Cohen *et al. EURASIP Journal on Advances in Signal Processing* 2013, **2013**:25
http://asp.eurasipjournals.com/content/2013/1/25

EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Coded unicast downstream traffic in a wireless network: analysis and WiFi implementation

Asaf Cohen*, Erez Biton, Joseph Kampeas and Omer Gurewitz

Abstract

In this article, we design, analyze and implement a network coding based scheme for the problem of transmitting multiple unicast streams from a single access point to multiple receivers. In particular, we consider the scenario in which an access point has access to infinite streams of data to be distributed to their intended receivers. After each time slot, the access point receives acknowledgments on previous transmissions. Based on the acknowledgements, it decides on the structure of a coded or uncoded packet to be broadcast to all receivers in the next slot. The goal of the access point is to maximize the cumulative throughput or discounted cumulative throughput in the system. We first rigorously model the relevant coding problem and the information available to the access point and the receivers. We then formulate the problem using a Markov decision process with an infinite horizon, analyze the value function under the uncoded and coded policies and, despite the exponential number of states, devise greedy and semi-greedy policies with a running time which is polynomial with high probability. We then analyze the two users case in more detail and show the optimality of the semi-greedy policy in that case. Finally, we describe a simple implementation of the suggested concepts *within* a WiFi open-source driver. The implementation performs the network coding such that the enhanced WiFi architecture is transparent above the MAC layer.

1 Introduction

The inherent broadcast nature of the wireless medium, which allows each transmission to be heard by all users simultaneously, makes network coding techniques pertinent. In such techniques, nodes do not necessarily forward incoming packets. Rather, they can transmit a manipulation (usually a linear combination [1]) of their incoming data. However, in order for such a combination to be valuable to multiple users, each such user needs to possess different piece of the information encoded into the combined packet. Accordingly, one of the key challenges in network coding techniques is to decide which packets to manipulate in each transmission. While efficient algorithms answer this challenge in the multicast setting [2], the problem of multiple unicast remains open [3].

On the down side, the wireless medium characteristics make wireless transmissions susceptible to losses due to noise and interference (i.e., low SNR and SINR). In order to cope with packet loss in MAC layer, conventional wireless protocols rely on retransmissions (e.g., WiFi, [4]). In such protocols, each packet has to be acknowledged by the intended receiver. Packets which are not acknowledged are retransmitted over and over again until they are received successfully by the receiver, or until dropped by the sender.

Typical last mile wireless Internet access architecture comprises a gateway, e.g., an access point (AP) or a Base Station (BS), to which all clients are wirelessly connected (e.g., WiFi, WiMAX, LTE). In such architecture, all traffic to and from the wired Internet must pass through the gateway via the wireless medium. Accordingly, all transmissions by the gateway are potentially heard by all clients associated with this gateway. In this article, we utilize these aforementioned wireless properties of channel, protocol and last mile architecture and suggest coded wireless retransmissions for downstream traffic. In particular, we suggest a novel scheme which is based on Markov decision process (MDP [5-7]), that combines multiple MAC layer retransmissions which are intended to different receivers, into a single packet transmission.

*Correspondence: coasaf@bgu.ac.il
Department of Communication Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, 84105, Israel

Springer

© 2013 Cohen et al.; licensee Springer. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1.1 Main contributions

Our contributions are thus as follows. First, we suggest an ongoing process in which the AP (or gateway) alternates between transmitting uncoded and coded packets. Receivers acknowledge packets they have received successfully and in addition provide feedbacks to the AP regarding packets they overheard which were not meant for them. Based on these feedbacks the AP chooses which retransmitted packets (if any) should be coded in each retransmission. Our model is inherently not multicast—*each receiver has a different stream as its demand*. Moreover, we assume an *infinite horizon* model, where there is no point in time in which all demands are met and the system reaches a terminating step. Packets arrive at the AP continuously, and only the current packets for each user are available for coding. Based on this model, we are able to analytically solve throughput problems. We believe that these two aspects of our model are of key importance, since this is the typical use of most wireless Internet access networks.

Second, we show that the aforementioned continuous transmission process can be modeled as a discrete time stochastic process, in which at each state the next state is determined solely based on the AP decision which packet to transmit next (i.e., which coded or un-coded packets should comprise the next transmission) and based on the channel state of each and every receiver which determines which nodes receive the next transmission. We suggest an AP *policy* which is based on MDP theory, in which the reward attained in each iteration corresponds to the number of successful packets received in each transmission.

Third, we leverage this continuous, infinite time stochastic model, to compute stationary behavior, which in turn allows us to define convergence, calculate the resulting asymptotic performance efficiently (using only a set of linear equations), and assess the benefit in coding directly and analytically. Specifically, we give the matrix equation that computes the cumulative expected reward (equivalent to the system throughput when a unit reward is given to decoding of one packet) for any state in the system given the transition probabilities and the reward vector. This enables us to directly compute the performance of *any* coded or un-coded strategy. For the two user case, we indeed give a few possible strategies and compute the resulting performance.

Fourth, we show that in order to reach an optimal decision, the AP needs to consider all possible future states of the system, channel states of all users and all possible actions and outcomes. This procedure certainly cannot scale to large number of users. Accordingly, we suggest a greedy approach in which at each transmission the AP tries to maximize the instantaneous reward received for each transmission (as opposed to maximizing an expected or discounted reward, which takes into account the expected rewards at future states). We further suggest an enhancement to the greedy approach, termed semi-greedy approach, which takes some concern into the future, without adding significant complexity to the greedy approach. In the semi-greedy approach, we also suggest a direct analysis for the simple case of two receivers, which besides the analysis of this simple case, also provides some insight into much larger scenarios. We evaluate both schemes via an extensive set of simulations, which show that our approach attains high gain over the traditional un-coded transmissions while maintaining long time fairness. Moreover, we show that the semi-greedy approach exploits the multi-user diversity in the system, putting more emphasis on serving the users with the best channels conditions at any given time.

Finally, we implement our scheme on a WLAN topology in which a single AP transmits unicast traffic to two receivers. We show that the suggested scheme can be easily implemented over a typical 802.11 card, with some modifications to the wireless driver. To the best of our knowledge, this is the first implementation of these concepts *within the WiFi driver*, and transparently from the upper layers. We further show that at least for this simple case, the experimental gain agrees with the one predicted by our analytical model.

1.2 Related study

At the basis of our study stands the already well understood concepts of *network coding* [1]. In this pioneering study, intermediate nodes in the network perform coding operation on the data in order to achieve certain rate goals. Indeed, it was shown that network coding can improve the network throughput significantly, and achieve the optimal performance in the multicast scenario. The theory of network coding includes linear [8,9] as well as non-linear coding techniques. In this article, we focus on coherent linear network coding.

Following [1], several important studies discussed various practical issues in network coding. [10] introduced the idea of *generations*, and suggested coding only over packets of the same generation. As generations advance, old generations are flushed. In a sense, the concept is useful in this study as well, when we suggest that if a user acknowledges receiving a packet intended only to him, neighboring users who overheard it in a previous transmission and buffered it, discard it. Practical aspects of network coding also include several key works on opportunistic coding. That is, protocols, algorithms and analysis aimed at understanding which packets to send coded, and which coding coefficient to use, given the senders (maybe limited) knowledge on the data available at the receivers. Coding using only local information and opportunistic

network coding was first introduced in [11,12] as COPE. While decentralized, this groundbreaking work was not tailored to the multiple unicast with one sender scenario we consider in this article. A polynomial time centralized algorithm, yet with guarantees only for the multicast scenario, was given in [2].

In [13], loss aware coding in wireless multi-hop networks was considered. Therein, having knowledge of the packets available at each users, the sender is trying to come up with a *sequence of transmissions* to satisfy as many users as possible. Thus, the model given in [13] is not the Markovian model we consider in this study, and is of *finite horizon* nature, i.e., where the set of packets to be transmitted is finite and known in advance to the sender, compared to the infinite sequences of data per user we consider here. Note, however, that the assumption that the sender is omniscient, knowing which packets received where, is similar to our setting, and many related studies [14-17].

Also related is the recent study by Sorour and Valaee [18]. Therein, a fixed set of packets is transmitted to all users. Then, after acknowledgments are received, the system computes which coded packets to send in order to satisfy the demands. A sequence of coded retransmissions is constructed, sent, and only when all users reconstruct all their intended packets the system continues to the next set of packets. Again, this is fundamentally different from the *infinite horizon* model we consider herein. In a sense, the model described in our study allows for coding across MBS frames (using the notation of [18]), whereas [18] allows coding only within an MBS frame. Moreover, the scheme in [18] is adapted to the case where *all loss probabilities are equal*. The successive studies in [19-21] also consider the finite horizon model, yet contribute significantly to our knowledge in terms of minimizing delay [19,20] or maximizing coding opportunities [21].

Studies on opportunistic coding for the finite horizon broadcast case, where users are interested in *all packets*, and the sender has a *fixed set of packets to send* also appeared in [15,16]. Nevertheless, a key difference compared to the model we define herein is in the *multicast demand structure*—eventually all users demand all the information available. While this demand structure is well-understood [1,8,9], the capacity region in the general multi-source multi-sink setting [22], as well as the multiple unicast setting [3], remains unsolved. In the context of the setting discussed in this article, the index coding problem [17], which also considers only a single sender with multiple receivers (having side information), is also unsolved in general. Note that index coding is, in general, over noiseless channels, and does not include the dynamic, error-prone and infinite time setting we consider herein.

In [23], the authors considered a similar star topology, with one server supporting several clients (receivers). The demand structure in [23] is more general compared to the previously discussed studies [14-16], that is, not necessarily all users require all information. However, the model in [23] is different than the one suggested in this article. First, it is a finite horizon model, where *all data is available at the server* (in advance) for coding. In the model discussed herein, only the packets intended for current transmission (or retransmission) are available at the server for coding. In many streaming applications, this is usually the case, and the server cannot code "future" packets as these, usually, are not available at the time of transmission. The model in [23] also assumes user can buffer all packets, even those who cannot be decoded immediately, requiring working over a large finite alphabet. Moreover, in this case, optimally deciding which coded packet to send is prohibitively complex. For this reason, coding in [23] is done over "classes" of users, and these classes are pre-defined and fixed for the entire transmission. Random linear network coding was used, where coding is only across classes of files. This converted the problem to multiple-multicast sessions (that is, a user is required to decode *all files* within its class in order to retrieve its own file). Finally, the main figure of merit therein was the *delay*. Indeed, a rule of thumb that arose from this work was to avoid coding across files (users, in our model), and code mainly over packets within the same file. The results under our model will be significantly different, suggesting it is strictly better to code across users, even though it is a multiple unicast scenario, as long as the subsets of users sought for in the opportunistic coding process are not too large, hence do not create a significant computational burden.

