h,i)$ indicates the profit host h gains from receiving item i, while $r(h,i)$ is 1 if i is broadcast on the channel to which host h listens, and 0 otherwise. While other measurers such as the average delay could also be used as an optimization metric, we believe that the profit-based approach is more adequate for the following reasons:

1. When the same data item is transmitted on multiple channels, the load on the network increases. Consequently, even if the initial load is below 1, the load due to multiple transmissions of the same item may exceed 100% of the network capacity. Measuring the average delay in such a case is not possible.

2. With the profit-based approach it is still possible to enforce the timely transmission of data items when necessary. The idea is to assign a very high profit to the transmission of these items while they are relevant, and to assign them a profit of 0 when they are no longer relevant.

3. The profit-based approach addresses the "best-effort" nature of the multicast push-based services, in the sense that it allows the residual bandwidth available to this service to be optimized regardless of its size.

For the dynamic model we assume that the base-station determines the association between hosts and logical broadcast channels for every multicast super-frame. This information is then broadcast to all the hosts in a special broadcast area located at the beginning of the first frame in every super-frame.

When the items are of fixed size, the greedy algorithm presented in Section 1 is optimal for SMBC-S. The time complexity of this algorithm is $O(C \cdot I \cdot \log(I))$, where C is the number of logical broadcast channels and I is the number of data items. When the items are of variable size, SMBC-S is the well-known NP-complete KNAPSACK problem. SMBC-D is equivalent to SMBC-S when there is a single channel. Similarly, SMBC-AMC is equivalent to SMBC-S when all the hosts have the same downlink reception quality. This implies that SMBC-D and SMBC-AMC are also NP-complete for variable size items. The IEEE-802.16 standard supports the concept of fragmentation. However, even with fragmentation, the scheduling problem for SMBC-S (and therefore for SMBC-D and SMBC-AMC) remains NP-complete[19]. Thus, and also because fragmentation contributes

marginally or even negatively to the overall profit, we do not consider it in this paper.

Unlike SMBC-S, SMBC-D is NP-complete even if all the items are of fixed size, as proved in the following lemma.

Lemma 3.1 *Assume there are at least 2 channels and that the items are of fixed size. Then, SMBC-D is NP-complete. Moreover, the problem remains NP-complete even if for every data item i and every host h, $m(h,i) \in \{0,1\}$.*

Proof: We show a reduction to SMBC-D from the well-known NP-complete GRAPH-PARTITIONING problem, defined as follows. Given a graph $G(V,E)$ with $|V| = 2n$, partition G into two equal size disjoint clusters such that the number of edges of E whose incident vertices belong to different subsets is minimized. We associate a data item with each node $v \in V$ and a host with each edge $e = (u,v) \in E$. Moreover, we assume that each host wants to receive only the two data items associated with its corresponding vertices. Namely, for $e = (u,v) \in E$, we have $m(e,u) = 1$, $m(e,v) = 1$ and $m(e,i) = 0$ for every $i \notin \{u,v\}$.

We consider a network with two logical broadcast channels and with a frame size of n slots, which is half of the number of data items. This implies that with two channels every data item is (a) broadcast either on channel A or on channel B; or (b) broadcast on both channels; or (c) broadcast on no channel. In order to ensure that each data item is broadcast on exactly one channel, we define an additional host, called $\alpha(v)$, for every node $v \in V$, and assign $m(\alpha(v),v) = |E|$ for every v. This ensures that each data item must be broadcast on at least one channel and therefore that every data item is broadcast on exactly one channel. Consequently, an algorithm that solves SMBC-D determines which of the n data items should be broadcast on each channel, and which channel each host should listen to. Maximizing the profit implies minimizing the number of edges in the cut. This is because, for every edge in the cut, the corresponding host can receive only one of the two data items it wants to receive, whereas for every edge not in the cut the associated host can receive both items. Therefore, an optimal solution for SMBC-D is an optimal solution for the GRAPH-PARTITIONING problem. ■

In what follows we show that SMBC-D has a C-approximation algorithm for any fixed number C of logical broadcast channels.

Lemma 3.2 *Assume there are C logical broadcast channels. Then, there exists a C-approximation algorithm for SMBC-D in the case where all sizes are equal, and a $2C$-approximation algorithm otherwise. Furthermore, these approximations are possible even when the logical broadcast channels are not of equal size.*

Proof: For every channel i, $i = 1,\ 2, \ldots, C$, solve the problem assuming that channel i is the only one. If all sizes are equal, then the greedy algorithm (Algorithm 1 below) solves this 1-channel problem to optimality. Otherwise, this algorithm yields 2-approximation. We obtain C gain values $g(1) \cdots g(C)$. Using only the solution for channel i, for which the corresponding value $g(i)$ is maximum, and leaving all the other channels empty would deliver a C-approximate solution for the fixed size and a $2C$-approximation solution for the variable size. ■

SMBC-D is similar to the Catalog Segmentation Problem addressed in [17]. The authors of [17] present for this problem a sampling-based approximation scheme. However, this scheme is efficient only for dense instances, when there exists an $\epsilon > 0$ such that every host requires at least a fraction ϵ of the data items. Given an arbitrary parameter $\delta > 0$, the algorithm's running time is $O((H + I)^{O(C \cdot \log C)}/\delta\epsilon)$. This algorithm produces a solution within $1 - \delta$ of the optimal. Because of the running time complexity of this algorithm, and because the condition on the density of the instance is unlikely to hold in the system considered in our paper, the algorithm of [17] is impractical.

