This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

Opportunistic Finite Horizon Multicasting of Erasure-coded Data

Gek Hong Sim, *Member, IEEE,* Joerg Widmer, *Senior Member, IEEE,* and Balaji Rengarajan, *Member, IEEE*

Abstract—We propose an algorithm for opportunistic multicasting in wireless networks. Whereas prior multicast rate adaptation schemes primarily optimize long-term throughput, we investigate the finite horizon problem where a fixed number of packets has to be transmitted to a set of wireless receivers in the shortest amount of time – a common problem, e.g., for software updates or video multicast. In the finite horizon problem, the optimum rate critically depends on the recent reception history of the receivers and requires a fine balance between maximizing overall throughput and equalizing individual receiver throughput. We formulate a dynamic programming algorithm that optimally solves this problem. We then develop two low complexity heuristics that perform close to the optimal solution and are suitable for practical online scheduling. We further analyze the performance of our algorithms by means of simulation. They substantially outperform existing solutions based on throughput maximization or favoring the user with the worst channel, and we obtain a 30% performance improvement over the former and a 120% improvement over the latter in scenarios with Rayleigh fading. We further analyze the performance of the schemes under imperfect state information and observe an even higher improvement over the benchmark schemes.

Index Terms—Finite horizon multicast, wireless multicast, opportunistic multicast scheduling, completion time, imperfect feedback

1 Introduction

In recent years, multicasting data to mobile users (e.g., for the purpose of video streaming, video conferencing, IPTV, distribution of news and alerts, or application and operating system updates) has gained in importance. As an example, the most recent mobile network architecture, LTE, includes the evolved Multimedia Broadcast Multicast Service (eMBMS) specifically for the purpose of distributing data and mobile TV content in a cellular network. Since the amount of such traffic in cellular networks is increasing rapidly and wireless resources are scarce and costly, improving the efficiency of wireless multicast is of high practical relevance.

The most common method for wireless multicasting is broadcasting. The base station (BS) transmits at some fixed low rate or the rate supported by the worst receiver to ensure that all receivers are able to receive the multicast transmission. This exploits the wireless broadcast gain, allowing a single transmission to simultaneously serve all receivers. Opportunistic multicast scheduling (OMS) improves over plain broadcast by exploiting multiuser diversity [1]. Having the BS transmit at a rate higher than the broadcast rate to the subset of receivers that can receive at this rate may improve overall throughput and minimize broadcast delay [2]. The intuition is that in an environment with variable channels, receivers that are not served in one slot due to bad channel conditions will be served in later slots when their conditions improve, and thus over time all receivers will eventually receive all the data. Hence, there is a tradeoff between multicast gain and multiuser diversity gain. Specifically, whenever the BS transmits a packet, it is necessary to select a suitable transmit rate, i.e., modulation and coding scheme (MCS). The MCS determines the amount of data transmitted per time slot as well as the packet loss probability. More robust MCSs transport less data and are more likely to be decodable.

Selecting the transmission rate and thus the subset of receivers to multicast to is a complex problem that has been the focus of a range of OMS algorithms [3]. To simplify the scheduling problem and improve performance when multicasting data, such algorithms often use erasure codes that ensure that with high probability, each packet received by a receiver is useful (unless the receiver already decoded all data) [4], i.e., the identity of the received packets is unimportant. Fixed rate LDPC [5] or rateless LT or Raptor codes [6] are examples of such erasure codes that work well in practice.

Most of the existing OMS algorithms [1], [2], [4], [7]–[9] consider the *infinite-horizon* multicast problem, where the sender has an infinite number of packets to send. The goal of the optimization is thus to maximize the *long-term* throughput to *all* receivers. This setting is a good approximation for the case of multicasting very large files [10]. In the infinite-horizon problem, the average channel quality seen by an individual receiver is likely to be close to the actual average of the channel distribution (law of large numbers). Therefore, it is unlikely to

- *G.H. Sim is with the Telematics department of Engineering, University Carlos III, Madrid, Spain and IMDEA Networks Institute, Madrid, Spain. Email: allyson.sim@imdea.org*
- *J. Widmer is with IMDEA Networks Institute, Madrid, Spain. Email: joerg.widmer@imdea.org*
- *B. Rengarajan is with Accelera Mobile Broadband, Santa Clara, CA. Email: balaji.rengarajan@gmail.com*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

see large differences in the number of received packets among the receivers (over a sufficiently large amount of time) and the state of the system in terms of the number of received packets can be neglected.

In practice, however, multicasting data with a size on the order of only hundreds or several thousands of packets is much more common, particularly for mobile networks. For example, mobile applications, news, and operating system updates often have a size of hundreds of kilobytes to several megabytes. Also, when streaming video, it is common to apply erasure coding to blocks consisting of one or several groups of pictures (GOP) [11]. A GOP usually comprises a few hundred packets, depending on the video rate. Such block sizes are a suitable tradeoff between coding efficiency and playout delay (caused by the decoding delay).

In the common *finite-horizon* multicast problem considered in this paper, the main objective is to minimize the completion time, i.e., the time needed for all receivers to receive enough data to decode the current block. The relevant optimization criterion is thus the throughput achieved over the duration of a block, rather than the long-term average throughput. Depending on the scenario, algorithms that are optimal for the infinite-horizon problem may be far from optimal for the finite-horizon case, as shown in this paper.

The finite horizon multicast problem is inherently more complex than the infinite horizon counterpart. When multicasting an infinite amount of data among a homogeneous group of receivers (i.e., with the same average channel conditions), the optimum tradeoff between multiuser diversity and multicast gain only depends on the number of receivers and their current channel conditions. In expectation, differences in the amount of data received by the different receivers will even out over time and therefore do not have to be taken into account. In contrast, the optimum decision in the *finite horizon* case depends on the amount of data received thus far. More accurately, it depends on the amount of data each receiver still needs to obtain in order to decode the full block of data and thus complete the reception. Intuitively, in case a receiver lags behind in the reception when many other receivers are also still far from completing the block, the lagging receiver may catch up by itself and jointly maximizing throughput for all receivers may be the optimum decision. If, however, all other receivers are close to completion, optimizing the MCS only for the lagging receiver may be the optimum choice to minimize overall completion time, as all other receivers are likely to complete before the lagging receiver in any case. In summary, having to take into consideration the receiver state in the optimization makes it much more challenging to find optimal solutions for the finite-horizon multicasting problem.

The main contributions of our paper are as follows:

• We formalize the finite horizon OMS problem and propose a dynamic programming (*Dyn-Prog*) based solution that optimally adapts the MCS to minimize the *completion time*, the time at which all receivers have successfully received the required amount of data.

• The high complexity of *Dyn-Prog* renders this approach unsuitable for many practical scenarios. We thus propose a very simple state-based heuristic that selects the MCS that maximizes the instantaneous throughput of the receiver with the lowest number of packets received so far (called *Max-Min*).

• We further design a more complex adaptive algorithm that selects the MCS that results in the expected future system state that has the lowest expected completion time. We estimate completion time using a weighted Euclidean distance metric. The corresponding weighted completion time algorithm (*Weighted-CT*) measures the distance between the different possible future states and the final state where all receivers completed, with weights based on average throughput estimates of the receivers. To deal with imperfect state information, we further design an estimation-based version of the algorithm, *Weighted-CTe*. It estimates the distribution of the current channel state based on outdated past feedback information and predicts the evolution of receiver states.

• We compare the performance of our low-complexity heuristics to the optimal *Dyn-Prog* solution as well as to existing approaches that greedily maximize the throughput for all receivers and a broadcasting scheme that always transmits to all receivers. We analyze scenarios with homogeneous and heterogeneous receiver sets under a basic multi-state channel model as well as Rayleigh fading. Under Rayleigh fading and with up to 16 users, the *Max-Min* algorithm provides a performance gain of 95% over the broadcasting scheme and a gain of 15% over the throughput maximization scheme scheme. At a slight increase in complexity, the *Weighted-CT* heuristic performs very close to the optimal *Dyn-Prog* strategy, with performance gains of 120% and 30% over the broadcast and throughput maximization schemes, respectively. For scenarios with more than 16 users the achieved gains are even higher. In scenarios with imperfect information, *Weighted-CTe* achieves gains of up to 130% and 60% over the prior schemes in homogeneous and heterogeneous scenarios, respectively. We further implement an energy model for wireless multicasting and analyze the energy efficiency of the different algorithms.

Our paper is organized as follows. Section 2 reviews prior work. In Section 3 we discuss our system model, including the channel model and the MCS dependent packet loss model. In Section 4, we provide the dynamic programming-based optimal scheduling algorithm, together with a basic example to provide some intuition into how the algorithm trades off receiver throughput depending on the system state. To address the problem of state space explosion and high complexity of the dynamic programming solution, we propose two low-complexity heuristics in Section 5. Simulation results that compare the different algorithms in terms of completion time and energy consumption are presented in Section 6. Finally Section 7 concludes the paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

2 Related Work

The idea of OMS was pioneered by Gopala and Gamal [1] who studied the tradeoff between multiuser diversity and multicast gain. They analyzed the performance of three different scheduling mechanisms that adapt the transmit rate to the user with the best channel, the worst channel, and the median channel, respectively. In their follow-up paper [7], they investigated the performance achieved by serving a fixed fraction of users. This restriction is relaxed in [12] and [4]. In [12], Ge *et al.* initiate a transmission if the multicast threshold is satisfied. As the threshold is pre-determined, the transmission rate at each slot is fixed. This limits achievable throughput in case all the receivers have good channel conditions in a slot and higher rate could be used. Kozat *et al.* show that dynamic selection ratios that select more than 50% of the users can achieve higher throughput [4]. They also present an algorithm where the user selection ratio depends on the channels of the receivers. Both [12] and [4] exploit erasure coding for reliable transmission.