In [14], the authors considered in more detail the specific case of coded retransmissions. That is, given the knowledge of which packets *were not received by their intended users, but overheard by others*, the authors suggest a MDP approach to identify good coding strategies. However, similar to [15,16], all receivers are interested in all packets. In the context of the underlying MDP, note that since [14] assumes a fixed, finite set of packets to be sent to all users, there is a *terminating state* for the chain, from which the optimal policy can be calculated using backward recursion. However, stating the analytical solution explicitly is not an easy task.

An information theoretic analysis of the intersession network coding model (multiple unicast), with users' ability to overhear packets intended to other users modeled as an erasure channel, was given in [24]. The system model used therein is the canonical 1-Hop relay model, where the intersessions are performed *through a relay*. Upper and lower bounds on the *capacity* of such a system were discussed. We also mention that there is an inspiring body

of work on network error correction, e.g., [25-27], and noncoherent network coding [28]. Yet, the majority of these studies focus on general network topology on the one hand and specific multicast demand structure on the other.

On the practical side, several studies implemented network coding concepts on real networks. We focus here only on works whose implementation is *below* the application layer (i.e., excluding network-coded content distribution and related studies). A pioneering work in this context is the already discussed [12]. The COPE header in this work resides between the routing and the MAC headers. The implementation runs on a 802.11a network as a *user space daemon*, that is, it sends and receives raw 802.11 frames. Random linear network coding on the iPhone was studied in [29], though, again, the implementation is on top of the WiFi driver, and not within it. Using Nokia mobile devices was suggested in [30]. In [31], the authors considered a chain topology, and gave numerical evaluation of the suggested iCORE scheme. Still, iCORE is a user space daeamon, which *uses* WiFi, but does not alter it. In this article, the implementation was *within* the WiFi driver, rendering the coding procedure transparent to all above layers.

2 System model

We first define the system model we use. We consider a downlink wireless model, with one serving access-point/base-station (sender) and K users/stations (receivers). At the sender, we assume an infinite stream of packets *for each user*. That is, the stream of packets $\left\{P_i^k\right\}_{i=1}^{\infty}$ is intended to the kth user. Thus, our demand structure is that of multiple unicast sessions, with one common sender. At the receiver, we assume packets transmitted to other users are cached (if heard), even though those packets are not intended to itself.

We consider a synchronized system, where the time is divided to discrete slots. At each slot, the sender can send one packet. Our channel model is as follows. At each slot, the packet sent is received at receiver k with probability p_k, independently of the other receivers and of the packets received previously (memoryless independent users). However, when a receiver correctly receives a packet, it broadcasts an acknowledgment, together with its user index i. We assume this acknowledgement is *correctly received by both the sender and all other receivers*. We comment on the possibilities to relax this assumption in Section 4.5.

At each time instant, the sender chooses a packet to send, according to its assessment of the *state* the system is in. However, since the state definition and evolution is intertwined with the coding strategy used, we first describe the possible coding strategies.

2.1 Network coding strategies

Each time a packet is sent, the sender can either choose a packet to send to a single user, or code together a few packets. We assume the standard coherent linear network coding model, e.g., [8,32]. The stream intended to each user can be represented as an infinite stream of bits. This stream is split into packets, each represented as m symbols in a finite field $\mathcal{F}_q$, for a total of $m\lfloor \log q \rfloor$ bits per packet.

In an uncoded packet sent by the access point, simply a packet intended to a single user is sent. In a coded packet, the access point sends a linear combination over $\mathcal{F}_q$ of packets. In our model, the access point *does not code over packets intended to the same receiver*, only across packets intended to different receivers. That is, a sent packet has the form $X = \sum_{j=1}^{K} a_j X_j$, where X_j is the packet *currently intended to user* j and $\{a_j\}_{j=1}^{K} \in \mathcal{F}_q^K$ are the coding coefficients. Note that an uncoded packet can be treated as a coded one, with all coefficients but one equal zero.

Since the access point keeps track of the current packet requested by a receiver, packets within a coded packet can be labeled using the receiver ID alone. To keep track of the linear combination of the uncoded packets contained in a coded one, a coefficient for each uncoded packet (i.e., for each receiver) is sent in the header of each coded packet. Similarly to prior work on network coding, we assume that the packet payload is sufficiently large compared to the header. This renders the overhead of the header negligible.

Clearly, the ability of a user to decode its intended packet depends on its available information. In the general network coding setting, a receiver keeps track of the coded packets it received. The coefficient vectors of these coded packets are stored in a matrix. Once it is full rank, the data can be decoded. In this study, however, we consider a different setting, where a user only keeps track of uncoded packets it received, or packets it decoded *at that time instant*, and disregards coded packets from which it cannot *instantaneously* decode original packets. For this reason, it is possible to limit the field $\mathcal{F}_q$ to be binary, and avoid the complexity burden of larger fields. This can be compared to network coding models where the field size must scale with the number of users (e.g., linear [2]). Note that the extension to a model where a user keeps track of coded packets and the associated vectors of coefficients, even if these cannot lead to decoding at the same time slot, is conceptually simple, but practically does not scale to a large amount of multicast sessions, as the dimension of the *state space* will be prohibitively large.

2.2 State model

Thus, motivated by the smaller computational and memory complexity associated with keeping track only of actual (uncoded) packets decoded by the users, from now

on we assume each receiver has $k-1$ buffers for storing at most one packet for each other receiver in the system. Namely, when a receiver overhears a packet intended to a different receiver, *or it can decode* a packet intended to a different user, it is able to store one such packet. This gives rise to our state model, which is at the basis of the system we propose. The state of the system at each time instant t is a $K \times K$ matrix S. The state matrix is initialized to the all-zeros matrix. When packets are received at the users, the state matrix updates as follows: for $k \neq k'$, $S_{k,k'} = 1$ if packet intended for user k was heard by user k' and $S_{k,k'} = 0$ otherwise. However, when a packet intended for user k is received by that user, $S_{k,k}$ remains 0. This is since we assume once a packet is received at the intended receiver, it immediately sends an acknowledgement, together with his index[a]. The sender thus knows that the packet was received, and that this user is now awaiting *its next packet*. Hence, $S_{k,k}$ remains 0. Moreover, since the acknowledgement was received by all other users as well, the users which buffered this packet can discard it, as it will no longer be used in coded packets. As a result, if a receiver k acknowledges receiving a packet, all other users who kept it discard it and we set $S_{k,k'} = 0$ for all k'. An example of the evolution of the state for three users is given in Table 1. Note that since there are three users, the state is represented by a 3×3 matrix. Additionally, the table only depicts the evolution of the states, assuming the actions (packets sent) as well as their results (where were the packets received) are as given in the table.

Clearly, it is also important to depict the actual *state transition matrix*, given the memoryless probabilities of packet losses. For the sake of clarity, we depict here only the two-users case. Table 2 includes the possible states. Table 3 includes the state transition matrix and reward vector with *uncoded* transmissions and equal loss probabilities p. In this policy, marked by π, the sender sends a packet at random, either to the first user or to the second, with equal probability. Table 4 includes the *coded* scenario, with unequal loss probabilities p_a and p_b. In this policy, marked by $\bar{\pi}$, since there are only two users, and users do not save coded packets, the only coding opportunity is when each user has a packet intended to the other (state S_4). In any other state, packet is sent at random, either to the first user or to the second, with equal probability.

Table 1 Evolution of the state for three receivers

t	t+1	t+2	t+3
Initial state	P_1^1 received by users 2, 3	P_1^2 received by user 1	$P_1^1 + P_1^2$ received by users 1, 2
$\begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}$	$\begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix}$	$\begin{pmatrix}0&1&1\\1&0&0\\0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}$
$t+4$	$t+5$	$t+6$	$t+7$
P_2^2 received by user 3	P_2^2 received by users 1,2	P_1^3 received by users 1,2	P_3^2 received by users 1,2
$\begin{pmatrix}0&0&0\\0&0&1\\0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\\0&0&0\\1&1&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\\0&0&0\\1&1&0\end{pmatrix}$

Table 2 State space for the two users case (coded and uncoded)

S_1	S_2	S_3	S_4
$\begin{pmatrix}0&0\\0&0\end{pmatrix}$	$\begin{pmatrix}0&1\\0&0\end{pmatrix}$	$\begin{pmatrix}0&0\\1&0\end{pmatrix}$	$\begin{pmatrix}0&1\\1&0\end{pmatrix}$

3 Markov decision processes—preliminaries

In this section, we give a brief description of the decision process model we use, the required notation and known methods and results which are relevant to this study and will be used throughout. A detailed, more thorough review of decision processes and reinforcement learning can be found in [5-7].

We consider a discrete time time-axis, $T = 0, 1, 2, \ldots$. Note that we do not consider a fixed time N denoting an end state, and rather consider an infinite time model. We assume the system is defined on a finite state space $\mathcal{S}$. As mentioned, in our model these states represent the knowledge at the terminals. At each time instant t, the sender takes an action $a_t(s_t)$, where s_t is the current state. The actions belong to a predefined set $\mathcal{A}$. Without loss of generality, $\mathcal{A}$ includes all possible actions from all states at all times. As an example, actions may include sending uncoded messages, sending coded messages, etc.

We assume a Markovian state transition structure, that is

$$\begin{aligned} &p_t(s_{t+1} = s' | s_t, a_t) \\ =&p_t(s_{t+1} = s' | s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0), \end{aligned}$$

and further assume stationarity, i.e., $p_t(s_{t+1} = s' | s_t, a_t) = p(s' | s, a)$. We assume the policies which govern the actions taken are Markovian, that is, $a_t = \pi_t(s_t)$. Hence, the policy depends only on the current state. It is also beneficial to assume that the policies are stationary, that is, $a_t = \pi(s_t)$. As a result, the degree of freedom available to the sender is in choosing the function $\pi : \mathcal{S} \mapsto \mathcal{A}$, its stationary control policy. Note that $\pi(s)$ can be a stochastic function. That is, at a certain state the system can choose at random who to send a packet to, or choose the coding coefficients at random. As mentioned, the stationarity assumption allows us to efficiently solve the equations for the optimal policy,

Table 3 State transitions for the two users case: uncoded transmission ($p_a = p_b = p$)

State transition matrix P^{π}	Reward vector r^{π}
$\begin{pmatrix} p^2+(1-p) & \frac{1}{2}p(1-p) & \frac{1}{2}p(1-p) & 0 \\ \frac{1}{2}(1-p) & p+\frac{1}{2}(1-p)^2 & 0 & \frac{1}{2}p(1-p) \\ \frac{1}{2}(1-p) & 0 & p+\frac{1}{2}(1-p)^2 & \frac{1}{2}p(1-p) \\ 0 & \frac{1}{2}(1-p) & \frac{1}{2}(1-p) & p \end{pmatrix}$	$\begin{pmatrix} 1-p \\ 1-p \\ 1-p \\ 1-p \end{pmatrix}$

describe the asymptotic system behavior, and gain insight on the benefits of different strategies.