Lemma 3.3 *Assume there are at least 2 channels and that the items are of fixed size. Then, SMBC-AMC is NP-complete. Moreover, the problem remains NP-complete even if for every data item i and every host h, $m(h, i) \in \{0, 1\}$.*

Proof: SMBC-D is a private case of SMBC-AMC when the downlink channel quality is identical for all the hosts. ■

4 Algorithms for the Various Models

We start with a greedy algorithm for SMBC-S. The algorithm selects for transmission in each logical broadcast channel the items that bring the highest weighted profit for the hosts that are associated

with this channel. The weighted profit of each item i in channel c, denoted $\pi(i,c)$, is computed as follows:

$$\pi(i,c) = \frac{\sum_{h \in H(c)} m(h,i)}{\text{size\_of}(i)},$$

where $H(c)$ is the set of hosts associated with logical broadcast channel c. A formal description of the algorithm is as follows:

Algorithm 1 *A greedy algorithm for SMBC-S*

For each logical broadcast channel $c \in [1 \cdots C]$ *do*

- *for every data item* i*, let* $\pi(i,c) = \frac{\sum_{h \in H(c)} m(h,i)}{size\_of(i)}$.
- *order the set* I *according to the value of* $\pi(i,c)$.
- *select items for transmission starting from the one with the highest* $\pi(i,c)$*. An item* i *is selected if the previously selected items have not consumed the entire bandwidth of* c *and if the remaining space is not smaller than* $size\_of(i)$. □

When all the items are of equal size, Algorithm 1 finds the optimal solution. When the items are of variable size, this is a 2-approximation algorithm. That is, its profit in the worst case is at least 1/2 that of an optimal algorithm[1]. It is possible to find the optimal solution for the variable size case using efficient polynomial-time algorithms in some private cases. If the frame F is not too big, an $O(F \cdot I)$ dynamic-programming algorithm for KNAPSACK can be used to find the optimal solution. If the range of item sizes is small, say $\{1,\ 2, \ldots, \text{maxItemSize}\}$, then an optimal solution can be found in $O(F \cdot \text{maxItemSize})$ time, again using a dynamic programming algorithm [16].

Consider a logical broadcast channel c, and let $H(c)$ be the set of hosts associated with this channel. Let RF be a random variable indicating the number of requests that are fulfilled during a multicast super-frame, and RNF be a random variable indicating the number of requests that are not fulfilled. To simplify the discussion, we assume that all the items are of equal size and that $m(h,i)$ is either 1 or 0. Hence, $m(i) = j$ if and only if item i is requested by exactly j hosts. Let

[1]To be more precise, the output of this algorithm should be compared to the schedule that contains only the maximum size item. From these two schedules, the one with the greater benefit is selected [16].

T be the number of items that can be accommodated in the considered logical broadcast channel. Let $\#(I \geq j)$ denote the number of items for which there exist at least j requests. Hence,

$$RNF = \sum_{j=1}^{H(c)} \max\{0; \#(I \geq j) - T\}. \tag{2}$$

To understand why Eq. 2 holds, recall that only the most popular T items are transmitted. Hence, for $j = 1$, we count all the data items for which there is any request. By subtracting from this number the value of T, we get the number of data items that are requested by any host but not transmitted. However, the penalty is 1 only for those items that are requested exactly once. The penalty for a non-broadcast item i with $m(i) = 2$ is 2. We "charge" this item 2 units of penalty in two steps: first, when we consider the number of items with at least one request, and then when we consider the number of items with at least two requests (i.e., $j = 2$). We continue with this rationale until $j = H(c)$ is reached, because this is the maximum number of hosts associated with the considered logical broadcast channel c. Let $Prob(item \geq j)$ denote the probability that a random item gets more than j requests. Therefore

$$\begin{aligned} E[RNF] &= \sum_{j=1}^{H(c)} E[\max\{0; \#(I \geq j) - T\}] \\ &\geq \textstyle\sum_{j=1}^{H(c)} \max\{0; E[\#(I \geq j) - T]\} \\ &= \textstyle\sum_{j=1}^{H(c)} \max\{0; I \cdot Prob(item \geq j) - T\}. \end{aligned} \tag{3}$$

Next, we present an algorithm for the dynamic model. This algorithm is based upon the observation that the problem in this model can be divided into two parts: (a) deciding which hosts should listen to each logical broadcast channel; (b) deciding which items should be broadcast on each channel. For the second task, Algorithm 1, or any other algorithm for the static case, can be employed. Hence, in what follows we focus on a clustering algorithm that determines which hosts should be grouped into a single channel before determining which data items will be broadcast on each channel.