The authors of [8] propose algorithms with a static selection ratio (fixed for all transmissions) and a dynamic selection ratio (adapted to the instantaneous channels for each transmission) that maximize overall throughput. In [2], the authors extend their work of [8] from homogeneous to heterogeneous scenarios, composed of different groups of homogeneous users. A similar optimization algorithm for multicast throughput maximization is proposed in [9]. While all of these works target the infinite horizon case, in [13] the authors consider scenarios with a finite number of multicast packets. Using extreme value theory, they derive the optimal selection ratio for each transmission that minimizes completion time. In contrast to our work, their optimization algorithm does not consider the state of the receivers in terms of the number of received packets. Wang *et al.* [14] consider both channel and receiver state for the finite horizon problem using a disjoint formulation technique in which, in a slot, the MCS is chosen to serve the user or users with the highest priority, which is determined based on instantaneous throughput and remaining packets. However, the objective of this paper is to improve the fairness between the users rather than optimizing for throughput or completion time.

All of the above papers use a simple outage based channel model, where packet errors are deterministic. Receivers with channel conditions better than some threshold receive the packet and all other receivers lose the packet. In real wireless systems, packet errors are much more random and depend on noise and interference. In our model, we explicitly take the relationship between the channel conditions, the chosen MCS, and the probability of error into account. Also Ho *et al.* [15] take the probability of error into account in their formulation for maximizing the opportunistic multicasting gain. However, their schemes is a simple instantaneous throughput maximization and is thus not suitable for the finite horizon problem. In summary, all of the above algorithms – except those of [13] and [14] – focus on the infinite-horizon scenario and are sub-optimal for the finite-horizon case we consider in this paper.

The problem of minimizing the overall delay for all users to receive a certain number of packets is studied in [16] through a dynamic programming approach. This work does not consider erasure coding over a larger block of data, but repeatedly multicasts a single packet until each receiver has obtained it. The BS then multicasts the next packet in the same manner, and so on. The goal of the optimization algorithm is therefore to minimize the number of transmissions required to multicast a single packet to all receivers, and the state of the system is the number of receivers that did not yet receive the packet. The algorithm adapts its decision to the changes in the set of users that still need to receive the packet and maximizes the throughput for those users. The approach is mainly suitable for a single homogeneous group of users, since its complexity increases exponentially with the number of user groups in heterogeneous scenarios. The method of multicasting a single packet repeatedly is also much less efficient than multicasting blocks of erasure coded packets as is done above.

The most basic scheme against which we compare our proposed algorithms is the *Broadcast* algorithm (called LCG user rate in [3]), where the transmission rate is limited by the receiver that currently has the worst channel. This scheme ensures successful transmission to all receivers, but may sacrifice a lot of throughput when channels are highly variable. We further compare against a scheme called *Greedy* that optimizes the selection ratio at each transmission opportunistically based on the current channel states of all receivers so as to maximize total throughput. This mechanism has a performance that is indicative of the different selection ratio based throughput maximization algorithms above.

Opportunistic multicast algorithm highly depends on the availability of the instantaneous channel information from the users. The impact of limited feedback is examined in [15] and [17]. In [15], the MCS decision is made based on the average SNR. This assumption is too conservative since a realistic channel is also characterized by the path loss, Rayleigh fading and shadowing. Huang *et al.* [17] determine the transmission rate for the opportunistic multicast scheduling in [13] based on the recent channel conditions instead of the average SNR. Recent channels, however, do not always accurately reflect the instantaneous channel and it highly depends on the feedback interval, which is not explicitly defined.

In summary, our main contribution over prior work is the optimization of completion time for finite horizon multicasting. We further model the packet error rate rather than just considering outage, and evaluate the proposed algorithms under multipath Rayleigh fading. Lastly, we also propose an estimation based algorithm that accounts for imperfect receiver and channel state information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

3 System Model

We model the system as a time-slotted broadcast system with a single Base Station (BS) and N mobile users within the coverage area of the BS. Each user must receive a block of data of B bits, called the block size. We assume that due to erasure coding, each packet transmitted by the BS and received by a user is useful if the user has received less than B bits. In case multiple blocks of data are to be transmitted in succession, the BS will start transmitting the next data block only after all the receivers received the current block.

A time slot is of fixed duration. Thus, the BS broadcasts a fixed number of symbols per slot, which – depending on the MCS – corresponds to a variable number of bits. We assume that the BS can select one of the M MCSs, indexed by $m = 1, \ldots, M$. The number of bits per slot that can be transmitted using MCS m is denoted by R_m.

Perfect Channel State Information (CSI) and knowledge of the number of bits a user has successfully received is assumed to be available at the base station prior to the transmission in each time slot. The users see independent channel instances $h_i[k]$ at each time slot k. The discrete-time channel model for the received signal $s_i^{\text{rx}}[k]$ at user i is given by:

$$s_i^{\text{rx}}[k] = h_i[k]s^{\text{tx}}[k] + n_i[k], \tag{1}$$

where $s^{\text{tx}}[k]$ is the signal broadcast from the BS at time slot k, and $n_i[k]$ is additive white Gaussian noise term with power spectral density N_0.

For the analysis, and in particular for dynamic programming solution, we assume a discrete set of C possible channel realizations $\mathcal{H}_i$ for user i. However, the heuristics we develop also work with continuous channels. The probability of user i seeing channel coefficient $h_i \in \mathcal{H}_i$ in slot k is $\alpha_i(h_i)$, i.e., $\alpha_i(h_i) = \text{P}(h_i[k] = h_i)$. The corresponding SNR is denoted by γ_{h_i}. The vector of channels of all users, also referred to as the channel combination, is denoted by $\mathbf{h} = \{h_i, i = 1 \ldots, N\}$. We denote by $\mathcal{H}$ the set of all possible channel combinations, and by $\alpha(\mathbf{h}) = \prod_{i=1}^{N} \alpha_i(h_i)$, the probability of a channel combination $\mathbf{h} \in \mathcal{H}$. Note that the total number of channel combinations is C^N.

In contrast to prior work, we do not assume deterministic channel outage but use the packet error rate (PER) for a given channel quality and MCS from [18]. For a channel instance $h_i \in \mathcal{H}_i$, the PER for user i under MCS m is represented by $p_i^m(h_i)$, and the corresponding packet success rate (PSR) is $q_i^m(h_i) = 1 - p_i^m(h_i)$.

In this paper, we use the following terms: *(1)* A *strategy* g specifies the MCS $g(\mathbf{h})$ for each channel combination $\mathbf{h} \in \mathcal{H}$. Hence, the total number of strategies is $S = M^{C^N}$. We denote by $\mathcal{G}$, the set of all possible strategies. *(2)* The *state* consists of the vector of the number of bits received by each user i, denoted by $\mathbf{x} = \{x_i, i = 1, \ldots, N\}$. The state space $\mathcal{X}$ consists of all states where the number of bits received by all users is positive and less than or equal to B.[1] The *initial state* where none of the users have any information is $\mathbf{x}^0$ and the end state where all the users have received B bits is denoted by $\mathbf{x}^B$. *(3)* A *policy* μ maps any given state $\mathbf{x} \in \mathcal{X}$ to the strategy $g_{\mathbf{x}}^{\mu}$ to be used in that state. *(4)* The *expected completion time* $D_\mu(\mathbf{x})$ is the mean time required to get from state $\mathbf{x}$ to the end state $\mathbf{x}^B$ under policy μ.

4 Optimization Problem

In this section, we consider the case of memoryless channels and formulate the problem as a stochastic shortest path problem [19] with cost per stage equal to 1 (the time needed per slot is fixed, $\tau = 1$) and no terminal cost. We assume that the probability of successfully receiving a packet is non-zero for every combination of MCS and channel condition, though it might be extremely low for some combinations.

4.1 Dynamic programming solution (*Dyn-Prog*)

Let $\mathcal{E} = \left\{\mathbf{e} \,\middle|\, |\mathbf{e}| = N, e_i \in \{0, 1\}\right\}$ be the set of all vectors of size N whose components take values 0 or 1. The transition probability from state $\mathbf{x} \in \mathcal{X}$ to state $\mathbf{y} \in \mathcal{X}$ when MCS m is used under channel combination $\mathbf{h}$ is given by:

$$\rho_{\mathbf{h}}^m(\mathbf{x}, \mathbf{y}) = \sum_{\substack{\min(\mathbf{x}+R_m\mathbf{e}, B)=\mathbf{y} \\ \mathbf{e}\in\mathcal{E}}} \left(\prod_{i=1}^{N} p_i^m(h_i)^{e_i} q_i^m(h_i)^{1-e_i} \right), \tag{2}$$

where the above minimization is defined element-wise. Note that in case every user experiences an erasure, the state remains unchanged.