With each current state, action and next state, we associate a reward $r(s,a,s')$. The reward associated with the states and action might reflect both the benefit of transition (e.g., decoding packets) and the cost in the specific action (e.g., the computational complexity in constructing a coded message). Again, the reward model does not depend on time, but can be stochastic, that is, with some probability packets are decoded and the reward is high and with some probability there is a loss or a decoding failure. Moreover, we will be interested in the *discounted reward* in the asymptotic regime, that is

$$V_{\gamma}^{\pi}(s) = E_{\pi}\left\{\sum_{t=0}^{\infty}\gamma^{t} r(s_t, a_t, s_{t+1}) \mid s_0 = s\right\},$$

where π is the policy used and s is the initial state. $0 < \gamma < 1$ is the *discount factor*. It determines the amount of memory in the performance measure. That is, for $\gamma \to 0$, the value per state is mainly affected by the current reward, and does not take into account future transitions, while $\gamma \to 1$ weights almost identically the entire sequence. $V_{\gamma}^{\pi}(s)$ is thus the asymptotic cumulative discounted reward of the system. Clearly, our goals are to compute $V_{\gamma}^{\pi}(s)$ for a given policy and find the optimal policy in terms of minimizing $V_{\gamma}^{\pi}(s)$. For the infinite horizon, stationary model we discuss in this article, these two goals are within reach. This way, it will be possible to, for example, understand when coding should take place and when uncoded packets are optimal and what is the resulting throughput in the system for each scheme.

The main two results for MDP in the stationary regime are the following.

Theorem 1 (e.g., [5,6]). *Let π be a stationary policy. Then, the asymptotic discounted reward V_{γ}^{π} is the unique solution to the following set of linear equations:*

$$V_{\gamma}^{\pi}(s) = \sum_{s' \in \mathcal{S}} p(s'|s,\pi(s))\left[r(s,\pi(s),s') + \gamma V_{\gamma}^{\pi}(s')\right],\ s \in \mathcal{S}. \tag{1}$$

To facilitate a vector representation, we denote by $V_{\gamma}^{\pi} \in \mathbb{R}^{|\mathcal{S}|}$ the vector of values achieved by the policy π, by P^{π} the state transition matrix under π, that is $P^{\pi}(s'|s) = p(s'|s,\pi(s))$, and by r^{π} the expected reward

$$r^{\pi}(s) = \sum_{s' \in \mathcal{S}} p(s'|s,\pi(s)) r(s,\pi(s),s').$$

Under this notation, we have

$$V_{\gamma}^{\pi} = \left(I - \gamma P^{\pi}\right)^{-1} r^{\pi}, \tag{2}$$

that is, in the stationary asymptotic regime, the total rewards achieved by each policy are easily calculated using a linear system. In our setting, (2) will be used to assess the value of a given policy, and compare it to other policies. For example, assess the value achieved by a certain coding policy compared to an uncoded one. However, in certain cases, it is interesting to compute the value function of the optimal policy directly, as well as the optimal policy itself. For this, the following results come in handy. In this case, however, we assume deterministic policies[b]. Define the following operator from $\mathbb{R}^{|\mathcal{S}|}$ to $\mathbb{R}^{|\mathcal{S}|}$, $T_{\gamma}^{*} : (T_{\gamma}^{*}V)(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p(s'|s,a)\left[r(s,a,s') + \gamma V(s')\right]$.

Theorem 2 (e.g., [5,6]). *For any initial value function V_0, we have*

Table 4 State transitions for the two user case: coded transmission

State transition matrix $P^{\bar{\pi}}$	Reward vector $r^{\bar{\pi}}$
$\begin{pmatrix} \frac{1}{2}[(1-p_a)+(1-p_b)] + p_a p_b & \frac{1}{2}p_a(1-p_b) & \frac{1}{2}(1-p_a)p_b & 0 \\ \frac{1}{2}(1-p_a) & \frac{1}{2}p_a + \frac{1}{2}[(1-p_b) + p_a p_b] & 0 & \frac{1}{2}(1-p_a)p_b \\ \frac{1}{2}(1-p_b) & 0 & \frac{1}{2}[(1-p_a)+p_a p_b] + \frac{1}{2}p_b & \frac{1}{2}p_a(1-p_b) \\ (1-p_a)(1-p_b) & p_a(1-p_b) & (1-p_a)p_b & p_a p_b \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2}(2-p_a-p_b) \\ \frac{1}{2}(2-p_a-p_b) \\ \frac{1}{2}(2-p_a-p_b) \\ 2-p_a-p_b \end{pmatrix}$

Table 5 Values of V_γ^π and $V_\gamma^{\bar{\pi}}$ with a discount $\gamma = 0.5$ and different values of packet loss probability p

Packet loss and policy	S_1	S_2	S_3	S_4
$p = 0.1$, uncoded	1.8	1.8	1.8	1.8
$p = 0.1$, coded	1.80231	1.82803	1.82803	2.708
$p = 0.5$, uncoded	1	1	1	1
$p = 0.5$, coded	1.01111	1.05556	1.055556	1.58889

1. *Value iteration: If $V_{n+1} = T_\gamma^* V_n$, then $V_n \to V^*$, the value function under the optimal policy.*
2. *Any policy π^* which satisfies $\pi^*(s) = \operatorname{argmax}_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[r(s,a,s') + \gamma V^*(s')\right]$ is an optimal policy.*
3. *Policy iteration: Assume $\bar{\pi}$ is defined according to $\bar{\pi}(s) = \operatorname{argmax}_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[r(s,a,s') + \gamma V^{\pi}\right]$ for some policy π. Then $V^{\bar{\pi}} \geq V^{\pi}$ with equality iff π is an optimal policy.*

4 A Markov decision process based approach

As mentioned in Section 3, in the stationary model under discounted reward, suggested policies can be evaluated analytically, with a closed form solution. We first give here a basic 2-users example. An access point has two streams of packets, one for each of the two users. The four possible states are the ones given in Table 2. Packets are received at each receiver with probability $1 - p$, independently of the other. Each time a receiver decodes *its intended packet*, a reward of 1 is received. The transition probability matrices and reward vectors were given in Tables 3 and 4. We can now find V_γ^π and $V_\gamma^{\bar{\pi}}$ directly according to (2). The solutions for different values of the packet loss probability p are given in Table 5. Note that the value function in the coded case is higher than that of the uncoded case for all states $S_i, i \in 1,\ldots,4$. Clearly, while the value function is constant for all states in the uncoded policy, in the coded policy it is significantly higher for S_4.

To conclude this example, we calculate the stationary distributions associated with each of the two policies—the uncoded and the coded one. That is, let v_s^π and $v_s^{\bar{\pi}}$ be the stationary distributions of the uncoded and coded policies, respectively. We have

$$v_s^\pi = (\nu_1,\ldots,\nu_{|\mathcal{S}|}): \quad (\nu_1,\ldots,\nu_{|\mathcal{S}|})^T P^\pi = (\nu_1,\ldots,\nu_{|\mathcal{S}|})^T. \tag{3}$$

$v_s^{\bar{\pi}}$ is calculated in a similar fashion. The total expected discounted reward is thus $R^\pi = (v_s^\pi)^T V_\gamma^\pi$ for the uncoded case and similarly $R^{\bar{\pi}}$ for the coded one. The results for different values of p are given in Table 6. Note that since we consider *cumulative discounted reward*, the total system throughput under a policy π, *in packets delivered per time slot*, is $(1-\gamma)R^\pi$.

The benefit in the coded policy is clear, and it grows larger as the packet loss probability increases. This will also be clear in the simulations. Note, however, that the system of equations is of dimension $|\mathcal{S}|$. For this reason, we include more efficient methods in the next sub-sections.

4.1 A greedy algorithm

The model discussed thus far, allowed us to analytically compute the reward associated with each state and action pair, as well as the expected discounted reward for a given policy. Moreover, using Theorem 2, it is possible to compute the optimal policy. However, in practical situations, the equations given in Theorem 2 are computationally intractable. Note that for K users, the state matrix is of size $K \times K$, with the K diagonal values fixed at 0. Thus, the state space is of size $2^{K(K-1)}$, making it practically impossible to list all states, a fortiori compute all entries of the state transition matrix analytically. Using a reinforcement learning approach, such as Q-learning, would also be intractable as is with such a state space. The reason is, at the basis on such methods, e.g., deterministic Q-learning, stands an update equation, $Q(s_n, a_n) \leftarrow r_n + \gamma \max_{a'} Q(s_{n+1}, a')$, where s_n and s_{n+1} are the current and next states, respectively, and r_n is the received reward. Thus, to be able to update Q, the system needs to keep track of Q values for all possible state-action pairs.

However, keeping track of the state *at a given time*, as well as understanding the next state, given the current one, the action taken and the channel status, is certainly a

Table 6 Stationary distribution and expected commutated discounted reward with a discount $\gamma = 0.5$ and different values of packet loss probability p

Packet loss and policy	S_1	S_2	S_3	S_4	Expected reward
$p = 0.1$, uncoded	0.826446	0.0826446	0.0826446	0.00826446	1.8
$p = 0.1$, coded	0.838028	0.0774648	0.0774648	0.00704225	1.81268
$p = 0.5$, uncoded	0.444444	0.222222	0.222222	0.111111	1
$p = 0.5$, coded	0.5	0.214286	0.214286	0.0714286	1.07143

tractable solution, with quadratic number of operations in each step and quadratic memory requirements. Thus, in this part of our study, we seek efficient algorithms which not only base their action only on the current state, but also do not require tracking rewards for previous states or solving directly systems of equation which are of the same size as the state space. We will later see, that while heuristic in nature, the *greedy and semi-greedy* algorithms we suggest, are proved to be efficient, and improve performance significantly compare to the un-coded version.

Consider the state matrix S seen at the access point at a given time. A *greedy* algorithm will aim at maximizing the instantaneous reward received for its action at that state (as opposed to maximizing an expected or discounted reward, which, according to the Belman equations in Theorem 2, takes into account the expected rewards at future states). For two matrices A and B, denote by $A \odot B$ the entry-wise multiplication of A and B, that is, the matrix whose entries are $a_{i,j}b_{i,j}$. We first have the following result.

Lemma 1. *Let S be the state matrix at a given time. Assume packet loss probabilities for all receivers are equal. Then, the action which maximizes the instantaneous reward is sending a packet which includes a XOR of the packets intended to a set of users v which form a maximal clique in the undirected graph whose adjacency matrix is the upper triangle of $S \odot S^T$.*

Proof. A reward is received if and only if a receiver decodes its *intended* packet. Thus, to maximize the total *instantaneous* reward, the sender should aim at maximizing the number of receivers who decode their intended packet at the current round.

Consider a single receiver. This receiver can decode its packet if and only if it is included in the coded packet received, together *only with a subset of packets the receiver overheard* and buffered (the empty set is this case represents an uncoded packet). Note that receivers do not buffer packets from which they cannot decode a packet immediately. In the matrix notation of our model, receiver i can decode its packet only if the action taken at that round XORs the packets intended to receivers indexed by the support of column i of S, together with its own intended packet.