The algorithm tries to group the hosts into C sets in the most efficient way. The algorithm starts with H sets, where H is the number of hosts in the system. In each iteration of the algorithm, two sets are combined into one, until only C sets remain. Therefore, after the ith iteration of the

algorithm there are $H-i$ sets, and the algorithm terminates with C sets after the $(H-C)$th iteration. In each phase we seek to combine the two sets that represent hosts with "similar demands." We adopt the approach proposed in the past for modern databases that need to classify a large number of documents into sets according to their similarity. To this end, we define a "demand vector" $v(S)$ for each set S as follows:

$$\forall \text{ data item } i,\ v(S)[i] = \frac{\sum_{h \in S} m(h,i)}{\text{size\_of}(i)}.$$

That is, $v(S)[i]$ indicates the importance of item i to the hosts in set S. This importance is normalized by the size of item i in order to account for different item sizes. We now seek to combine the two sets S and S' whose demand vectors are most similar, because the nodes in S and S' are more likely to be interested in the same data objects. The concept of inner product can be used in order to find similar vectors. Namely, the two sets whose demand vectors have the largest inner product are considered the most similar and are therefore combined into a single larger set.

There are two related problems with this approach:

1. It is likely that a demand vector of a set that already contains a large number of nodes will have the greatest similarity with one of the other vectors. Consequently, the total demand of the hosts in such a set will exceed the bandwidth of a single channel.

2. It is likely that sets whose demand is very small (e.g., only one document) will not be merged with other sets. Consequently, the bandwidth of the channel allocated to such a set might be wasted.

These two problems may lead to unbalanced, and therefore sub-optimal, partitioning, where some of the sets have too many important items while other have too few to justify a single channel. Such a problem also occurs when the concept of vector inner product is used for the clustering of related documents in a large collection. A review of some commonly used normalization techniques can be found in [20]. The most common technique is *cosine normalization.* Using this technique, each vector $v(S)$ is normalized to have a unit L_2 norm.

In what follows each set S of nodes is represented by a *binary* vector $v[S]$. This vector has the value '1' for each item i required by any host in S. It does not matter how many hosts in S

require this item. We then normalize each vector to have a unit L_1 norm by dividing it by its norm $|v(S)| = \sum_i v(S)[i]$. Formally, this algorithm works as follows:

Algorithm 2 *A clustering algorithm for SMBC-D using a unit L_1 norm*

- *for every host h, create a set $S_h = \{h\}$*
- *repeat $H - C$ times (until the number of sets is C).*
  - *compute the demand vector $v(S)$ for every new set S as follows:*
    $\forall$ data item i, $v(S)[i] = 1$ if i is required by some host in S, and $v(S)[i] = 0$ otherwise.
  - *normalize the demand vector of every new set to have a unit norm by dividing it by $\sum_i v(S)[i]$.*
  - *for every two normalized vectors $v(S)$ and $v(S')$ whose inner product has not been computed so far, let $v(S, S') = v(S)^T \cdot v(S')$.*
  - *find the two sets S and S' with the maximum value of $v(S, S')$; combine these two sets into a single new one.* □

In terms of computational complexity, the creation of the initial H vectors requires $O(H \cdot I)$ operations and the computation of the inner product of each two vectors requires additional $O(H^2 \cdot I)$ operations. Then we have $H - C$ steps during which we need to find a pair of vectors for which the inner product is maximum, create a new normalized vector v' by combining these two vectors, and compute the inner product of this new vector v with the existing vectors. Since these operations are repeated for $H - C = O(H)$ steps, the overall complexity of Algorithm 2 is $O(H^2 \cdot I)$.

In practice, since these vectors are likely to be sparse, we can significantly speed up Algorithm 2 by computing the inner product of the sparse vectors and storing them in time and space proportional to the number of non-zero entries.

Next, we generalize Algorithm 2 for general norms:

Algorithm 3 *A clustering algorithm for SMBC-D using a general norm*
The algorithm is similar to Algorithm 2, except that instead of normalizing the demand vector to

have a unit L_1 norm, we normalize it to have a unit L_p norm[2]*. The L_p norm is denoted by $\|x\|_p$, where $\|x\|_p = \left(\sum_{i=1}^{n} \sqrt[p]{x_i}\right)^p$.* □

Finally, we generalize Algorithm 3 to the case where items are of variable size. The main idea is to represent every item in the demand vector not by a single bit, but by a number of bits proportional to its size. A careful implementation will not write these bits explicitly.

Algorithm 4 *A clustering algorithm for SMBC-D with variable size items*
The algorithm in similar to Algorithm 2, except that

- *in each vector $v[S]$ the value of entry $v[S]$ is determined as follows: $\forall$ data item i, $v(S)[i] = size\_of(i)$ if i is required by some host in S, and $v(S)[i] = 0$ otherwise.*
- *$v(S)$ is normalized to have a unit L_p norm.* □