The state space is finite, and there clearly exists a finite integer K such that there is a positive probability of terminating after K steps irrespective of the policy. Thus, the optimal policy μ^* satisfies Bellman's equations for every state $\mathbf{x}$:

$$D_{\mu^*}(\mathbf{x}) = \min_{g\in\mathcal{G}} \left(\tau + \sum_{\mathbf{h}\in\mathcal{H}} \alpha(\mathbf{h}) \sum_{\mathbf{y}\in\mathcal{X}} \rho_{\mathbf{h}}^{g(\mathbf{h})}(\mathbf{x}, \mathbf{y}) D_{\mu^*}(\mathbf{y}) \right), \tag{3}$$

and the optimal strategy in state $\mathbf{x}$ is given by

$$g^{\mu^*}(\mathbf{x}) = \operatorname*{argmin}_{g\in\mathcal{G}} \left(\sum_{\mathbf{h}\in\mathcal{H}} \alpha(\mathbf{h}) \sum_{\mathbf{y}\in\mathcal{X}} \rho_{\mathbf{h}}^{g(\mathbf{h})}(\mathbf{x}, \mathbf{y}) D_{\mu^*}(\mathbf{y}) \right). \tag{4}$$

Since the state space is finite, there are several options to solve for the optimal policy as well as the minimum expected completion time. We choose a simple value iteration approach. Starting from the end state $\mathbf{x}^B$, we use Bellmann's equation Eq. 3 to determine the completion

1. Note that to reduce the state space, we can use a normalized block size that is measured in units of the greatest common divisor of the MCS rates instead of in bits together with a corresponding normalized rate. As an example, for a block size of $B = 1800$kbits, $M = 2$ and MCSs with rates of 6Mbps and 9Mbps, the normalized block size is $B = 600$ and the normalized rate is $R_1 = 2$ and $R_2 = 3$, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

times of the states that only depend on the end state (for which the completion time is known to be 0). We then proceed in the same manner to determine the expected completion times of states that only depend on states for which the completion time is already known, until the completion times for all states are computed. This process also yields the optimum policies from Eq. 4.

4.2 A simple two user example

Consider a scenario with two users ($N = 2$), with identically distributed channels. Let $\mathcal{H}_1 = \mathcal{H}_2 = \{L, H\}$, and $\mathcal{H} = \{HH, HL, LH, LL\}$, where L and H denote channels with low and high channel quality, respectively. The base station can choose one of three MCSs in each slot. The probability of packet error when MCS m is used is denoted by $p^m(L)$ and $p^m(H)$ for both users under the low and high channel, respectively. A strategy is defined by specifying the MCS to be used for each vector channel in $\mathcal{H}$.

Here, Bellman's equation at state $\{x_1, x_2\}$ is:

$$\begin{aligned} &D_{\mu^*}(\{x_1, x_2\}) = \\ &\tau + \min_{g \in \mathcal{G}} \sum_{\mathbf{h} \in \mathcal{H}} \alpha(\mathbf{h}) \Big(p^{g(\mathbf{h})}(h_1) p^{g(\mathbf{h})}(h_2) D_{\mu^*}(\{x_1, x_2\}) \\ &\quad + p^{g(\mathbf{h})}(h_1) q^{g(\mathbf{h})}(h_2) D_{\mu^*}(\{x_1, \min(x_2 + R_{g(\mathbf{h})}, B)\}) \\ &\quad + q^{g(\mathbf{h})}(h_1) p^{g(\mathbf{h})}(h_2) D_{\mu^*}(\{\min(x_1 + R_{g(\mathbf{h})}, B), x_2)\}) \\ &\quad + q^{g(\mathbf{h})}(h_1) q^{g(\mathbf{h})}(h_2) D_{\mu^*}(\{\min(x_1 + R_{g(\mathbf{h})}, B), \\ &\qquad\qquad \min(x_2 + R_{g(\mathbf{h})}, B)\}) \Big). \end{aligned}$$

We evaluate the optimal policy in a scenario where the H and L channels for the users are $\mathcal{H}_1 = \mathcal{H}_2 = \{5\text{dB}, 28\text{dB}\}$. The probabilities of L and H are $\alpha_1(L) = \alpha_2(L) = 0.75$ and $\alpha_1(H) = \alpha_2(H) = 0.25$. We choose such a highly variable channel, since it makes it easier to demonstrate the tradeoffs that the algorithm makes in the different regions of the state space. The probability of each channel combination in $\mathcal{H}$ can be easily obtained by multiplying the respective channel probabilities. For simplicity, we use $M = 3$ MCSs with normalized rates of $R_1 = 1$, $R_2 = 4$ and $R_3 = 9$. The PER for each MCS and channel instance is listed in Table 1.

TABLE 1
PER for different MCS and SNR value pairs

$p_i^m(h_i)$	$m = 1$	$m = 2$	$m = 3$
$\gamma_H = 28dB$	0	0	0.08
$\gamma_L = 5dB$	0.23	0.97	1

In Fig. 1, a drift vector (arrow) reflects the optimal policy at that state. It shows the expected future state, given the optimum MCSs chosen for the different channel combinations. Hence, the length of a vector indicates the throughput obtained by the corresponding policy. (Note that for better readability, we only plot policy vectors for a subset of states and increase their lengths.) We set the normalized block size $B = 200$. At the initial state $\mathbf{x}^0 = \{0, 0\}$ the optimal policy is $g(\mathbf{h}) = \{1, 3, 3, 3\}$, i.e., MCS $m = 1$ is used for channel combination LL and MCS $m = 3$ is used for channel combinations LH, HL, and HH. This particular policy is a greedy policy which gives the maximum throughput to both users. This policy is also used in almost all states up to $\mathbf{x} = \{140, 140\}$. Closer to the boundaries of the state space, the policy changes from greedy to increasingly favoring the user that is lagging behind. This accounts for the fact that the leading user is likely to finish before the trailing user, even if MCS decisions are optimized for the trailing user. It aims to prevent the loss of multiuser diversity caused by one user finishing early. For $\{140 < x_1 \leq 160, x_2 < 140\}$ and $\{x_1 < 140, 140 < x_2 \leq 160\}$, the predominant policies are $g(\mathbf{h}) = \{1, 3, 2, 3\}$ and $g(\mathbf{h}) = \{1, 2, 3, 3\}$, respectively, where a more conservative MCS is chosen when the trailing user has a low channel quality. This policy sacrifices throughput to prevent the trailing user from falling further behind. Even closer to the boundaries, the policies are $g(\mathbf{h}) = \{1, 3, 1, 3\}$ and $g(\mathbf{h}) = \{1, 1, 3, 3\}$, further trading off overall throughput for a higher packet reception probability for the trailing user. When both users received a similar number of bits and are close to the end state $\mathbf{x}^B$, the algorithm also chooses a more conservative MCS indicated by a shorter arrow length to avoid overshooting (i.e., unnecessarily delivering more than B bits to both users).

While solving the stochastic shortest path problem minimizes average completion time and provides the optimal policy, the size of the state space and the computational complexity increase exponentially with N, the number of users. Therefore, the above approach is not practical for actual implementation in a network.

5 State-Aware Heuristics

Due to the high complexity of the *Dyn-Prog* algorithm introduced in the previous section, we propose two low-complexity heuristics that mimic its behavior.

5.1 Maximize minimum throughput (*Max-Min*)

This heuristic is based only on the current state and the current channel conditions. At each slot, the user with the least number of received bits is identified as the worst user, who is most likely to require the highest number of slots to receive all data. Note that this is indeed the case when the users are homogeneous and have an identical channel distributions. In the case of heterogeneous users, the users that have channel conditions that are worse (on average) are more likely to be the trailing users and thus most likely to finish last. In each slot, the algorithm uses the MCS that maximizes the throughput for the trailing user. If both users have the same number of bits, the algorithm greedily maximizes sum throughput for both. Note that since the algorithm ignores by how much the worst user is trailing, it may react too conservatively in case the difference between users is small.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

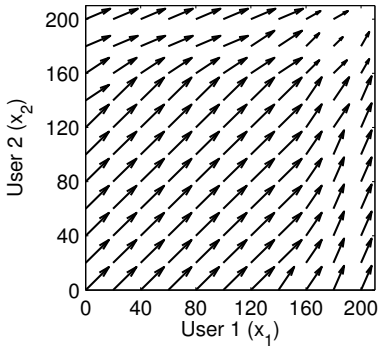


Fig. 1. Policy given by the (*Dyn-Prog*) algorithm

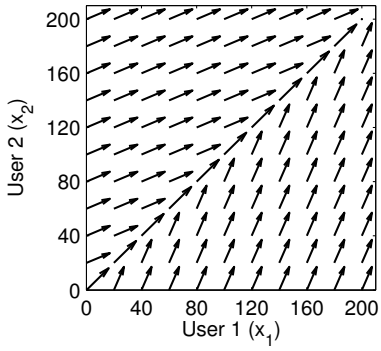


Fig. 2. Policy given by the *Max-Min* algorithm

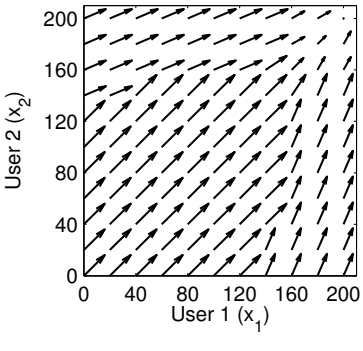


Fig. 3. Policy given by the *Weighted-CT* algorithm

Fig. 2 depicts the average drift resulting from such a policy in a scenario with the same parameter setting as explained in Section 4.2 for homogeneous users. There are two predominant strategies that are used for all states off the diagonal. As *Max-Min* sacrifices overall throughput in favor of the trailing user as soon as a user falls behind. The resulting strategies are $g(\mathbf{h}) = \{1, 3, 1, 3\}$ and $g(\mathbf{h}) = \{1, 1, 3, 3\}$. On the diagonal, *Max-Min*'s sum throughput maximization leads to the same strategy as in the *Dyn-Prog* solution, except for the last states before finishing. Since in contrast to *Dyn-Prog*, *Max-Min* does not explicitly take expected completion time into account, it does not switch to more conservative symmetric strategies of $g(\mathbf{h}) = \{1, 2, 2, 2\}$ and $g(\mathbf{h}) = \{1, 1, 1, 1\}$, respectively. The latter would deliver (with a lower packet loss probability) just the required number of bits to finish, compared to using the highest MCS $m = 3$, which may deliver more bits than necessary or result in packet loss.