Now, consider two receivers, i and j, and assume the sender wishes to satisfy both (hence receive a reward for both). Clearly, the coded packet should include both packets in the XOR, otherwise at least one of them will not be able to decode. However, this means receiver i must have the packet intended to j and vice versa. That is, $S(i,j) = S(j,i) = 1$. Thus, to satisfy $n \leq K$ receivers $i_1, \ldots, i_n$ simultaneously, the coded packet must XOR at least those n packets, and the $n \times n$ sub-matrix consisting of only rows and columns $i_1, \ldots, i_n$ of S must have all its entries as 1 except the diagonal. This sub-matrix corresponds to a clique of size n in the directed graph whose adjacency matrix is S, or in the undirected graph whose adjacency matrix is the upper triangle of $S \odot S^T$. Clearly, in this case, to receive a reward n there is no need to code on extra packets, besides the n intended to these users. This completes the proof. □

We call the undirected graph whose adjacency matrix is the upper triangle of $S \odot S^T$, the graph induced by S. Lemma 1 thus gives rise to a conceptually simple policy. We summarize this policy in Algorithm 1. The idea behind the policy is simple: find the largest cliques in the graph induced by S. If there are several largest cliques, choose one at random, with uniform probability on the maximal cliques. If there are no cliques—send a random uncoded message. Note that for the case of unequal loss probabilities, the requirements stated in the Proof of Lemma 1 remain: in order to receive a reward, a user must code its own packet, and this can be done if and only if the coded packet received includes its intended packet and packets it overheard and buffered. However, due to the unequal error probabilities, the *sender might prefer* smaller cliques in which the receivers have lower loss probabilities, compared to larger cliques with high loss probability. Thus, the modification of the algorithm to this setting is straight forward.

Algorithm 1 Greedy Policy (S)

```
action = (0, ..., 0)
X = UpperDiag(S ⊙ S^T)
{C_1, ..., C_l} = FindMaxCliques(X)
if {C_1, ..., C_l} = φ
    then i = RandomIndex(K)
         action(i) = 1
    else clique_index = RandomIndex(l)
         for i = 1 to max_clique_size
         do action(C_clique_index(i)) = 1
return action
```

At first sight, Algorithm 1, involves a computationally hard problem, since the problem of finding a maximal clique in a graph is known to be NP-complete [33]. Moreover, there are graphs with exponentially many large cliques [34], and Algorithm 1 requires to list all and choose one at random (this is done to avoid starvation of cliques not listed first when one maximal clique is sought). However, as the next lemma asserts, with high probability, the graph induced by S has cliques of at most logarithmic size (in K), hence a *polynomial time algorithm* which searches for bounded size cliques can approximate closely the performance of the best greedy scheme, and a *sub-exponential time* algorithm is asymptotically optimal.

Lemma 2. *Assume the system starts at the initial all-zero state, and proceeds according to Algorithm 1. Assume equal packet loss probabilities. Then, at each stage of the algorithm, with high probability, the largest clique in the graph induced by the state matrix S is of size $O(\log K)$.*

Proof. The proof is by considering the evolution of the state sequence up to a give time $t \in \mathbb{Z}^+$, and is based on the celebrated results in [35,36] regarding cliques in random graphs. For a random adjacency matrix, having 1 with probability $1-q$, the size of the largest clique, X_K, satisfies $X_K/\log(K) \to 2/\log(1/(1-q))$ with probability 1. Thus, we wish to show that with high probability, the worst the state S can be, is a random i.i.d. adjacency matrix, with probabilities $(q, 1-q)$ for some *fixed* q.

At slot zero, that is, $t=0$, S contains no cliques. As long as there are no cliques, packets are sent uncoded. Assume an uncoded packet is sent to receiver i. Since this packet is received at each receiver, including i itself, independently of the other receivers with probability $1-p$, p being the packet loss probability, this results in an OR operation between the current ith row in S *and a random row*, having ones with probability $1-p$ and zeros otherwise. That is, an OR between the current state and a vector representing the receivers of the packet in the current slot. The OR operation is since some receivers might have buffered the packet in a previous transmission. Now, if receiver i received its intended packet, the whole line in S will be set to zero immediately. Otherwise, each entry in the row will have, independently of the other entries, 1 with probability $1-p^r$, r being the number of times the current packet intended to user i was transmitted since the last time this user decoded a packet (i.e., since the last time the row was zeroed). In other words, each entry is still i.i.d., yet with a much larger probability for 1. Yet, as long as r is finite, $1-p^r$ is bounded away from 1.

Clearly, if cliques are found, and packets are sent coded, then users which decoded their intended packets zero the corresponding rows, hence more than one row can be zeroed. Users which did not decode, may either remain in the same state, or decode *other packets*. However, from a received packet, a user can decode at most one packet, if received correctly (similar to the uncoded case), so the distribution of the rows either does not change compared to the uncoded case or includes more zero rows.

We now wish to assess the probability of finding a clique of size larger than $2\log K/\log(1/(1-q))$, for some fixed $q>0$. Take $q<p$ and set $r=\left\lfloor \frac{\log q}{\log p} \right\rfloor$. At time t, a row will have ones with a Bernoulli(q) distribution only if in the last r transmissions which included the intended packet, it was not decoded by the intended receiver. This happens with probability at most p^r. However, to have cliques of size larger than $2\log K/\log(1/(1-q))$ with a probability bounded away from zero, we need $\Omega(K)$ rows with high density. This happens with probability p^{Kr}, which is small for large enough K. □

At this point, a few remarks are in order. The first important consequence of Lemma 2, is that if the procedure FindMaxCliques(X) in Algorithm 1 *is replaced with a one which searches for cliques of bounded size*, the algorithm is guaranteed, with high probability, to yield the same performance as the one with the original procedure, only now the complexity is polynomial in K. This is since it can, at worst, exhaustively search for such cliques. In particular, fix some $0<q<p$. The probability of finding a clique of size larger than $2\log(K)/\log(1/(1-q))+1$, is at most $p^{K\left\lfloor \frac{\log q}{\log p} \right\rfloor}$. Thus, with large enough K, this probability can be made arbitrarily small for any $q<p$. Numerical results for the performance of Algorithm 1 in various setting are given in Section 5. An important observation from the results therein, is that in practice, the sizes of the largest cliques is small, rendering the computational problem relatively easy while still allowing for a high percentage of coding gain over the uncoded scheme.

The results in Lemmas 1 and 2 were stated only for equal loss probabilities on all links. However, this assumption was merely for the ease of presentation. It is straightforward to extend these results to unequal probabilities, as long as the packet loss probability is bounded away from 0. The only difference is that then cliques should be weighted by their *expected reward*, given the various loss probabilities of the members of the clique. For equal loss probabilities, the expected reward is a linear function of the size (with a constant which depends on p), hence the algorithm searches for maximal cliques.

4.2 A semi-greedy approach

Clearly, the drawback of Algorithm 1 is in its inability to foresee future states with large rewards. As soon as a clique is identified, a coded packet intended to that clique is sent. However, it is obvious that if one could devise a policy targeted at *setting the grounds* for states with larger rewards, yet without redundant packets, it will perform better. Of course, the optimal solution is solving the backwards equations based on Belman's criterion (finite horizon) or using value or policy iteration (infinite horizon). Yet, these solutions do not scale up to large systems, with a large number of users.

In this sub-section, we propose a *semi-greedy* approach, which performs significantly better than the greedy one (see Section 5), yet does not require the complexity burden of tracking rewards per state, which is infeasible merely due to the large number of states, nor does it require sending redundant packets. At the heart of this

approach stands the following observation: empty rows in the state matrix S, reduce the probabilities of finding large cliques significantly. Hence, the semi-greedy approach will first *send uncoded packets to users whose packets were not heard by any other user*, and only if no such empty rows exist, it will send coded packets to the largest cliques. As a result, the steady state of the system will tend to a denser matrix, with higher probabilities for large cliques. The policy is summarized in Algorithm 2.

Algorithm 2 Semi-Greedy Policy (S)

$action = (0, \ldots, 0)$
$\{i_1, \ldots, i_l\} = \text{FindEmptyRows}(S)$
if $\{i_1, \ldots, i_l\} = \phi$
  $action = \text{GREEDY POLICY}(S)$
  else $row\_index = \text{RandomIndex}(l)$
    $action(row\_index) = 1$
return *action*

It is important to note, though, that the logarithmic bound on the size of the largest clique will still hold in this case, as any row in the state matrix for the semi-greedy algorithm will still be an i.i.d. row with ones at some probability $0 < q < 1$ (if it is not the all-zeros row intentionally).

Corollary 1. *Assume the system starts at the initial all-zero state, and proceeds according to Algorithm 2. Then, at each stage of the algorithm, with high probability, the largest clique in the graph induced by the state matrix S is of size* $O(\log K)$.

Proof. The corollary follows from the same reasoning Lemma 2 follows. The difference, however, is in keeping the state matrix S with as less empty rows as possible. Yet, as an empty row in S causes the access point to send an uncoded packet, the empty row will remain empty with probability $1-p$ (the packet was decoded by the intended user), or be replaced with a row of random i.i.d. entries ($1-p$ being the probability for 1 and p for 0), in which case it will contribute to the graph at most like a row in a random adjacency matrix of a graph, as it is drawn uniformly and independently of the other rows in the matrix. □

It is important to note that the benefits in Algorithm 2 are intimately related to multi-user diversity gain in wireless systems [37]. To see this, note that a user with a relatively better channel than the others, is more likely to have the corresponding line in the state matrix S zeroed. Thus, the impact of sending packets to users whose corresponding rows are all-zero, is in *serving the users whose channel states are better at the current time slots*. When channel conditions change, and different users observe better channels, the focus will switch to these users, yielding, on the average, better performance, without compromising fairness. This is also very clear in the simulation results given in Section 5.

4.3 Performance analysis for two users

We consider an asymmetric channel model, where p_a and p_b are the error probabilities of packets from the access point to users a and b, respectively. We focus on the average capacity given in terms of delivered packets per slot, rather than the discounted reward.

The sender may transmit a coded packet only when the system is at state S_4. Indeed, at this state a clique is formed. At all other states there is no clique, and the greedy scheme would transmit a random uncoded packet. The system state transitions matrix and reward vector were given in Table 4 ("coded transmission"). With the transition matrix at hand, it is easy to devise the system stationary probabilities Π, as well as the average system reward C (i.e., system capacity in packets per slot). For the symmetric case of $p_a = p_b = p$,

$$\Pi = \left[\frac{1+2p-p^2}{1+4p+2p^2}, \frac{p(1+p)}{1+4p+2p^2}, \frac{p(1+p)}{1+4p+2p^2}, \frac{p^2}{1+4p+2p^2}\right], \tag{4}$$

and

$$C = \frac{1+3p-p^2-3p^3}{1+4p+2p^2}. \tag{5}$$

In the semi-greedy algorithm, the difference is in states S_2 and S_3. In S_2 (S_3), an uncoded packet is sent to user b (a) with probability 1. The system state transitions matrix and reward vector are given in Table 7.