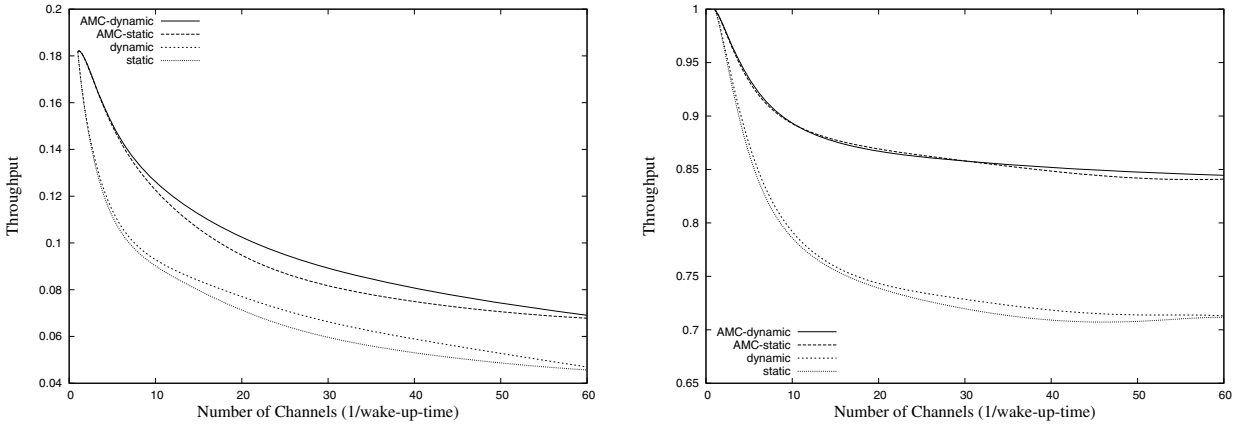
The algorithms for the AMC model are discussed in the next section.

5 Algorithms for the AMC Model and Simulation Results

In the following section we present algorithms for the AMC model, and simulations results for all the algorithms discussed in the paper. We built our own simulator, which measures the average profit of every algorithm during each multicast super-frame. In order to make the comparison between the various models and algorithms as "clean" as possible, we ignored some of the implementation-dependent issues. For example, we do not take into account the overhead required for telling each host when it should wake up in the dynamic and the AMC-dynamic models. We justify this omission by showing that even if this overhead is ignored, the advantage of these two models over the static and the AMC-static models is in any case not big enough to justify the extra complexity. We also ignore the overhead for studying the PHY layer quality of each host, and we assume no correlation between successive super-frames.

Many parameters can be determined by the simulator: the number of hosts H, the number of data items I, the distribution of requests, the distribution of profit, the bandwidth available for

[2]Technically, in order for the definition to be a mathematical norm, p must be ≥ 1.



(a) 2^{10} QPSK slots in a super-frame (b) 2^{16} QPSK slots in a super-frame

Figure 4: Performance of the algorithms vs. the number of logical broadcast channels (1/wake-up-time) for the uniform distribution

multicast, the data item size distribution, the distribution of the PHY quality of the hosts, and more. For this study, we selected a typical set of parameters and changed some of them mainly when we wanted to show that the change significantly alters the results (e.g., when the distribution of requests changes from uniform to Zipf).

Figure 4 shows the normalized throughput as a function of the number of logical broadcast channels (C) in the case where the total number of broadcast slots in each multicast super-frame is 2^{10} and for the case where it is 2^{16}. Recall that the number of channels is inversely proportional to the time during which a host needs to be active. With 1 channel, a host must be active at all times, whereas with 50 channels it needs to be active only 2% of the time. Recall also that the normalized throughput (y-axis) is defined as

$$\sum_{h,i}(m(h,i)\cdot r(h,i))/\sum_{h,i} m(h,i).$$

For the study illustrated in Figure 4 we consider $H = 64$ hosts and $I = 2^{16}$ potential fixed size data items. The distribution of requests is uniform, where for every m and h, $m(h,i) = 1$ with probability 0.022.

For the static model Algorithm 1 is used, after the base-station balances the number of hosts associated with each broadcast logical channel. For the dynamic model Algorithm 3 is used, with norm of $p = 0.5$. As discussed later, this value of p gave us the best results. For the AMC model, it

is assumed that the base-station distinguishes between hosts that can receive data only using QPSK and hosts whose channel is good enough to receive data using 16-QAM. We further assume that half of the hosts belong to the first category and half belong to the second. Different assumptions – e.g., regarding the distribution of QPSK hosts vs. 16-QAM hosts – would obviously lead to different results (see Figure 7).

When C is even, the base-station allocates half of the logical broadcast channels to the QPSK hosts and half of the channels to 16-QAM hosts. The number of slots in a 16-QAM broadcast logical channel is 2 times the number of slots in a QPSK channel: 2^{11} in Figure 4(a) and 2^{17} in Figure 4(b). Both graphs show 2 AMC curves. In the first one, denoted "AMC-static," the base-station maps the QPSK hosts to a QPSK channel and the 16-QAM hosts to a 16-QAM channel randomly and in advance. Of course, when the PHY quality on the downlink of a host changes, this host might have to be assigned to a different channel. However, this AMC variant is still considered static because the association between a host and a particular QPSK or 16-QAM channel is fixed, regardless of the data items requested by the host during every super-frame. After deciding which host should listen to which channel, the base-station uses Algorithm 1 in order to determine what data items should be transmitted on every channel. In the second AMC variant, denoted "AMC-dynamic," we use Algorithm 3 with norm of $p = 0.5$ to map QPSK hosts with similar requests to the same QPSK channel, and 16-QAM hosts with similar requests to the same 16-QAM channel. Afterwards, Algorithm 1 is used for each individual channel.