Overall, we note that compared to the optimal *Dyn-Prog* algorithm, *Max-Min* is more conservative and ensures that the progression of state is with high probability along the diagonal where both users have the same number of bits.

5.2 Weighted completion time (*Weighted-CT*)

In many cases, favoring the trailing user is overly conservative. In particular, when the number of pending bits is large for all users and the relative lag is small, the probability that the currently trailing user actually finishes last may be small. We now present a heuristic that more closely models the decisions taken by the *Dyn-Prog* algorithm to achieve a better tradeoff between instantaneous sum throughput and balancing the number of pending bits (i.e., the state) for the different users.

At a given slot k, with state $\mathbf{x}$ and channel $\mathbf{h}$, we evaluate the average drift and determine the expected next state, $\mathbf{y}^m$, when using MCS m as:

$$y_i^m = x_i + q_i^m(h_i[k])R_m,\ i = 1, \ldots, N \tag{5}$$

Note that the expected state may be real valued. We then estimate the additional time required, on average, for all users to receive B bits given that they are in state $\mathbf{y}^m$. Since computing the actual estimated remaining completion time is computationally intensive as discussed in the previous section, we use a weighted Euclidean distance metric instead. The distance is taken as the bits required for completion, divided by a weight that reflects the average rate at which a user progresses through the state space. The estimated completion time $\tau_{\mathbf{y}^m}$ is thus:

$$\tau_{\mathbf{y}^m} = \sqrt{\sum_{i=1}^{N} \left(\frac{B - y_i^m}{w_i} \right)^2} \tag{6}$$

and the chosen optimum MCS is $m^* = \arg\min_m \tau_{\mathbf{y}^m}$.

We choose weight w_i that is proportional to the average throughput achieved by the user under a hypothetical policy that chooses the MCS that maximizes the rate of each channel:

$$w_i = \sum_{h_i \in \mathcal{H}_i} \max_m \alpha_i(h_i) q_i^m(h_i) R_m\,. \tag{7}$$

As the actual choice of MCSs in the algorithm in turn depends on the policy (and thus on the states as well as the channels of the other users), it is hard to determine the true average rate. At the same time, it is not necessary to estimate this rate very accurately; it is only necessary to obtain approximately the right relative differences in user's completion time estimates that lead to the correct choice of MCS m^*. The hypothetical policy to determine the weights above is a very simple but effective method to capture these approximate relative throughput differences among the users.

In practical scenarios, the channel distribution of individual users may not be known in advance. Further, the channel statistics of a mobile user may change over time. In such settings, we use an exponentially weighted moving average to track the user's weight. Given an instantaneous channel instance $h_i[k]$ at slot k, the estimated weight of user i, $\hat{w}_i[k]$, is given by:

$$\hat{w}_i[k] = (1 - \beta)\hat{w}_i[k-1] + \beta \sum_{m=1}^{M} q_i^m(h_i[k])R_m\,, \tag{8}$$

where β is a sufficiently small constant.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

Fig. 3 shows the drifts for the *Weighted-CT* algorithm. Our choice of weights indeed captures well the relative desirability of the different states. While the set of strategies is not as rich as with the *Dyn-Prog* approach – in particular at the transition between the greedy throughput maximization strategy and the more conservative border strategies – the strategies in most of the state space are almost the same. Most importantly, this is true for states around the diagonal *which are much more likely to occur in reality than states far off the diagonal* where the number of bits for the two users differs a lot.

5.3 Weighted-CT with rate estimation (*Weighted-CTe*)

In the previous sections, we assumed that perfect channel and state information is available at the BS. In practice, however, feedback from receivers is delayed and reporting channel and receiver states information incurs overhead and can thus only be done periodically. To deal with such imperfect and outdated information, we design an estimation-based version of the algorithm, *Weighted-CTe*. It estimates the probability distribution of the current channel state based on the outdated past feedback and predicts the evolution of receiver states.

Assume that σ is the delay between the actual channel measurement at user i and the use of that information at the sender. The sender can now estimate the probability of the *current* channel $h_i[k]$ having the channel state h_i, given the outdated channel information $h_i[k-\sigma]$, as

$$\hat{\alpha}_i^k(h_i) = \mathrm{P}\left(h_i[k] = h_i \in \mathcal{H}_i \mid h_i[k-\sigma]\right) . \tag{9}$$

The estimated reception probability of user i when MCS m is used is:

$$\hat{q}_i^m(h_i[k-\sigma]) = \sum_{h_i \in \mathcal{H}_i} \hat{\alpha}_i^k(h_i) q_i^m(h_i), \tag{10}$$

With this, Eq. 5, Eq. 7 and Eq. 8 can be rewritten, replacing the actual reception probability for a known channel $q_i^m(h_i[k])$ with the estimated reception probability $\hat{q}_i^m(h_i[k-\sigma])$.

6 Results

In this section, we evaluate the performance of our proposed algorithms in homogeneous and heterogeneous user scenarios and compare them to the existing *Broadcast* and *Greedy* schemes discussed in Section 2. For simple scenarios ($N = 2$ and $C = 2$), we also compare our results to the optimal *Dyn-Prog* solution. We start with simple scenarios to provide an intuition for the algorithms that helps to better understand the more complex scenarios. We then study the impact of block size B and the number of users N on the performance of the algorithms under multipath Rayleigh fading channels with the ITU Pedestrian B path loss model [20]. The Doppler frequency is 10 Hz and the coherence time is $t_c = 40$ ms. Given that multicast traffic will be sent concurrently with other unicast data traffic, only some of the slots of an LTE frame can be used for multicast. In the simulations with Rayleigh fading, we use two out of the ten slots (or sub-frames) of a frame for multicast. Each slot has a duration of 1 ms and thus the transmission time interval (TTI) is equivalent to 5 ms. Finally, we analyze the performance in a more practical scenario with limited and imperfect feedback from the users.

The algorithms are evaluated with two performance metrics: the system completion time D (in all scenarios) and the energy consumption (in the multipath Rayleigh fading scenarios). The completion time D corresponds to the time required for all of the users in the system to receive the whole block of data. It is measured in slots of 1 ms. The energy consumption is measured in Joule.

In all of the simulations, we consider three modulation schemes: QPSK, 16-QAM, and 64-QAM, with channel coding and data rates that are corresponding to CQI=3 to CQI=15 in [18]. The MCS determines the number of transmitted bits and the PER for the instantaneous channel quality in a slot. The block size B used throughout this section is 6400kbits unless otherwise specified. This block size roughly corresponds to the size of a group of pictures (GOP) for a video with DVD quality [21].

6.1 Completion time comparison to the optimal *Dyn-Prog* solution in simple scenarios

In this section, we first analyze the performance of the different algorithms in a simple $N = 2$ user scenario with $C = 2$ channel instances and $M = 13$ MCSs (CQI $= 3$ to CQI $= 15$ in [18]). In such a simple scenario, it is possible to obtain the optimum *Dyn-Prog* solution. The system model that we use here is the one described in Section 4.2. We analyze both homogeneous and heterogeneous user scenarios. In these scenarios, we set the stationary channel probabilities for the H and L channel to $\alpha(H) = 0.25$ and $\alpha(L) = 0.75$, respectively.

6.1.1 Homogeneous network

In the homogeneous scenario, both users have the same channel statistics but independent channel instances. We present the results of increasing the channel variability δ, where δ is the SNR difference between the H and L channel of each user. Here, the lowest δ is 0.7dB ($\gamma_H = 9.0$dB and $\gamma_L = 8.3$dB) and the highest δ is 20.0dB ($\gamma_H = 21.0$dB and $\gamma_L = 1.0$dB). The H and L channel pair of a user is picked such that the average throughput the user would obtain in a single-user scenario does not change. As δ increases, the SNR of the H channel increases and the SNR of the L channel decreases.

Fig. 4 shows how channel variation impacts the performance of the algorithms in a homogeneous scenario. In this scenario, both *Greedy* and *Weighted-CT* perform close to the optimal *Dyn-Prog* solution. As the users have the same channel distribution, exploiting opportunistic gain and maximizing the instantaneous throughput as *Greedy* does is a good strategy. The low complexity

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

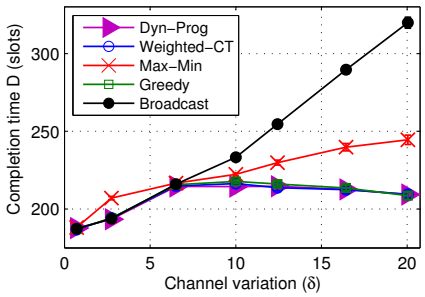


Fig. 4. Homogeneous network with increasing channel variability δ for $N = 2$ and $B = 6400$kbits.

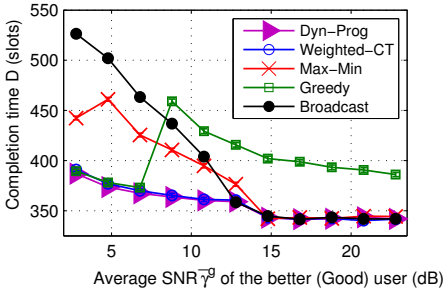


Fig. 5. Heterogeneous network with increasing heterogeneity for $N = 2$ and $B = 6400$kbits.

heuristic, *Weighted-CT*, weighs the homogeneous users equally according to Eq. 7 and transmits with the MCS that leads to the expected future state with the lowest completion time using Eq. 5 and Eq. 6. *Weighted-CT* trades off the instantaneous opportunistic throughput and the homogeneity of the receiver states to minimize the completion time in a manner similar to *Dyn-Prog*. It thus performs slightly closer to *Dyn-Prog* than *Greedy*. In contrast to *Greedy*, *Weighted-CT* exploits opportunistic multicast less aggressively in case this leads to one of the users trailing too far behind. However, since the users are homogeneous, this does not happen often and therefore the performance differences are small.