Table 7 State transitions for the two terminal case: semi-greedy scheme

State transition matrix $P^{\hat{\pi}}$				Reward vector $r^{\hat{\pi}}$
$\frac{1}{2}[(1-p_a)+(1-p_b)]+p_ap_b$	$\frac{1}{2}p_a(1-p_b)$	$\frac{1}{2}(1-p_a)p_b$	0	$\frac{1}{2}(2-p_a-p_b)$
0	$(1-p_b)+p_ap_b$	0	$(1-p_a)p_b$	$1-p_b$
0	0	$(1-p_a)+p_ap_b$	$p_a(1-p_b)$	$1-p_a$
$(1-p_a)(1-p_b)$	$p_a(1-p_b)$	$(1-p_a)p_b$	p_ap_b	$2-p_a-p_b$

The following equations depicts the stationary probabilities and the average capacity for the symmetric case of $p_a = p_b = p$,

$$\Pi = \left[\frac{1-p}{2+p}, \frac{1+p}{4+2p)}, \frac{1+p}{4+2p)}, \frac{p}{2+p}\right], \tag{6}$$

and,

$$C = \frac{2-2p^2}{2+p}. \tag{7}$$

We can now compare the system capacity obtained with the greedy and the semi-greedy scheme relatively to the average capacity of an uncoded scheme. Figure 1a,b depict the system capacity gain as a function of the error probabilities p_a and p_b. Table 8 depicts the users and system capacity, for various error probability values, for the (i) uncoded case, (ii) the greedy scheme, and (iii) the semi-greedy scheme. The figures in the table clearly show the advantage of the semi-greedy scheme in terms of system capacity. Notice that the first rows in Table 8 are identical to the first rows of Table 6 up to a factor of $\frac{1}{1-\gamma}$, which is exactly the sum of the geometric series of the discounted reward factor γ.

It is clear by now that the semi-greedy scheme obtains a higher capacity than the uncoded as well as the greedy schemes. Proposition 1 indicates that, for the two users case with a symmetric channel, i.e., $p_a = p_b = p$, this policy is optimal in terms of maximal average capacity (reward).

Proposition 1. *Consider the two users case with a single packet buffer size. The Semi-Greedy Policy is the optimal among all policies in terms of maximizing the average capacity.*

Table 8 Two users capacity values

Packet loss and policy	user *a*	user *b*	System
$p_a = 0.1, p_b = 0.1$, uncoded	0.45	0.45	0.9
$p_a = 0.1, p_b = 0.1$, greedy	0.453	0.453	0.906
$p_a = 0.1, p_b = 0.1$, semi-greedy	0.47	0.47	0.943
$p_a = 0.1, p_b = 0.2$, uncoded	0.45	0.4	0.85
$p_a = 0.1, p_b = 0.2$, greedy	0.455	0.404	0.86
$p_a = 0.1, p_b = 0.2$, semi-greedy	0.66	0.26	0.921
$p_a = 0.1, p_b = 0.4$, uncoded	0.45	0.3	0.75
$p_a = 0.1, p_b = 0.4$, greedy	0.457	0.305	0.762
$p_a = 0.1, p_b = 0.4$, semi-greedy	0.815	0.09	0.905

Proof. Intuitively, under our setting of a single packet buffer size, each packet should be transmitted uncoded at least once. Then, it could be retransmitted in coded packets. To maximize the average capacity, an optimal policy would minimize the additional uncoded retransmissions. Clearly, the semi-greedy scheme obtains this by sending each packet uncoded only once and sending all retransmissions coded. In addition, from a symmetry reasoning, at state S_0 it does not matter whether to transmit to user *a* or *b*. Accordingly, the semi-greedy policy transmits either to user *a* or *b* at equal probabilities. More formally, the proposition follows from the dynamic programming optimality equation (i.e., Bellman's equation) [5]. □

4.4 Fairness

Also clear from Table 8, is the potential unfairness of the semi-greedy scheme (unequal capacities due to unequal loss probabilities). As discussed previously, the semi-greedy scheme increases the system capacity by both

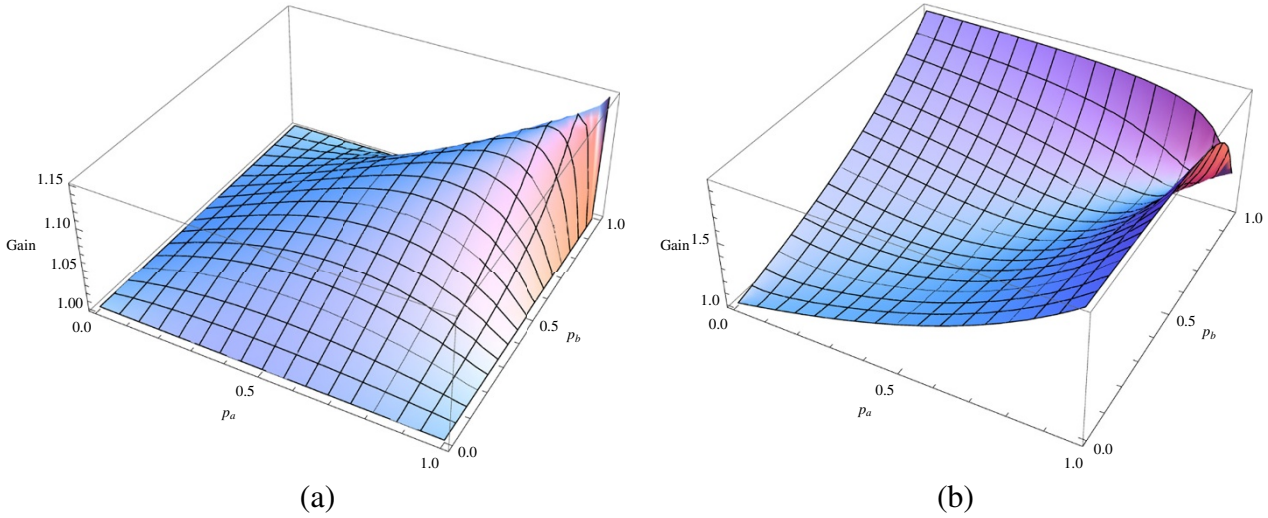


Figure 1 System capacity gain as a function of the error probabilities p_a and p_b. (a) Greedy **(b)** Semi-greedy.

maximizing the usage of coded retransmission as well as by *opportunistically transmitting to terminals with higher success probabilities*. Indeed, users with worse channel conditions may suffer from a short time fairness. With mobile users, it is expected that their channel condition would vary such that the long term fairness prevails.

However, a more rigorous solution to the problem is possible, even for the case when mobility alone does not suffice. To see this, consider the *expected, cumulative* reward vector of the semi-greedy scheme in Table 7. For $p_a < p_b$, it is clear that maximizing the reward results in preferring state S_3 over S_2, i.e., the system is biased towards user *a* decoding its packets. Nevertheless, an important benefit of the MPD-based approach in this article, is that this bias can be canceled using an appropriate weighting of the reward *given for each transition*. By increasing the reward *given for decoding a packet by user b*, compared to the reward given for decoding by user *a*, one can create an artificial bias towards user *b*. In fact, similar reward weighting can be used to impose *quality of service* constraints or any other fairness mechanism.

4.5 Missing acknowledgements

The schemes suggested in this article utilize an Automatic Repeat reQuest (ARQ) mechanism which is compliant with IEEE 802.11. Nevertheless, it is important to note that the requirement that packets be acknowledged immediately (which is mandatory for the IEEE 802.11 protocol) is not compulsory in our scheme which can tolerate some delay between the packet and its corresponding ACK. Accordingly, if a certain delay is acceptable, the access point can act according to reports received from the users, once such reports are indeed received, and send the right coded packets. Of course, this requires larger buffers at the stations. Furthermore, it is also important to note that the assumption that acknowledgements sent by the receivers are received correctly by other receiving stations is not mandatory and was made only for the sake of simplicity. Specifically, receiving the acknowledgments by the users is required only in order for a user to know if a packet intended to a different user should be kept or discarded. Thus, if an acknowledgement sent by a user is not received by other users, the uninformed users keep obsolete packets. Such packets could be discarded after a certain timeout, or once these users identify newer packets are being sent (by examining a sequence number in a packet). Hence, if users do not receive acknowledgements sent by others, there is no degradation in performance, only a requirement for slightly longer buffers.

The access point uses the acknowledgements in order to assess the state. Misinterpretation of the state can result in degradation of performance. Nevertheless, the theory of MDP includes well-established algorithms for cases when the state is only partially observed [38,39]. If the observation of the state is kept within some mild fidelity from the true state, these algorithms perform very well. Furthermore, in a WiFi architecture, which is at the basis of this article, immediate acknowledgements are compulsory, and the protocol does not function without a bidirectional communication between the access point and the user. Hence, assuming acknowledgements are received properly at the access point is reasonable.

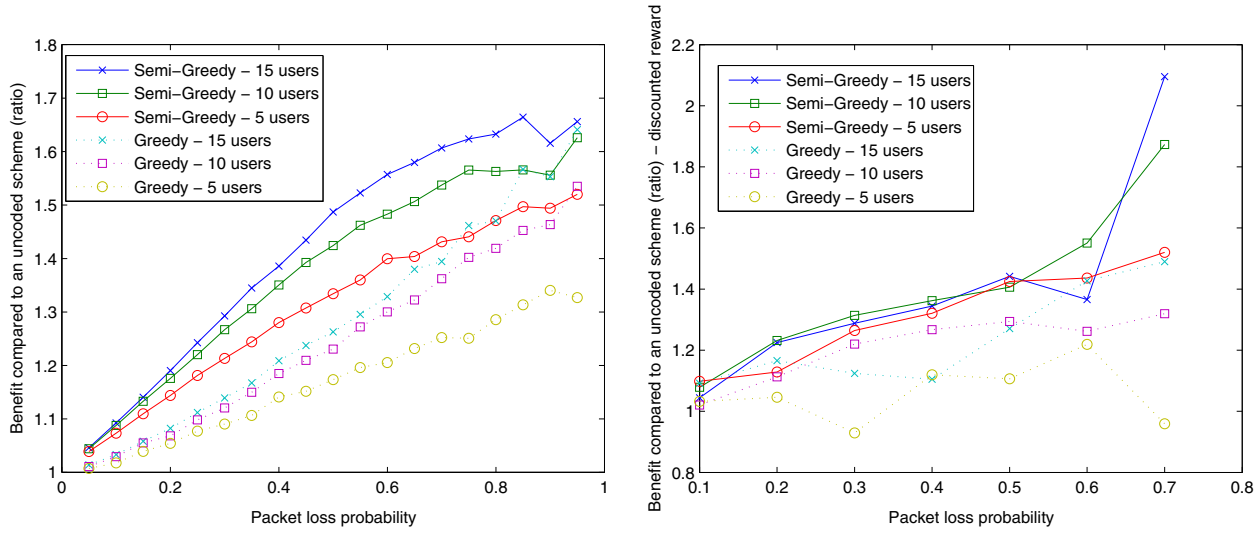
5 Numerical analysis

In this section, we evaluate and compare the *Greedy* and *Semi-greedy* algorithms by simulation. We show that both algorithms significantly improve the performance compared to the uncoded version. Furthermore, we test the algorithms in various channel conditions, and examine fairness issues and their ability to adjust to varying conditions. This implementation also verifies that the proposed algorithms can be executed for a large number of users, compared to the prohibitive complexity of listing the entire state space.