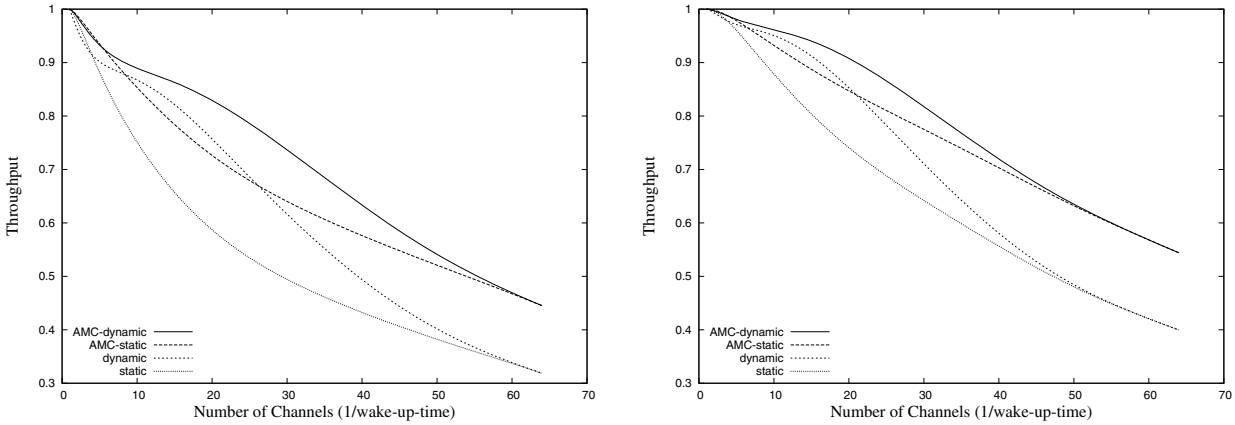
Consider Figure 4(a) first. The best results for the static and for the dynamic models are obtained, of course, when $C = 1$, namely, when all the hosts are active at all times. However, in that case the hosts expend their energy faster than in any other case. When the number of channels C increases, we see a drop in the throughput because the number of slots received by each host is proportional to $1/C$. We can also see that the dynamic algorithm does not contribute much. This is mainly because the uniform distribution of requests gives this algorithm only very limited ability to group hosts with many similar requests. Finally, we can see that the AMC model in this case is much more efficient than the dynamic model. The throughput of AMC with one channel is similar to that of the other models, because the single channel must have the most aggressive and least efficient

modulation and coding scheme. When the number of channels increases to two, the throughput of this model is maximum: one of the channels is for QPSK hosts, and the other for 16-QAM hosts. However, when the number of logical broadcast channels increases further, the throughput drops because there is nothing more to be gained by dividing the hosts into additional groups, while the regular penalty of each host receiving a smaller number of slots in a multicast super-frame is still incurred. It is also evident that AMC-dynamic is not much better than AMC-static.

Consider now Figure 4(b), where the number of slots in every frame increases by a factor of 64. For one channel all the algorithms achieve throughput of 1, because the number of slots is equal to the number of data items. However, when the number of channels increases, and the hosts expend less energy, the throughput decreases for all the models. This figure shows again minor differences between the static model and the dynamic model, as opposed to significant differences between these two models and their corresponding AMC variants. Unlike Figure 4(a), however, this figure shows the AMC schemes getting their maximum for one channel.

Data collected and analyzed by many researchers suggests that Web use follows a Zipf distribution. For example, in [2] the authors study statistics on the number of visitors to WWW sites by examining usage logs for 120,00 sites. They found that both in the case of all sites and the case of sites in specific categories, the distribution of visitors per site follows a universal power law. Similar findings are reported by [11] and many others. In order to understand the implication of the request distribution on the various models and algorithms discussed in this paper, we study the case where the request distribution follows a power law. To this end, we set the probability that the ith item is required to be proportional to $1/i$. We consider only 2^8 broadcast slots in every super-frame (2^9 if the frame is assigned to 16-QAM hosts), compared to 2^{10} or 2^{16} for the uniform distribution, because a larger number of slots easily gives a throughput of 1 regardless of the chosen scheme. This time we consider variable size data items. Three sizes were used: 4, 8 and 12 slots, with the same distribution. The merit associated with each packet was set to be proportional to the packet length. We used the same algorithms as for Figure 4, except that Algorithm 4 replaces Algorithm 3 (still with $p = 0.5$).