Broadcast performs worse than the other schemes because its transmission rate is limited by the lowest instantaneous channel. The impact is more pronounced when δ is larger since the SNR of the L channel is lower. Generally, *Max-Min* performs worse than the other schemes but *Broadcast*. In case the trailing user has a better instantaneous channel than the other user, *Max-Min* transmits at a higher rate than *Broadcast*, which is beneficial in a homogeneous scenario since exploiting opportunistic gain is a good strategy. However, when the trailing user has a worse instantaneous channel, *Max-Min* may send at the broadcast rate, and thus performs worse than *Dyn-Prog*, *Weighted-CT*, and *Greedy*. For $\delta = 3.7$dB, transmitting at the broadcast rate is the right decision. Here, all schemes transmit at the broadcast rate except for *Max-Min*, which thus performs worse.

Note that the completion time of *Greedy*, *Weighted-CT* and *Dyn-Prog* is not constant – it first increases, then slightly decreases – although the hypothetical single user throughput would be equal for the different δ, as described in the setup above. This change is due to the fact that the algorithms select the transmit rate depending on the channel instances of both users and thus the distribution of transmit rates is different from the single-user case.[2]

2. For example, in the single-user case, $\mathcal{H} = \{H, L\}$ and the probability to transmit at the rate that corresponds to H is $\alpha(H) = 0.25$ and $\alpha(L) = 0.75$ for L. In the two-user case, $\mathcal{H} = \{HH, HL, LH, LL\}$. Here, maximizing the opportunistic gain is optimal and the probability to transmit at the rate for H is $\alpha(H) = \alpha(HH) + \alpha(HL) + \alpha(LH) = 0.44$ and at the one for L is $\alpha(L) = \alpha(LL) = 0.56$.

6.1.2 Heterogeneous network

In a heterogeneous channel scenario, the *good* user (g) has a better *average* channel than the *bad* user (b). In Fig. 5, we evaluate the completion time when increasing the average SNR of the *good* user $\bar{\gamma}^g$ from 2.8dB to 22.8dB and fixing the average SNR of the *bad* user $\bar{\gamma}^b$ at 2.8dB. Higher average SNR results in a higher rate, thus as an overall trend the completion time decreases. Here, the difference between each the H and L channels of both users is always $\delta = 12.4$dB.

On the left extreme of Fig. 5 (at $\bar{\gamma}^g = 2.8$dB), the users are homogeneous and the relative performance of the algorithms is as discussed in Section 6.1.1. *Weighted-CT* performs close to the optimal *Dyn-Prog* for all $\bar{\gamma}^g$. This confirms that also for heterogeneous scenarios, the computation of y_i^m and $\tau_{\mathbf{y}}^m$ based on the average rate estimate w_i leads to MCS decisions almost identical to those of *Dyn-Prog*. When $\bar{\gamma}^g$ is low (users are homogeneous), it maximizes aggregate user throughput, whereas when $\bar{\gamma}^g$ is high (users are heterogeneous), it is more conservative towards the *bad* user.

When the difference between the *good* and the *bad* user is sufficiently large (at $\bar{\gamma}^g = 8.8$dB), *Greedy*'s completion time increases drastically and for higher $\bar{\gamma}^g$ it performs worse than the other algorithms. For these high channel differences, serving the good user alone provides a higher sum throughput than serving both users. Therefore *Greedy* serves users sequentially – first the *good* user at a high rate until the user finishes and only then the *bad* user. In contrast, *Broadcast* and *Max-Min* schemes which always favor the trailing user outperform the *Greedy* scheme. *Broadcast* is the best strategy when it is optimal to only serve the user with the worst channel (for $\bar{\gamma}^g \geq 14.8$dB). At $\bar{\gamma}^g = 12.8$dB, *Max-Min* performs slightly worse than *Broadcast* because with some small probability the *good* user may still trail, causing the algorithm to transmit at a too high rate resulting in a very low PSR at the *bad* user. *Max-Min* experiences an increase in D for $\bar{\gamma}^g = 4.8$dB for the same reason.

To further analyze the algorithms, we show the instantaneous sum throughput at each time slot, averaged over 200 simulation runs. Instantaneous sum throughput is the total throughput of the users remaining in the system

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

in a given slot. Fig. 6 shows a homogeneous scenario, corresponding to $\bar{\gamma}^g = 2.8$dB in Fig. 5. Fig. 7 shows a heterogeneous scenario, corresponding to $\bar{\gamma}^g = 12.8$dB in Fig. 5. Note that the user throughput is zero for a user that has already received all the data.

It is optimal to exploit opportunistic gain and serve the better user when the users are homogeneous. Therefore, Fig. 6 shows that *Broadcast* achieves a lower sum throughput than the other schemes because its rate is limited by the worst instantaneous channel. Consequently, the users finish later than the other schemes (taking between 500 to 600 slots). *Max-Min* transmits at a higher rate when the trailing user has a better channel and thus achieves higher sum throughput than *Broadcast*. As discussed before, the performance of *Weighted-CT* and *Greedy* is close to the *Dyn-Prog* scheme, which is also evident in the sum throughput curves.

Fig. 7 shows the sum throughput of the schemes when users are heterogeneous. At time slots ≤ 100, *Greedy* has the highest sum throughput because it first transmits at a high rate to the *good* user only, to maximize opportunistic gain. The *good* user can be served at around three times the rate of the *bad* user. Consequently the sum throughput drops significantly in time slot ≈ 100 when the *good* user finishes, leaving only the *bad* user in the system. Clearly, in this scenario serving primarily the *bad* user is the better strategy, since also the *good* user will receive those data.

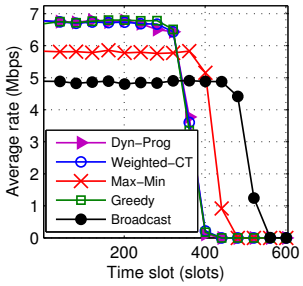


Fig. 6. Instantaneous sum throughput for a homogeneous network.

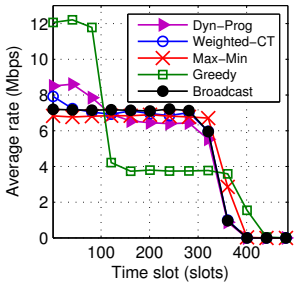


Fig. 7. Instantaneous sum throughput for a heterogeneous network.

In contrast to *Greedy*, with *Dyn-Prog*, *Broadcast*, and *Weighted-CT* the users finish relatively close together, since the algorithms serve both users simultaneously. However, *Dyn-Prog* has a higher sum throughput than *Max-Min*, *Broadcast*, and *Weighted-CT* for time slots ≤ 100. The optimum strategy is to opportunistically favor the *good* user *as long as both users have a small difference in terms of the amount of received data.* This effect is depicted in Fig. 8 where the state space visited by the *Dyn-Prog* algorithm is shown. The darker the square, the more often the corresponding states are visited. The *good* user receives more data than the *bad* user at the beginning of the transmission (i.e., when the *bad* user has 2000kbits, the *good* user usually has $\approx$ 3000kbits, but may have as few as 2000kbits). For time slots ≥ 100 in Fig. 7, *Dyn-Prog* transmits at the rate of the trailing *bad* user to ensure that the *bad* user catches up. Since a scheme that transmits at the rate of the *bad* user sometimes serves only the *bad* user, *Dyn-Prog* yields a lower sum throughput than *Broadcast*, *Max-Min*, and *Weighted-CT* for time slots ≥ 100. Fig. 8 reflects the characteristic of *Dyn-Prog* serving the *bad* user for $x_{bad} \geq 1500$kbits and $x_{good} \geq 2500$kbits where the progress along the x-axis (*bad* user) is larger than that for the *good* user.

Weighted-CT has a lower sum throughput than *Dyn-Prog* (see Fig. 7) for time slots ≤ 100 because the computation of the expected completion time is sub-optimal – it does not take into account all of the (exponential number of) strategies that the optimal *Dyn-Prog* algorithm explores. From Fig. 9, we see that the sub-optimality of *Weighted-CT* causes it to be more conservative at the beginning of the transmission, and in turn achieves a slightly higher rate later on since the states of the *good* and *bad* user are closer together. Nonetheless, *Weighted-CT* performs very close to *Dyn-Prog* compared to the other algorithms for all $\bar{\gamma}^g$ since *Dyn-Prog*'s slightly more aggressive initial behavior only provides a marginal reduction in completion time.

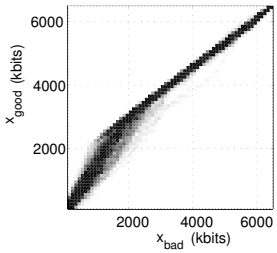


Fig. 8. State space visits for *Dyn-Prog* at $\bar{\gamma}^g = 12.8$dB.

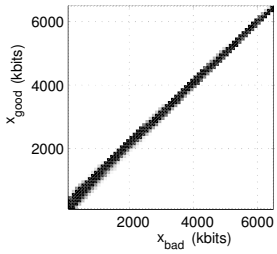


Fig. 9. State space visits for *Weighted-CT* at $\bar{\gamma}^g = 12.8$dB.

6.2 Completion time comparison in Multipath Rayleigh Fading networks

In the following, we use the flat multipath Rayleigh fading model with path loss (ITU-R Pedestrian B [20]) and evaluate the performance in both homogeneous and heterogeneous scenarios for different numbers of users N (see Section 6.2.1) and block sizes B (see Section 6.2.2). From this section onwards, due to the complexity of *Dyn-Prog* mentioned in Section 5, we only evaluate the performance of *Weighted-CT*, *Greedy*, *Max-Min* and *Broadcast*.