We implemented the two algorithms in MATLAB. We modeled the channel between the AP and user *i* according to Bernoulli distribution with parameter p_i. Accordingly, user *i* receives *each transmitted packet* with probability $1 - p_i$. We further assumed small coherence time (fast fading), i.e., the loss probability between two subsequent transmissions is *i.i.d.* and non-correlated channels, i.e., the loss probability between two different users is independent. Since both algorithms rely on finding the largest cliques in the graph induced by the state matrix, which as previously explained is known to be NP-complete in general, we have used the MATLAB function [40], which allows bounding the maximal clique size, hence bound the complexity according to Lemma 2. As baseline for comparison we used the traditional uncoded case.

We start by investigating the gain of the two algorithms as a function of number of users and as a function of loss probability. Accordingly, we varied the number of users between 5, 10, and 15 and assumed that the AP always has traffic to send to each user. For each different number of users, we examined the performance for various loss probabilities *p*, ranging between 0.05 to 0.95 (for completeness we ran each set up for all loss probabilities, including very high ones, even though such loss probabilities are not common in real networks). Each setup was run for 20000 cycles. The results are depicted in Figure 2a.

As expected, for very low loss probability the gain in using any retransmission scheme is small, as when loss probability approaches 0 there are no retransmissions, neither in the uncoded nor in the coded schemes (hence no coding opportunities). For example for loss probability of 0.05 for 10 users the gain over the uncoded scheme is 1% and 4%, for the *greedy* and *semi-greedy* algorithms, respectively. As can be seen in the figure, the greater the

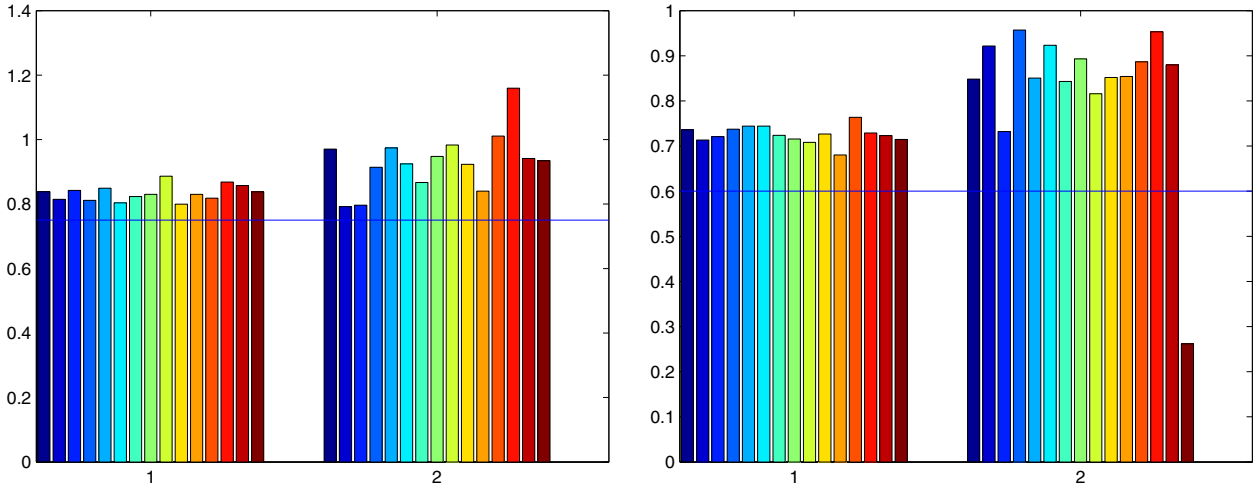


(a) Average Reward (b) Discounted Reward

Figure 2 The benefit in the Semi-Greedy and Greedy approaches compared to the uncoded scheme. (a) Average reward ($\gamma = 1$, normalized by the number of cycles). **(b)** Discounted reward ($\gamma = 0.95$).

loss probability the higher the coding opportunities, hence the higher the gains. For example for 10 users and loss probability of 0.5 the gain over the uncoded scheme is 23% and 42%, for the *greedy* and *semi-greedy* algorithms, respectively. Since the same holds also for number of users, i.e., the higher the number of users the greater the coding opportunities, the gain is an increasing function of the number of users. Furthermore, even though both algorithms offer high gains over the uncoded scheme, Figure 2a clearly depicts that the *semi-greedy* algorithm provides much higher gains than those of the *greedy* algorithm. For example, for loss probability of 0.3 the gain of the *semi-greedy* algorithm is 2.2 times, 2.4 times and 2.1 times higher than those of the *greedy* algorithm, for the 5, 10, and 15 users, respectively.

To better understand the gain of the two algorithms, Figure 3 shows the average rewards seen by each user, for the 15 users setup for different loss probabilities (the



(a) Packet loss probability 0.25 (b) Packet loss probability 0.4

Figure 3 Histograms of the average rewards seen by each user. Left bars—Greedy algorithm. Right—Semi-Greedy. The Horizontal line represents the expected reward of the uncoded policy. Values are normalized by the number of users. **(a)** Packet loss probability 0.25. **(b)** Packet loss probability 0.4.

left bars correspond to the greedy algorithm and the right to the semi-greed). Each bar in the histogram represents the average reward seen by each user, where the average reward obtained by each user is normalized with respect to the number of users. The vertical line represents the average reward per transmission for the uncoded case. As can be seen in the figure, the average reward seen by each user is high compared to the uncoded case for all loss probabilities for both algorithms. Recall that the expected reward for packets which are sent uncoded for both algorithms is $(1-p)$, which equals the one seen by the uncoded approach. It is interesting to note that variance of average reward obtained by different users is quite high.

Next, we examined the rewards per transmission. Figure 4 shows a reward distribution comparison between the two algorithms for loss probability of 0.5 and 15 users. As expected, the semi-greedy algorithm yields greater average reward per transmission, confirming its superiority in terms of throughput gain over the *greedy* algorithm. Note also that the reward is a lower bound on the clique size to which the coded packet was transmitted. That is, the reward distribution gives good indication regarding the number of packets XORed in a coded packet, and a good indication regarding the sparseness of the matrix, and accordingly the complexity of finding the cliques. As can be seen in the figure, for both algorithms the matrices are relatively sparse.

Figure 2b depicts the asymptotic cumulative *discounted* reward of the system when $\gamma = 0.95$ for the *semi-greedy* and the *greedy* approaches compared to the uncoded scheme. The dependency on the error probability, the benefit over the uncoded scheme and the benefit of the semi-greedy algorithm over the greedy one are similar to those seen in the average reward setting. Note, however, that due to the exponential discount, results are effectively averaged over a much smaller window size, and hence are noisier.

Next, we compared the two algorithms when the channels experienced by different users were not identical, i.e., different loss probabilities for different users. We ran a setup of ten users, where each user had different loss probability. Specifically, the loss probability ranged from 0.05 for user 1 to 0.5 for user 10. Figure 5 depicts the results where the x-axis presents the user I.D. (which corresponds to the loss probability, i.e., loss_probability $=$ $0.05 \cdot$ userID). The y-axis depicts the normalized user throughput.

As can be seen, regarding throughput, the *semi-greedy* algorithm provides a higher average throughput, nonetheless, regarding fairness, the *greedy* algorithm is much more fair than the *semi-greedy* one. The *greedy* algorithm distributes the throughput quite evenly, giving only slight advantage to users with low loss probability, e.g., the users with 0.05 and 0.5 loss probabilities get throughput of 0.08 and 0.06, respectively. The *semi-greedy* algorithm gives many more transmission opportunities to users with good channel quality (low loss probability).

Finally, we continued examining the performance of the two algorithms when the user's loss probabilities were not constant. We modeled each channel as a Markov channel, where the channel of each user alternated between good and bad. The loss probabilities were 0.05 and 0.5 for the good and bad channels, respectively. The transition probability between the two states was 0.01. Figure 6

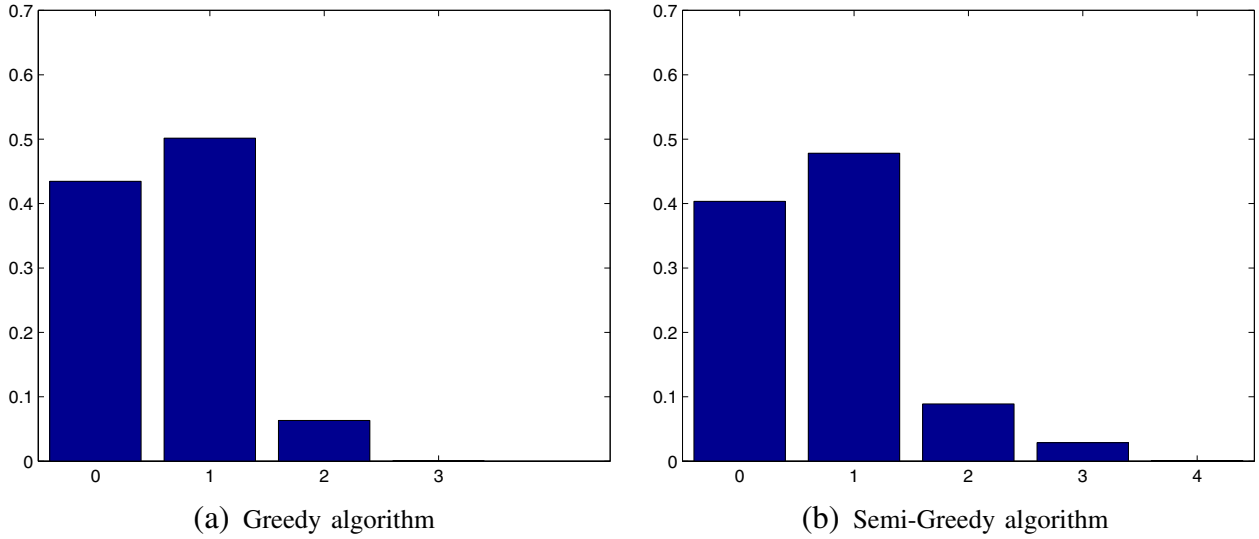


Figure 4 Reward distribution per iteration for packet loss probability 0.5 and 15 users. For both algorithms, clique sizes are moderate and can be found efficiently. **(a)** Greedy algorithm. **(b)** Semi-Greedy algorithm.