The results are shown in Figure 5. Because the model considered here is different from the one



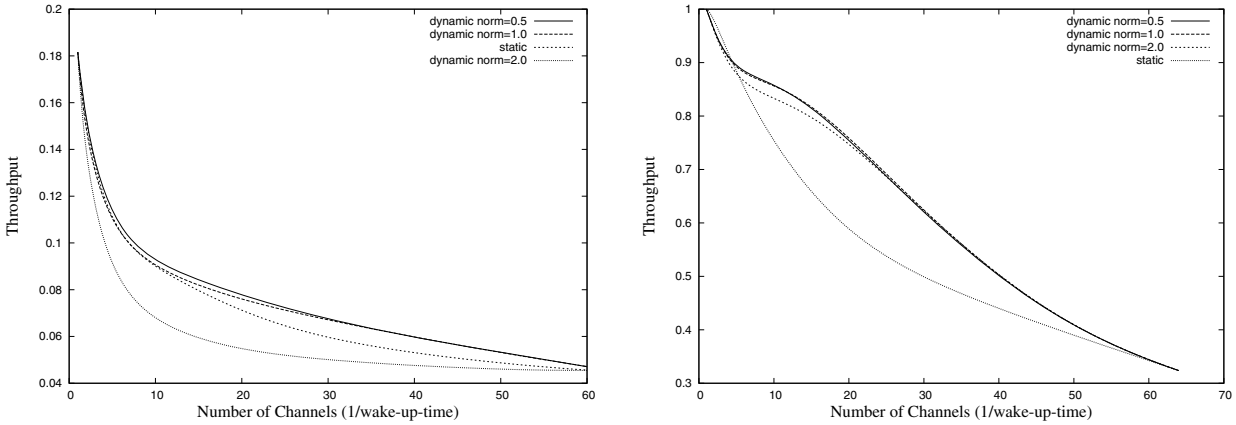
(a) number of items is 2^8 (b) number of items is 2^{12}

Figure 5: Performance of the algorithms vs. the number of logical broadcast channels (1/wake-up-time) for the Zipf distribution

considered earlier for the uniform distribution – in particular, the number of broadcast slots in every super-frame is different – we are interested mainly in comparing the differences in throughput for the different models.

We can see that the advantage of the dynamic algorithm over the static one is more tangible than for the uniform distribution of requests. Moreover, the drop of the two static curves with the number of channels is more substantial than the corresponding dynamic curves. This is because when the number of channels (and the size of each frame) is not too small, the dynamic algorithm is able to find good segmentations. In contrast, if the number of channels is small, the dynamic algorithm may be forced to merge not closely related vectors. This difference was not observed in the uniform distribution, where almost all segmentations are roughly the same.

As already indicated, for the dynamic model we used Algorithm 3 with $p = 0.5$ for the uniform distribution with fixed size items, and Algorithm 4 with $p = 0.5$ for the Zipf distribution with variable size items. To justify this selection, we show in Figure 6(a) the performance of Algorithm 3 and in Figure 6(b) the performance of Algorithm 4 for different values of p. All the other parameters for Figure 6(a) are similar to those considered in Figure 4 and the other parameters for Figure 6(b) are similar to those considered in Figure 5. Recall that the rationale behind Algorithm 3 and Algorithm 4 is to merge similar vectors. However, if two pairs of vectors have the same number of



(a) uniform distribution fixed size items (b) Zipf distribution variable size items

Figure 6: Performance of the dynamic algorithm vs. the number of logical broadcast channels (1/wake-up-time) for different norms

'1's in their intersection (inner product), then merging the pair whose total number of '1's is smaller is likely to yield better results when bandwidth is insufficient. This is better achieved with a norm of 0.5 or 1 than with a norm of 2.

In the previous graphs we assumed that half of the hosts have a good downlink channel, and are therefore capable of receiving their broadcast items using 16-QAM rather than QPSK. As already noted, the performance of the AMC model depends mainly on the distribution of 16-QAM vs. QPSK hosts. This can be seen clearly in Figure 7, which depicts the throughput for AMC-static under different distributions. In this case we consider again fixed size items with uniform distribution. The number of hosts is 64 and the total number of broadcast slots in a multicast super-frame is 2^{16}. The static model is equivalent to the case where all the hosts are only QPSK capable, and it is represented in the figure by the lowest curve. We can see that even if the number of 16-QAM capable hosts is only 1/3 of the total number of hosts, the throughput is up to 20% better than that of the static model. Note, however, that when the number of QPSK hosts is 5/6 and the number of channels is smaller than 10, the static algorithm is slightly better than the static-AMC algorithm. This is because when the number of channels is small, the bandwidth allocated to the 16-QAM hosts is more than sufficient, while it is insufficient for the QPSK hosts.

The AMC-dynamic and AMC-static approaches require that the base-station have accurate

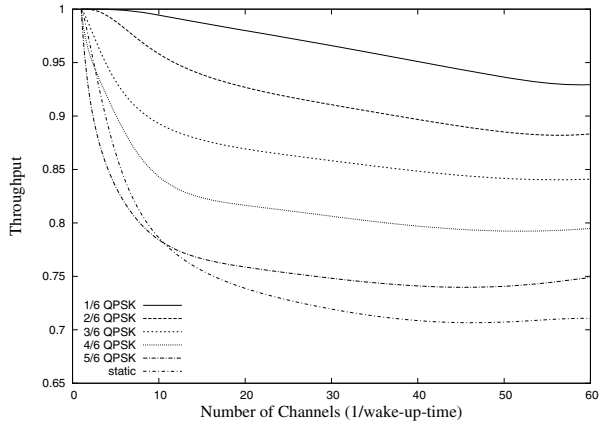


Figure 7: The throughput of the AMC model for different distributions of QPSK hosts