In a homogeneous scenario, we place the users equidistant from the BS while in the heterogeneous scenario, the users are randomly distributed within the cell coverage area of radius 250m. The multipath Rayleigh channel distributions of the users are independent and identically distributed (i.i.d.). The transmit power is set such that the edge user is still able to receive data with some probability of success at the lowest MCS. Note that since we use the pedestrian channel model, the rate of change in channel SNR is rather low.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

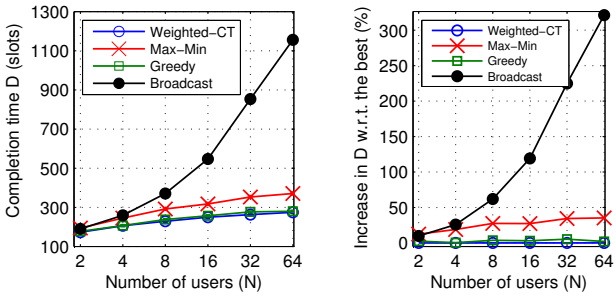


Fig. 10. Impact of increasing N for homogeneous multi-path Rayleigh fading scenario for $B = 6400$kbits.

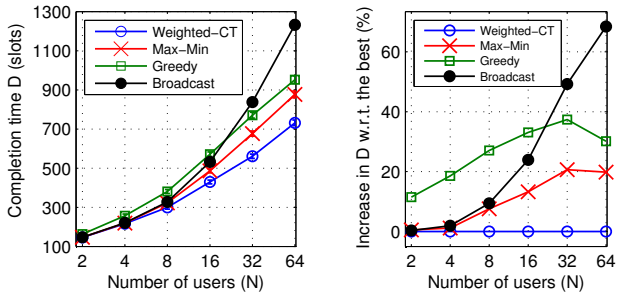


Fig. 11. Impact of increasing N for heterogeneous multi-path Rayleigh fading scenario for $B = 6400$kbits.

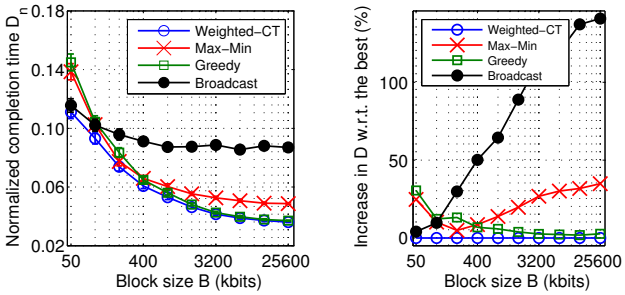


Fig. 12. Impact of increasing B for homogeneous multi-path Rayleigh fading scenario for $N = 16$.

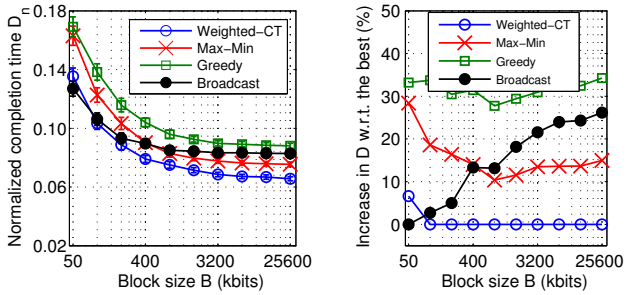


Fig. 13. Impact of increasing B for heterogeneous multi-path Rayleigh fading scenario for $N = 16$.

6.2.1 *Impact of increasing the number of users* N

Here, we observe the impact of increasing N exponentially from 2 to 64 and fixing B at 6400kbits. Fig. 10 and Fig. 11 show this impact for a homogeneous and heterogeneous scenario, respectively. The completion time increases with N because a higher N increases the probability that there is a user with a low SNR channel. To make it easier to compare the relative performance of the schemes, we also include a graph that shows the relative increase of the completion time for each scheme compared to the best scheme, for each scenario.

Homogeneous scenario. The channel variation (i.e., the difference between the best and the worst channel instance) of the Rayleigh fading channel is high and thus we can observe similar relative performance between the schemes in Fig. 10 and Fig. 4 (a homogeneous 2 user scenario in Section 6.1.1) for $\delta \geq 10$dB. As mentioned, the optimal scheme for homogeneous scenario is the one that exploits opportunism and transmits at a higher rate to achieve maximum throughput at each time instant. Therefore *Weighted-CT* and *Greedy* achieve a lower D compared to the other schemes. As N increases, which user is the trailing user changes more often and the trailing user may be a user with a better channel. Since it is better to opportunistically serve users with better channel in a homogeneous scenario, *Max-Min* performs better than *Broadcast*. *Broadcast* performs worst and its completion time increases with N because the probability that there exist a user with a very low SNR in a given slot is higher for higher N.

Heterogeneous scenario. In Fig. 11, the heterogeneity (i.e., the difference of the average channels) between a user and its nearest neighbor is higher when the number of users in the system is small. Therefore, we observe that the relative performance of the schemes in a Rayleigh fading channel (see Fig. 11) for small N is similar to the performance of the heterogeneous 2 user scenario (see Fig. 5 in Section 6.1.2) for higher $\bar{\gamma}^g$. According to the explanation in Section 6.1.2, *Broadcast* performs close to optimal and *Greedy* that optimizes for opportunistic gain performs worst when there is one clearly worst user. As N increases, the user's density increases and thus the users that are close to each other have a very small difference in terms of the average channels. This resembles a homogeneous scenario. In such a scenario, transmitting at the broadcast rate and conservatively serving all users results in higher D than using opportunistic gain among the users that are near each other. Therefore *Greedy* performs better than *Broadcast* for higher N. For high N, the multicast rate is highly affected by the users located at the edge of a cell (i.e., edge users). Since the trailing user is normally an edge user, *Max-Min* may by chance serve the correct users. Therefore, it outperforms *Greedy* for serving the important users and *Broadcast* since it is less conservative. As before, *Weighted-CT* outperforms all other schemes since it exploits the users' weight and optimizes the rates at which the edge users are served.

6.2.2 *Impact of increasing the block size* B

Here, we fix N to 16 and examine the impact of increasing B exponentially from 50kbits to 25600kbits. For ease of comparison, the results are presented in terms of normalized completion time, $D_n = D/B$. Fig. 12 and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

Fig. 13 illustrate the impact of increasing B on D_n. When B is small, a scheme requires very few slots to receive a block and thus the number of the channel samples is small. Due to this, the average SNR of the channel samples and the average SNR of the channel distribution may differ significantly. Note that D is determined by the worst users in the system. If the average SNR of the channel samples of some of the users is lower than the average SNR of the channel distribution, this causes a higher D. This impact reduces for larger B. According to the law of large numbers, for larger B, the average SNR obtained from the larger number of channel samples is closer to that of the distribution. The larger D allows the system to stay in steady state (where all users are still far from finishing) for a longer period of time. As a consequence, D_n becomes flatter (i.e., the increase in D is approximately linear with B).

Homogeneous scenario. When B is small (e.g., $B = 50$kbits), the users with a better instantaneous channel are more likely to finish very early (within very few slots) and the users with a worse instantaneous channel remain in the system. In such a scenario, an algorithm performs best if it serves all the users at the rate of those with the worse channel instances. Therefore *Broadcast* performs well. *Weighted-CT* performs best since it is more conservative towards the users with a worse instantaneous channel. In contrast, *Greedy* performs worst because it at first serves users with a better instantaneous channel and only serves the users with worse instantaneous channel later. *Max-Min* selects a user among the trailing users randomly when more than one user has the lowest amount of received data. When this randomly selected trailing user is not the user with the worst instantaneous channel, it yields a higher instantaneous throughput but this causes longer D in the future. For larger B, the performance difference is as explained in Section 6.2.1 for the homogeneous scenario when $N = 16$.

Heterogeneous scenario. In a heterogeneous scenario, the reason for the performance difference between *Greedy*, *Max-Min*, and *Broadcast* in Fig. 13 for small B is similar to that explained for Fig. 12. *Greedy* performs worst regardless of B since it always favor the better users (refer to Section 6.1.2 for a detailed explanation). Since serving the worse users is important in this scenario, *Max-Min* always performs better than *Greedy* because the trailing user is one of the worse users. *Weighted-CT* performs slightly worse than *Broadcast* for $B = 50$kbits because for a very short completion time, there is a discrepancy between the estimated rate (computed using Eq. 8) of the channel samples and that of the channel distribution. This incorrect rate estimation causes *Weighted-CT* to make the wrong decisions (i.e., choosing a wrong MCS). For larger B, higher number of channel samples allows *Weighted-CT* to estimate the users' rate and completion time more accurately and thus it performs better than the other schemes.

6.3 Evaluation of the Energy Consumption

This section presents the energy consumption of the experimented schemes in the homogeneous and heterogeneous multipath Rayleigh fading scenarios with perfect feedback.

At slot k, a user is either in the *on* state or in the *idle* state. A user is in the *on* state if it is scheduled for reception and *idle* state otherwise. The parameters for computing the energy consumption are listed in Table 2[3].

TABLE 2
LTE power model parameters

Symbol	Description	Value
ξ_{on}	*on*'s state base power	1210.7 ± 85.6 (mW)
ξ_{idle}	*idle*'s state base power	594.3 ± 8.7 (mW)
θ_{dl}	Power per Mbps	51.97 (mW/Mbps)

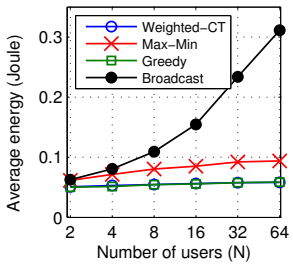


Fig. 14. Average energy consumed in homogeneous multipath Rayleigh fading scenario for $B = 6400$kbits.