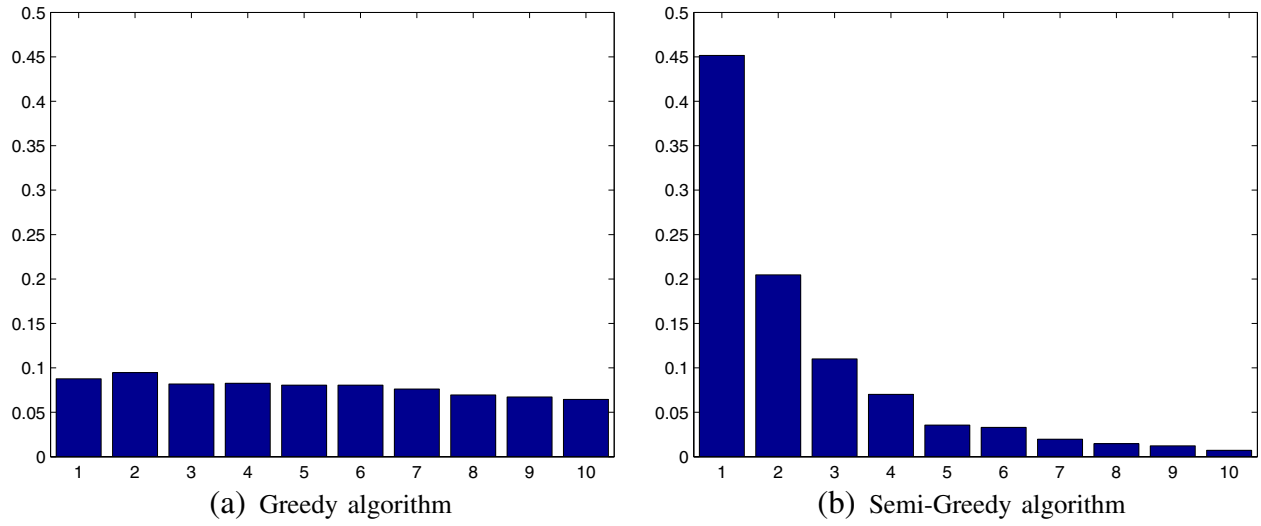


Figure 5 User throughput with unequal packet loss probabilities. Probabilities range from 0.05 for user 1 to 0.5 for user 10. **(a)** Greedy algorithm. **(b)** Semi-Greedy algorithm.

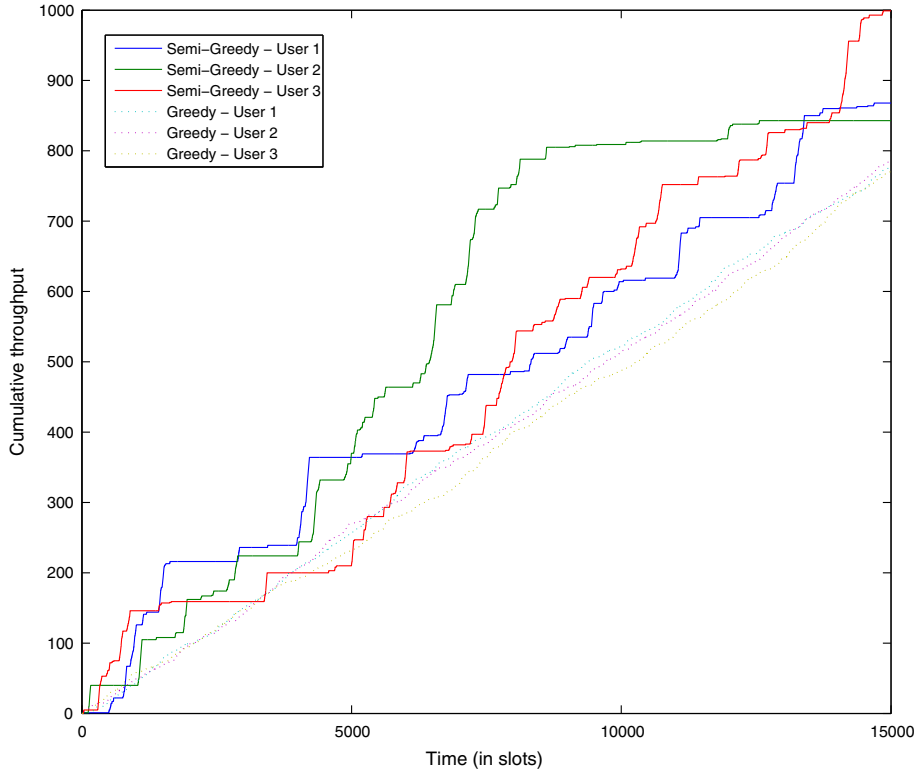


Figure 6 Cumulative throughput of the users for Markov channels, with a loss probability 0.05 at the "good" state and a loss probability of 0.5 at the "ba" state. The state transition probability is 0.01. In the greedy algorithm, the throughput increases at a constant rate, determined by the average loss probability seen by the user. However, the semi-greedy algorithm exploits the multi-user diversity, serving users with good channels while neglecting ones with bad channels. As a result, users with good channels experience a very steep increase in the reward, and the total system throughput is higher in the semi-greedy approach.

depicts the results for the case of three users. The x-axis depicts the time (in this setup, 15000 rounds), and the y-axis shows the cumulative throughput for each of the three users. As can be seen in the figure, the semi-greedy algorithm works in an opportunistic fashion, i.e., serving users with good channels while deferring the ones with the bad channels. Obviously, the aggregate system throughput is higher in the semi-greedy approach. Still, in the long run, the system is quite fair, giving all users approximately the same throughput.

6 WiFi implementation and testing

In this section, we consider the design and implementation of coded retransmissions in an IEEE 802.11 (WiFi) wireless network, operating in infrastructure mode (commonly termed WLAN). Note that this mode of operation is perfectly suited to the suggested schemes, as all traffic passes through the access point.

In order to allow 802.11 devices to support the NC suggested schemes, besides the basic network operation such as coding and decoding messages or implementing the decision policy at the AP, only a few modifications which are related to the standard are required. Specifically it is required to: (I) allow users to accept packets of which they are not the addressee, and to realize local buffers to store such packets; (II) stop the automatic MAC layer retransmissions (i.e., to set the *dot11LongRetryLimit* to one); (III) modify 802.11 MAC headers to incorporate network coding information; and IV) design new MAC control packets which provide the status of each user.

6.1 Implementation on an Atheros chipset

Here, we describe a simple WiFi implementation of the suggested scheme using off the shelf 802.11 devices. Specifically, we implement the scheme on cards with an Atheros chipset (ATHEROS AR5007G chipset), operating the open source ath5k drivers (2.6.32 version) for a Linux environment (kernel 2.6.32). We realize the suggested scheme in a simple network comprising two stations and an AP. Due to some hardware limitations we implement a slightly modified version of the suggested scheme which we describe below. We distinguish between the enhancements required by the AP (transmitter), required by the stations (receivers) and those necessitated by the standard.

6.1.1 Frame format

In order for users to keep track of the packets received intended for other users, the header of each uncoded frame includes a two byte sequential frame index, Figure 7. We utilize a special multicast address to mark all coded frames. In addition, in all coded frames, besides the two byte sequential frame index which is included in the header, an additional two times two bytes are included in the header, indicating the sequence numbers of the two frames that are coded, Figure 7.

6.1.2 Station

In contrast to 802.11, and in order to support the NC procedure, a station needs to receive packets not addressed to it. Accordingly, we set each station network interface card (NIC) to work in promiscuous mode, which means that each station captures all frames sent by the AP, even if it is not its intended addressee. If a regular frame is received successfully at the addressee station, the station sends immediately an ACK, in accordance to the 802.11 standard. On the other hand if the station receives a frame which is not addressed to it, it stores it locally in a hashed buffer, as illustrated in Figure 8a. Once a coded frame is received, (recognized according to the designated multicast address) the relevant two byte header with the frame sequence number is used to locate the hashed frame. Then, the station retrieves the missing frame by XORing the XORed received frame with the hashed frame (if available). If a frame cannot be decoded from the NC retransmission (e.g., due to unavailability of both frames), no action is taken. Note that in such an event the packet is not going to be retransmitted, i.e., it is going to be dropped. Furthermore, it is important to note that even though we ran our experiment only on two receivers hence when receiving a coded packet the receiver knows that one of the XORed packets is intended for it, in the first part of the payload we also XORed the two MAC addresses of the intended receivers, such that our implementation also applies to more than two receivers.

6.1.3 AP

The first modification in the AP driver to allow coded retransmissions, is to stop its automatic retransmissions. Accordingly, we set the 802.11 retry limit to 0, i.e., no retransmission attempts. Nonetheless, in contrast to 802.11 where a frame is dropped when it reaches the retry limit, in our implementation if the AP does not receive an ACK message for a certain packet, it stores it in a pre-allocated buffer. A separate buffer is allocated to each user, Figure 8a. As soon as two "failed" frames (waiting for retransmission) addressed to each of the two stations are available, the AP codes the two frames into a single retransmission frame. Coding is obtained by using a XOR operation. Then, the XORed frame is transmitted to a predefined multicast address which differentiates coded frames from regular uncoded frames, Figure 8c.

In contrast to the algorithm presented in Section 4 the AP does not work in a Stop and Wait manner, in which it stops all transmissions to a certain station upon frame failure until the frame is either received correctly or dropped. Rather, in our implementation it stores the un-acknowledged frame in the aforementioned buffer and

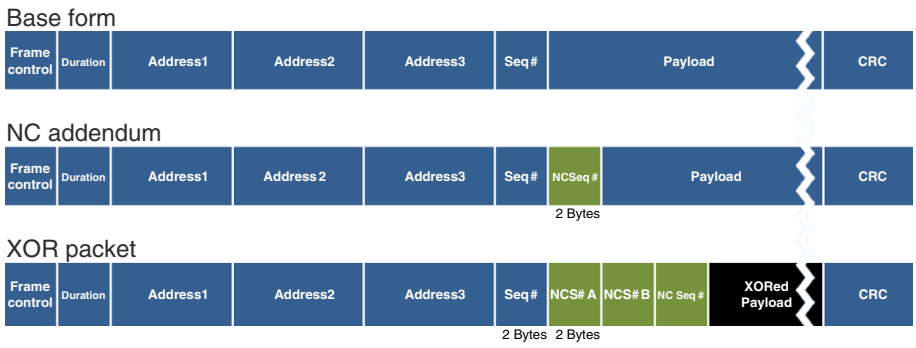


Figure 7 Regular frame, modified uncoded frame and network-code-frame structure.

continues sending subsequent frames to this user (i.e., selective repeat manner). Note that the unique sequential frame index included in each frame enables the support of such selective repeat mechanism and allows frame reordering. In order to avoid buffer overflows due to the selective repeat mechanism (which has an infinite window size), as soon as the buffer size crosses a threshold, all frames in the buffer are transmitted uncoded and the buffer is flushed. Additionally, for each un-acknowledged packet which is stored in the buffer, a time stamp is attached. A time-out mechanism is implemented such that the packet is retransmitted uncoded upon expiration of the time-out. It is important to note that in contrast to the algorithm presented earlier, no status packets are sent by the users to indicate which packets in their buffer are meant to the other user and were not acknowledged. In our implementation the AP assumes that each un-acknowledged packet is received by the other receiver, hence can be used for the coded packets. Consequently, the AP can send a coded packet that one or both receivers cannot really decode. Accordingly, as previously mentioned a retransmitted packet which cannot be decoded is lost. An enhancement in which a user periodically or upon request sends a status message which includes the unreceived packets can be easily implemented.