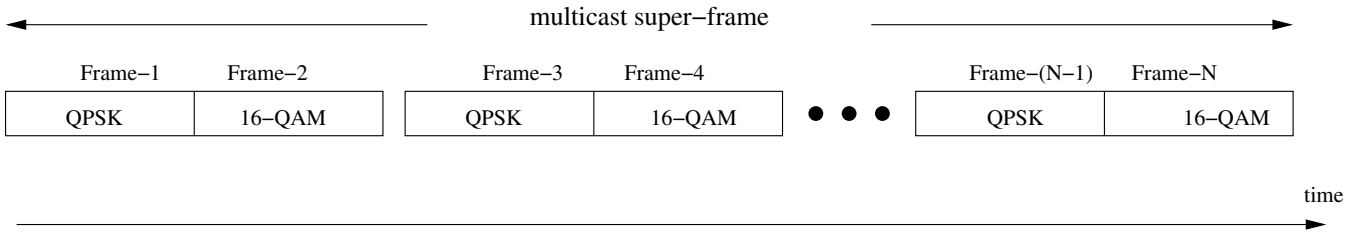
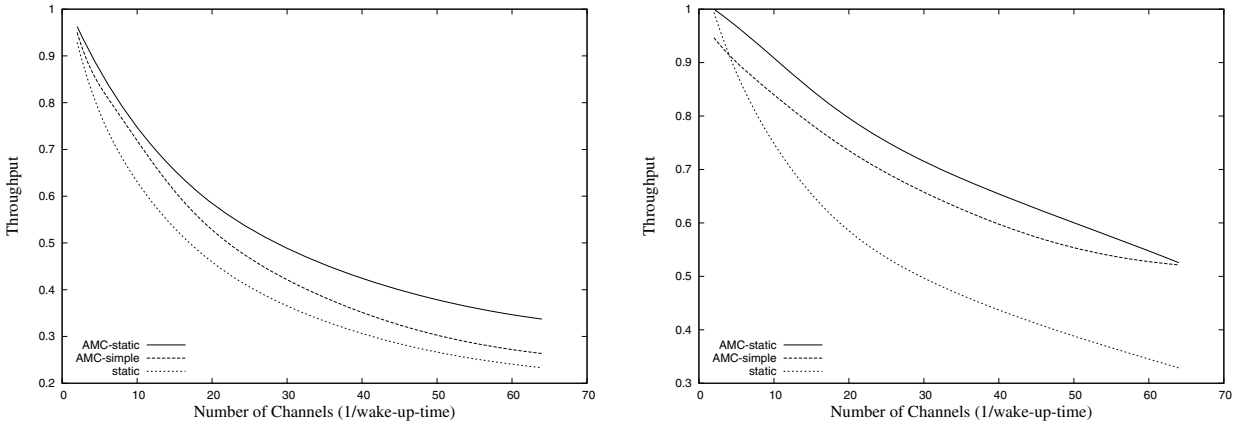


Figure 8: The AMC-simple scheme

information about the PHY quality of every receiving host before every super-frame starts. We now present an interesting variant for the AMC model, which does not require this knowledge. This variant is referred to as "AMC-simple." The idea is that the hosts are divided in advance into pairs of logical channels (see Figure 8). One channel in each pair is dedicated for QPSK receivers and one for 16-QAM receivers. The number of slots in the 16-QAM frame is of course 2 times that in the QPSK frame due to the spectral efficiency of 16-QAM. The base-station is *not* assumed to know the PHY quality of every receiver. For each pair it schedules the most profitable items in the QPSK channel, assuming that all the hosts will be listening to this channel. It then schedules the most profitable items in the 16-QAM sub-channel, assuming again that all the hosts will be listening to this sub-channel. In this way, the set of items scheduled to the QPSK channel in every pair is a subset of the items scheduled to the corresponding 16-QAM channel. Since every host knows its pair of channels and its PHY quality, it knows whether to wake up during the QPSK channel or the 16-QAM channel of this pair.



(a) 32 items, 1/2 of the users are QPSK (b) 256 items, 1/4 of the users are QPSK

Figure 9: Performance of simple-AMC vs. number of logical broadcast channels (1/wake-up-time)

Obviously, AMC-simple cannot perform as well as the AMC-static and AMC-dynamic schemes, because the decision about the items to be transmitted in every frame is not based on the exact requirements of the hosts listening to the corresponding channel. Rather, the decision is based on the requirements of a larger group of nodes. However, this scheme is much more practical than the other AMC variants, and for the Zipf distribution it offers a good trade-off between AMC scheme's performance and the non-AMC static scheme's simplicity. Figure 9 compares the results of the static scheme, the AMC-static scheme and the AMC-simple scheme, for the Zipf distribution and variable size items. The parameters here are similar to those considered in Figure 5, except that the number of items is smaller (32 in Figure 9(a) and 256 in Figure 9(b)) and, in Figure 9(b), there are only 1/4 QPSK hosts and 3/4 16-QAM hosts. We can see that when there are not many items (Figure 9(a)) AMC-simple performs much better than static scheme, but worse than the AMC-static scheme. When the number of 16-QAM hosts is relatively high, the performance of AMC-simple is even closer to that of AMC-static. When we increase the number of items, or decrease the number of hosts with good channel, or change the distribution of requests from Zipf to uniform, the performance of AMC-simple is much closer to that of the static scheme.