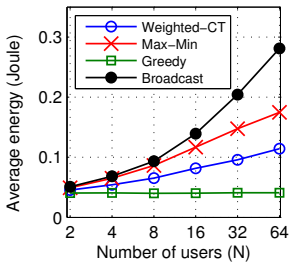


Fig. 15. Average energy consumed in heterogeneous multipath Rayleigh fading scenario for $B = 6400$kbits.

As expected, the energy consumption is proportional to the completion time. This can be seen in the homogeneous scenario where a scheme with the lowest completion time (*Weighted-CT*) in Fig. 10 (see Section 6.2.1) has the lowest energy consumption in Fig. 14.

Although *Weighted-CT* has lower completion time than *Greedy* in Fig. 11, its average energy consumption is higher (see Fig. 15). The energy consumption is not only affected by the completion time but it is also affected by the number of slots a user stays in the *on* state. Table 2 shows that a user in the *on* state consumes more than twice the amount of base energy than it does in the *idle* state. In a heterogeneous scenario, *Weighted-CT* consumes higher average energy than *Greedy* because it tries to serve more users and transmits at a lower rate than *Greedy*. Transmitting at lower rate causes the better users to stay in the *on* state for more slots. In contrast, *Greedy* first transmits at a high rate and reduces the number of slots the better users stay in the *on* state. As a result, its average energy consumption is lower. Increasing N does not decrease the rates at which users are served hence the average energy consumption of

3. The energy consumption in LTE system is presented in [22] and [23]. We ignore the energy consumed by the wake up operation since it is less than 5% of the energy consume in an *idle* state [22].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

Greedy remains the same as N increases. In contrast, for *Weighted-CT*, a higher N causes more users to stay in the system for a longer time and thus its average energy consumption increases with N in Fig. 15.

In summary, the average energy consumption depends on two main factors: the completion time and the percentage of the slots the users stay in the system (especially in the *on* state).

6.4 Impact of imperfect and limited state information

The schemes presented in this paper need either the channel state or the receiver state information or both to select an MCS. In wireless networks, frequent feedback of the state information substantially increases system overhead and thus reduces system throughput. In this section, we examine the impact of imperfect feedback information on all schemes. We also include *Weighted-CTe* that estimates the probability distribution of the current channel state based on the outdated past feedback (see Section 5.3). First, we analyze the impact of increasing the feedback interval (i.e., reducing the frequency of feedback) of the state information. We then study the effect of limiting the number of users that periodically feed back their state information to the BS.

6.4.1 Impact of the feedback interval

As mentioned above, transmitting feedback at each slot is costly. Here, we model a feedback system where users only send feedback every λ slots. We then analyze the impact of the feedback interval λ on the completion time D. The feedback slot (the slot at which a user reports) is asynchronous, but λ is the same for all users.

We choose $\lambda \in \{5\text{ms}, 10\text{ms}, 20\text{ms}, 40\text{ms}, 80\text{ms}, 160\text{ms}\}$ (these are the common periodic feedback intervals in an LTE system [24]), such that we can observe the impact of λ when it is less than or greater than $t_c = 40ms$. For $\lambda \leq t_c$, the current channel state and the one reported in the last feedback (i.e., the last available channel state) are correlated, otherwise, they are uncorrelated. When feedback is unavailable, the BS updates the receiver state assuming that the amount of data received in the current slot is equivalent to that reported in the last available feedback packet. When feedback is available, the BS updates the receiver state to the current receiver state.

Fig. 16 and Fig. 17 depict the impact of different feedback intervals λ on D in the homogeneous and the heterogeneous scenarios, respectively. D increases with λ because the correlation between the current channel state and the last available channel state decreases and thus the uncertainty about the channel state increases. As mentioned, the receiver state is updated regardless of the availability of feedback information. The difference between the current and the estimated receiver state is usually small and thus has a minimal impact on the chosen MCS. Hence, delayed channel state has a much higher impact than the receiver state on the completion time.

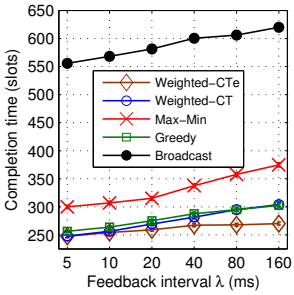


Fig. 16. Impact of increasing λ for a homogeneous multipath Rayleigh fading scenario, $B = 6400$kbits, $N = 16$.

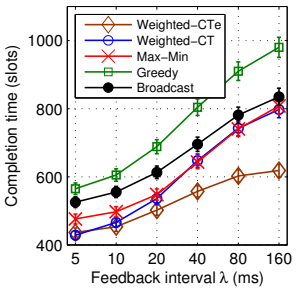


Fig. 17. Impact of increasing λ for a heterogeneous multipath Rayleigh fading scenario, $B = 6400$kbits, $N = 16$.

Fig. 16 and Fig. 17 show that D continuously increases with λ. For $\lambda > t_c$, the last available channel state is only useful up to time t_c. From time t_c until the next feedback is received, the current and the last available channel state are uncorrelated. Therefore, increasing λ beyond t_c increases the fraction of time where the current and the last available channel state are uncorrelated. When comparing the impact of λ on D in the homogeneous (see Fig. 16) and the heterogeneous (see Fig. 17) scenarios, the impact is greater in the latter. In a homogeneous scenario (with a sufficiently large N), the difference between the chosen MCS based on the last available channel state and the one that would be chosen based on the current channel state is usually small. In contrast, in a heterogeneous scenario, an outdated channel state may cause an algorithm to select a very different MCS compared to the MCS that would be selected based on the current channel state since the MCS largely depends on the channel state of a small subset of users. Choosing either a too low or too high MCS increases D. As a result, the impact of delayed state information is higher in the heterogeneous scenario than in the homogeneous scenario. For $\lambda > t_c$, although *Weighted-CT* may make the wrong MCS choice due to outdated last feedback, it performs as good as or better than the other schemes except for *Weighted-CTe*.

Weighted-CTe outperforms the other schemes in all scenarios, especially for higher λ. The MCS that is chosen for an outdated channel is the one that is optimal for the distribution of the estimated channel (since little is know about the channel). As a result, *Weighted-CTe* that picks the MCS using channel estimation significantly outperforms all the other schemes especially for high λ. With channel estimation, the completion time of *Weighted-CTe* improves over *Weighted-CT* by 17.5% and 30% in homogeneous and heterogeneous scenarios, respectively. As mentioned, the difference of the selected MCS between the current channel state and the last available channel state is larger in the heterogeneous scenario therefore the impact of channel estimation is more evident in this scenario.

We also investigate the performance of *Broadcast*, *Max-*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

Min, and *Greedy* when channel estimation similar to *Weighted-CTe* is applied. Due to space limitation, we exclude the graphs. With channel estimation, *Max-Min* and *Greedy* achieve improvements of up to 29% and 21%, respectively in terms of D compared to *Max-Min* and *Greedy* without channel estimation. In contrast, *Broadcast* with channel estimation performs worse than all the other schemes, including the *Broadcast* scheme without channel estimation. This is because the estimated channel of *Broadcast* is the worst channel in the channel distribution.

6.4.2 Impact of interval feedback from fewer users

In this sub-section, the feedback load is further decreased by reducing the number of reporting users (N_{rep}). We examine the minimum N_{rep} for each scheme that achieves the same D as that when all users give feedback. We first elaborate the method on the selection of the reporting users. We then present the minimum N_{rep} for $N = \{2, 4, 8, 16, 32\}$ for a feedback interval $\lambda = 20$ms.

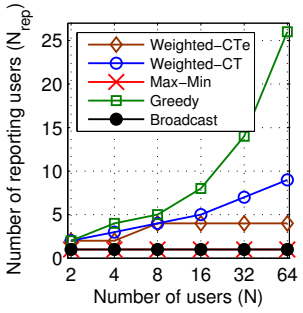


Fig. 18. N_{rep} in a homogeneous multipath Rayleigh fading scenario, $\lambda = 20$ms.

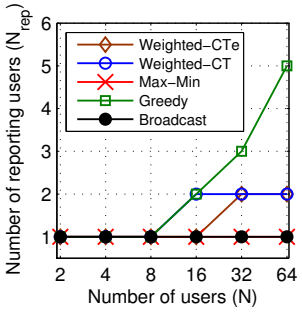


Fig. 19. N_{rep} in a heterogeneous multipath Rayleigh fading scenario, $\lambda = 20$ms.

Regardless of N, *Broadcast* decides its transmit rate based on the worst instantaneous channel. *Max-Min* decides based on the channel of the trailing user. Therefore, the minimum N_{rep} for *Broadcast* and *Max-Min* is one. Since *Weighted-CT* and *Weighted-CTe* favor the users with lower receiver state (these users require longer time to receive B), these users are selected as the reporting users. *Greedy* maximizes throughput and it highly depends on the complete instantaneous channel statistic to achieve an opportunistic gain, thus it performs better for higher N_{rep}. For *Greedy*, the reporting users are selected randomly.

In Fig. 18 and Fig. 19, we show the minimum N_{rep} required by each scheme for different N. Note that we do not show the completion time here. The completion time is similar to that depicted in Fig. 10 (for a homogeneous scenario) and Fig. 11 (for a heterogeneous scenario) since for $\lambda = 20$ms the current channel state and the last available channel state are still correlated. In Fig. 10, *Greedy* performs close to *Weighted-CT* but the required N_{rep} is more than twice as large as that of *Weighted-CT* (see Fig. 18), and consequently has higher overhead than *Weighted-CT*.