6 Conclusions

In this paper we addressed the trade-off between energy efficiency and throughput for multicast services in mobile networks. We proposed a scheme for sending multicast data to mobile hosts without requiring that they be active all the time. We considered N consecutive downlink frames as a single "multicast super-frame." For each host to be active only $1/C$ of the time, the base-station needs to send the multicast information destined for a particular host only in the specific N/C frame(s) in those multicast super-frames during which the host is active. The optimization problem we defined is what data should be transmitted in the broadcast region of each frame and when each host should be active in order to maximize the throughput for a given value of C.

We presented three different models for the association between hosts and channels: the static model, the dynamic model, and the adaptive modulation/coding (AMC) model. The last employs cross-layer optimization by dividing the hosts into groups according to the quality of their downlink PHY channels. We proved that for the case where data items are of variable length, the optimization problem for all the models is NP-complete. For the case where the items are of equal size, an optimal solution can be found for the static model using a polynomial-time algorithm, but for the dynamic and the AMC-dynamic models this problem is still NP-complete. We then presented algorithms for the three models and compared their performance. Our main conclusions are that the dynamic model indeed performs better than the static model, especially for the Zipf distribution, but it hardly justifies the significant added complexity. The AMC model performs significantly better than the other two models, even if only a small portion of the hosts have a good downlink PHY channel. Finally, the AMC-simple model offers a very good trade-off between performance and implementation cost, especially if the number of hosts with a good downlink channel is relatively large and the number of data items is small.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communication environments. In *Proceedings of ACM SIGMOD*, 1995.

[2] L. Adamic and B. Huberman. The nature of markets in the world wide web. *Quarterly Journal of Electronic Commerce*, 1:5–12, 2000.

[3] A. Afonso, F. Regateiro, and M. Silva. Dynamic data delivery to mobile users. In *10th International Workshop on Database and Expert Systems Applications, Florence, Italy*, September 1999.

[4] D. Aksoy, M. Franklin, and S. Zdonik. Data staging for on-demand broadcast. In *Proceedings of the 27th VLDB Conference*, 2001.

[5] K. Asatani and Y. Maeda. Access network architectural issues for future telecommunication networks. *IEEE Communications Magazine*, 36(8), August 1998.

[6] M. Bakhuizer and U. Horn. Mobile broadcast/multicast in mobile networks. *Ericsson Review*, (1), 2005.

[7] D. Barbara. Mobile computing and databases—a survey. *IEEE Trans. Knowledge and Data Eng.*, 11(1):108–117, 1999.

[8] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *SIGCOMM*, pages 56–67, 1998.

[9] M. Cagalj, J. Hubaux, and C. Enz. Minimum-energy broadcast in all-wireless networks: Np-completeness and distribution issues. In *The 8th ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2002.

[10] R. Cohen, L. Katzir, and D. Raz. Scheduling algorithms for a cache pre-filling content distribution network. In *INFOCOM'2002, NYC, NY*, June 2002.

[11] A. Downey. Evidence for long-tailed distributions in the internet. In *Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2001)*, November 2001.

[12] S. Hameed and Nitin H. Vaidya. Log-time algorithms for scheduling single and multiple channel data broadcast. In *Mobile Computing and Networking*, pages 90–99, 1997.

[13] C. Hsu, G. Lee, and A. Chen. A near optimal algorithm for generating broadcast programs on multiple channels. In *The 10th International Conference on Information and Knowledge Management, CIKM'01*, 2001.

[14] Institute of Electrical and Electronics Engineers Inc. IEEE Draft 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, April 2005.

[15] D. Katsaros and Y. Manolopoulos. Web caching in broadcast mobile wireless environments. *IEEE Internet Computing*, May-June 2004.

[16] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 2004.

[17] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Segmentation problems. *Journal of the ACM*, 51(2):263–280, March 2004.

[18] G. Muhl, A. Ulbrich, K. Herrmann, and T. Weis. Disseminating information to mobile clients using publish-subscribe. *IEEE Internet Computing*, May-June 2004.

[19] Nir Namman and Raphael Rom. Analysis of transmission scheduling with packet fragmentation. *Discrete Mathematics and Theoretical Computer Science*, (4):139–156, 2001.

[20] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Research and Development in Information Retrieval*, pages 21–29, 1996.

[21] Chi-Jiun Su and Leandros Tassiulas. Broadcast scheduling for information distribution. In *INFOCOM*, pages 109–117, 1997.

[22] D. Tosi. An advanced architecture for push services. In *Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03)*, 2003.

[23] J. Wieselthier, G. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast wireless networks. In *INFOCOM 2000*.

[24] J. Wieselthier, G. Nguyen, and A. Ephremides. Algorithms for energy-efficient multicasting in static ad hoc wireless networks. *ACM Mobile Networks and Applications (MONET)*, pages 226–263, June 2001.