As depicted in Fig. 18 and Fig. 19, N_{rep} increases with N for *Weighted-CTe*, *Weighted-CT*, and *Greedy* because more users are needed to reflect the channel statistics of the important users for each scheme. For *Greedy* to realize the opportunistic gain, the feedback from one user is as important as any other users in the system and thus a higher N_{rep} is required.

In Fig. 18, it is interesting to note that the minimum N_{rep} of *Weighted-CTe* is lower than that of *Weighted-CT* in the homogeneous scenario. As explained in the previous section (Section 6.4.1), a high λ reduces the number of MCS choices when channel estimation is used. Therefore, only a small N_{rep} is needed for *Weighted-CTe*. On the other hand, *Weighted-CT* has a wider range of MCS choices thus including more users results in smaller D. As discussed in Section 6.4.1, in the heterogeneous scenario, it is important to select the edge users since the completion time is determined by those users. The number of the edge users increases with N and thus N_{rep} is larger for larger N. *Weighted-CT* and *Weighted-CTe* have the same N_{rep} in Fig. 19 except for $N = 16$. This small difference is caused by the number of MCS choices as explained above.

To recap, the minimum number of reporting users N_{rep} depends on the network scenario. Since users are usually randomly distributed in actual mobile networks, they are heterogeneous. As shown in our result (see Fig. 19), the BS (for any algorithm it uses) can achieve its best performance with low overhead because it only needs feedback from a small fraction of users in the system to achieve the minimum completion time for each scheme.

7 Conclusions

In this paper, we investigated the finite horizon opportunistic multicast scheduling problem, where a wireless base station transmits a fixed amount of erasure coded data to a set of receivers. We designed an algorithm based on dynamic programming that to the best of our knowledge, is the first to explicitly take into account the system state in terms of the received amount of data at each receiver for the selection of the optimum modulation and coding scheme. In addition to the well known tradeoff between broadcast gain and multi-user diversity gain that is inherent to opportunistic multicast scheduling, the finite horizon nature of our problem introduces an interesting further tradeoff, namely that of equalizing the completion times of the users versus the total system throughput. This tradeoff is state dependent. Intuitively, throughput maximization is a reasonable strategy as long as users are far from finishing, whereas the closer users are to finishing, the more important it becomes to allow lagging users to catch up rather than optimizing throughput for all. Based on these insights, we designed two simple and practical heuristics that perform close to the more complex dynamic programming solution and that outperform existing approaches that do not consider the receiver state.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2015.2418766, IEEE Transactions on Mobile Computing

We performed an extensive range of simulations for homogeneous as well as heterogeneous user scenarios and show that our heuristics outperform the existing *Greedy* and *Broadcast* schemes by as much as 30% and 120%, respectively, in Rayleigh fading scenarios. Since reducing the system overhead improves the system throughput. We also present results on the impact of increasing the feedback interval and reducing the number of users that give feedback on the proposed schemes. In particular, we extend our heuristic to estimate the current channel state based on outdated last feedback. This extension further improves performance by 17.5% and 30% in homogeneous and heterogeneous scenarios, respectively.

REFERENCES

[1] P. K. Gopala and H. E. Gamal, "Opportunistic multicasting," in *IEEE ASILOMAR*, Nov 2004.

[2] T.-P. Low, M.-O. Pun, Y.-W. P. Hong, and C.-C. J. Kuo, "Optimized opportunistic multicast scheduling (OMS) over heterogeneous cellular networks," in *IEEE ICASSP*, Apr. 2009.

[3] A. Richard, A. Dadlani, and K. Kim, "Multicast scheduling and resource allocation algorithms for OFDMA-based systems: A survey," *IEEE Commun. Surveys Tuts*, pp. 1–15, 2012.

[4] U. Kozat, "On the throughput capacity of opportunistic multicasting with erasure codes," in *IEEE INFOCOM*, Apr. 2008.

[5] A. Shokrollahi, "An Introduction to Low-Density Parity-Check Codes," ser. Lecture Notes in Computer Science. Springer, 2002.

[6] ——, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[7] P. K. Gopala and H. E. Gamal, "On the throughput-delay tradeoff in cellular multicast," in *IEEE IWCMC*, Jun. 2005.

[8] T.-P. Low, M.-O. Pun, and C.-C. J. Kuo, "Optimized opportunistic multicast scheduling over cellular networks," in *IEEE Globecom*, Dec. 2008.

[9] W. Huang and K. L. Yeung, "On maximizing the throughput of opportunistic multicast in wireless cellular networks with erasure codes," in *IEEE ICC*, Jun. 2011.

[10] E. Veshi, A. Kuehne, and A. Klein, "Comparison of different multicast strategies in wireless identically distributed channels," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013, pp. 3010–3015.

[11] D. L. Gall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM - Special issue on digital multimedia systems*, vol. 34, no. 4, pp. 46–58, Apr. 1991.

[12] W. Ge, J. Zhang, and S. Shen, "A cross-layer design approach to multicast in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 3, pp. 1063–1071, Mar. 2007.

[13] T.-P. Low, M.-O. Pun, Y.-W. P. Hong, and C.-C. J. Kuo, "Optimized opportunistic multicast scheduling (oms) over wireless cellular networks," *IEEE Transaction on Wireless Communications*, vol. 9, no. 2, pp. 791 –801, Feb. 2010.

[14] Y. Wang, X. Wang, M. Li, and J. Qu, "A parity-based opportunistic multicast scheduling scheme over cellular networks," in *IEEE DASC*, Dec. 2013, pp. 565–568.

[15] Q.-D. Ho and T. Le-Ngoc, "Adaptive opportunistic multicast scheduling over next-generation wireless networks," *Wirel. Pers. Commun.*, vol. 63, no. 2, pp. 483–500, Mar. 2012. [Online]. Available: http://dx.doi.org/10.1007/s11277-010-0145-y

[16] W. Huang and K. L. Yeung, "Optimal opportunistic multicast for minimizing broadcast latency in wireless networks," in *IEEE ICC*, May 2010.

[17] S.-M. Huang, J.-N. Hwang, and Y.-C. Chen, "Reducing feedback load of opportunistic multicast scheduling over wireless systems," *IEEE Communications Letters*, vol. 14, no. 12, pp. 1179–1181, Dec. 2010.

[18] C. Mehlführer, M. Wrulich, J. C. Ikuno, D. Bosanska, and M. Rupp, "Simulating the long term evolution physical layer," in *IEEE EUSIPCO*, vol. 27, 2009, p. 124.

[19] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. I*, 3rd ed. Athena Scientific, 2005.

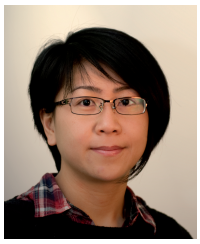
[20] I. Recommendation, "1225,guidelines for evaluation of radio transmission technologies for imt-2000,," *International Telecommunication Union*, 1997.

[21] R. del Negro. (2014) Bitrate & gop calculator. [Online]. Available: http://dvd-hq.info/

[22] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *ACM MobiSys*, 2012, pp. 225–238.

[23] A. Asadi and V. Mancuso, "On the compound impact of opportunistic scheduling and d2d communications in cellular networks," in *ACM MSWiM*, 2013, pp. 279–288.

[24] S. Sesia, I. Toufik, and M. Baker, *LTE: the UMTS long term evolution*. Wiley Online Library, 2009.



Gek Hong Sim is currently a Ph.D. student in University Carlos III and she is also affiliated with IMDEA networks Institute, Madrid, Spain. Gek Hong received her Bachelor Degree (Engineering majoring in Telecommunication) and first Master Degree (Engineering Science) from Multimedia University, Malaysia in 2007 and 2011, respectively. She was awarded with Master Degree in Telematics Engineering from University Carlos III Madrid in 2012. She was working on multicast networks. Her main research interest are multicast scheduling in LTE and MAC layer optimization for 60GHz communication.

Dr. Joerg Widmer (M'06 – SM'10) is a Research Professor at IMDEA Networks Institute in Madrid, Spain. He received his M.S. and PhD degrees in computer science from the University of Mannheim, Germany in 2000 and 2003, respectively. His research focuses primarily on wireless networks, ranging from MAC layer design and interference management to mobile network architectures. From 2005 to 2010, he was manager of the Ubiquitous Networking Research Group at DOCOMO Euro-Labs in Munich, Germany, leading several projects in the area of mobile and cellular networks. Before, he worked as post-doctoral researcher at EPFL, Switzerland on ultra-wide band communication and network coding. He was a visiting researcher at the International Computer Science Institute in Berkeley, CA, USA and University College London, UK. Joerg Widmer authored more than 100 conference and journal papers and three IETF RFCs, holds several patents, serves on the editorial board of IEEE Transactions on Communications, and regularly participates in program committees of several major conferences. Recently, he was awarded an ERC consolidator grant as well as a Spanish Ramon y Cajal grant. He is senior member of IEEE and ACM.

Dr. Balaji Rengarajan (M'09) is currently an algorithms architect with Accelera Mobile Broadband, CA, USA. Previously, he was a staff researcher at IMDEA Networks, Madrid, Spain. He received his Ph.D. and M.S. in electrical engineering from the University of Texas at Austin, USA in 2009 and 2004 respectively, and his B.E. in Electronics and Communication from the University of Madras in 2002. He was the recipient of a 2003 Texas Telecommunications Engineering Consortium (TxTEC) graduate fellowship and a 2010 Marie-Curie Amarout Europe Programme fellowship. He is also the recipient of the best paper award at the 23rd International Teletraffic Congress (ITC), 2011. His main research interests lie in the analysis and design of wireless and wireline telecommunication networks.