6.2 Implementation results

In our experiment, and in accordance to the algorithm, the AP generated packets to each one of the users at constant rate. Whenever two un-acknowledged packets were stored at the AP, one for each user, a single XORed packet was sent. In order to control the loss probability on each link, we artificially dropped packets at the receiver according to a fixed probability denoted by p. We varied the loss probability between zero (i.e., all packets are received successfully) and one (i.e., all packets are dropped). Note that coded packets were also subject to losses, according to the same probability p as uncoded packets. For comparison we also show in some of the figures analytical results of three other schemes, (i) no retransmission—each packet is transmitted exactly once, accordingly an unsuccessful attempt on the first transmission results in packet loss (denoted by dashed line in the figures). (ii) Uncoded with single retransmission—each packet can be retransmitted at most once (i.e., retry limit is set to one) where the retransmitted packets are sent uncoded (denoted by dashed dotted line). Since the uncoded retransmissions scheme allows for the same number of sent packets more overall transmissions than the coded scheme, we also examine a hybrid of the two first schemes, in which the number of transmissions (rather than the number of

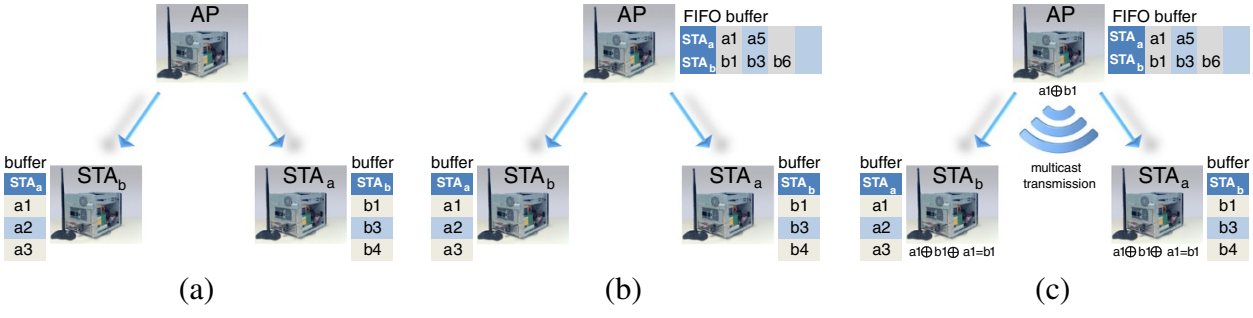


Figure 8 Implementation on an Atheros chipset. (a) System topology and settings **(b)** Access point buffer **(c)** Access point retransmission scheme.

transmitted packets) is the same. (iii) Hybrid scheme—each un-acknowledged packet is retransmitted once more with probability half or dropped with probability half (denoted by dotted line).

We first examined the effect of network coding on air time utilization. We compare the number of successfully received packets by both receivers with the total number of packets sent, i.e., $\frac{\text{total received packets}}{\text{total transmitted packets}}$. Note that system utilization for all uncoded schemes is the same as it counts the number of successfully received packets, regardless of whether or not the received packet is a retransmission. Figure 9a depicts the results.

As can be seen in Figure 9a the coded schemes (denoted by circles in the figure) are always better than the uncoded schemes (squares) as far as airtime utilization is concerned. Note that per-packet overhead is not taken into account in the figure. Nonetheless such overhead is negligible, an extra 2 Bytes for uncoded packets and an extra 12 Bytes for the coded packets. The dashed line represents the expected analytical results for the coded scheme (expressed as the fraction of successfully received packets).

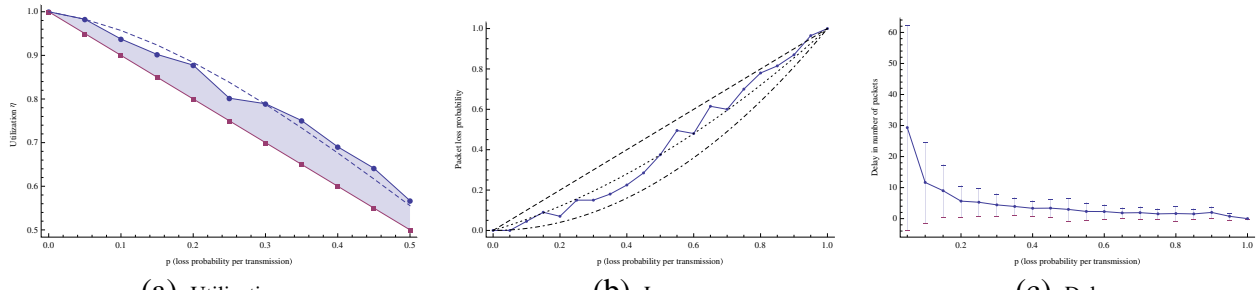
Next, we evaluate the effect of not sending status packets. Recall that in our implementation the AP does not know which packets were received by each unintended user, and assumes that each unacknowledged packet was received by the other user. Furthermore, XORed packets are not acknowledged by the receivers. Accordingly, a XORed packet which is not received or cannot be decoded (i.e., its coupled packet was not received) by the receiver is lost. In Figure 9b, we show the packet loss probability as a function of the link loss probability, p. As expected the no retransmission scheme generates the highest packet drop for all values of p. Since in the uncoded single retransmission, the retransmissions are not coded and are dedicated to the intended receiver, the least drop packets are produced. Obviously this drop packet gain comes at the price of more packets being sent altogether. Interestingly the hybrid scheme is inferior to the coded scheme with respect to packet loss probability, i.e., more packets are dropped than at the coded scheme for $0 < p < 0.5$ and is superior as far as packet loss probability is concerned for $0.5 < p < 1$. The reason is that for the coded scheme to receive a coded packet successfully, relies not only on the acceptance of the coded packet itself but also on receiving the coupled packet successfully. Accordingly the mean packet loss probability is $p(p + (1-p)p)$, where the first p relates to the original transmission loss, and the terms in the parentheses refer to the retransmission of the coded packet which can be either lost or received successfully but its coupled packet was lost. For the hybrid scheme, the mean packet loss probability is $\frac{1}{2}p + \frac{1}{2}p^2$, which indeed is greater than the coded loss probability for $p < 1/2$ and less for $p > 1/2$.

Finally, we examine the retransmission delay due to the coding mechanism. Recall that the AP waits for unacknowledged packets from both receivers before sending a retransmission. In Figure 9c, we show the average number of packets which are transmitted between the first transmission attempt and its coded retransmission.

Obviously, and as can be seen in the figure, the greater the link loss probability (p), the less the number of transmissions needed before the AP has two un-acknowledged packets, one for each receiver, hence can send a coded packet. On the other hand the less p is, the longer a retransmitted packets needs to wait before it can be coupled with another lost packet to the other receiver.

7 Conclusion

In this study, we considered the problem of multiple unicast streams from one sender to a group of receivers under a general wireless channel setting. We suggested a coded solution to the problem, which codes over packets intended to different users. Although unnecessary in the



(a) Utilization (b) Losses (c) Delay

Figure 9 Experimental results (a) Comparison of the total number of successfully received packets with the total number of packets sent. Circles—coded scheme, experimental; Squares—uncoded; Dashed—coded, theoretical. **(b)** Packet loss probability as a function of the link loss probability. Dashed—scheme (i); Dashed-dot—scheme (ii); Dotted—scheme (iii); Circles—coded, experimental. **(c)** Average number of packets transmitted between the first transmission attempt of an unsuccessful packet andits coded retransmission.

noiseless scenario, in the noisy scenario coding across different streams is beneficial in case packets intended to one receiver are overheard by others.

We suggested a MDP framework to analyze the coded and uncoded schemes, and showed how it can be exploited to find good coding strategies. Moreover, we suggested two coding schemes, which although work over a system with an exponentially large state space, do not keep track of the rewards per state (in order to solve optimality equations), yet perform significantly better than uncoded schemes. These coding schemes are based on finding large cliques in graphs, which can be solved efficiently in the graphs relevant to these coding schemes.

Finally, we depicted an architecture to implement the coded solutions within WiFi devices, in a way which is transparent to the users of the WiFi network. The basic concepts of the architecture were implemented and tested on a WiFi testbed, resulting in a coded retransmissions version of WiFi.

Endnotes

[a]Alternatively, one can associate a packet index. However, since the sender does not send a packet to a user until the previous one was decoded correctly, it suffices to index them using the user index alone.

[b]For random policies, one can define the optimization over the relevant distributions. While we use random policies in this study, solving such optimization problems will not be required.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This research was partially supported by EU FP7-FLAVIA project, contract number 257263, and by the Israeli MOITAL RESCUE consortium.

Received: 16 July 2012 Accepted: 25 January 2013
Published: 19 February 2013

References

1. R Ahlswede, N Cai, SYR Li, RW Yeung, Network information flow. IEEE Trans. Inf. Theory. **46**(4), 1204–1216 (2000)
2. S Jaggi, P Sanders, P Chou, M Effros, S Egner, K Jain, L Tolhuizen, Polynomial time algorithms for multicast network code construction. IEEE Trans. Inf. Theory. **51**(6), 1973–1982 (2005)
3. R Dougherty, K Zeger, Nonreversibility and equivalent constructions of multiple-unicast networks. IEEE Trans. Inf. Theory. **52**(11), 5067–5077 (2006)
4. G Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function. IEEE J. Sel. Areas Commun. **18**(3), 535–547 (2000)
5. M Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. (John Wiley & Son Inc., New, York, NY, 1994)
6. D Bertsekas, J Tsitsiklis, in *Proceedings of the 34th IEEE Conference on Decision and Control*, vol. 1. Neuro-dynamic programming: an overview (New Orleans, LA, 1995), pp. 560–564
7. D Bertsekas, Dynamic programming and optimal control Vol. I and II. (Athena Scientific, Belmont, MA, 1995)
8. R Koetter, M Médard, An algebraic approach to network coding. IEEE/ACM Trans. Network. **11**(5), 782–794 (2003)
9. T Ho, M Médard, R Koetter, DR Karger, M Effros, J Shi, B Leong, A random linear network coding approach to multicast. IEEE Trans. Inf. Theory. **52**(10), 4413–4430 (2006)
10. P Chou, Y Wu, K Jain, in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, vol. 41. Practical network coding (Monticello, IL, 2003), pp. 40–49
11. S Katti, D Katabi, W Hu, H Rahul, M Medard, in *Proc. 43rd Annual Allerton Conference on Communication, Control, and Computing*, vol. 2. The importance of being opportunistic: Practical network coding for wireless environments (Monticello, IL, 2005), pp. 756–765
12. S Katti, H Rahul, W Hu, D Katabi, M Médard, J Crowcroft, XORs in the air: practical wireless network coding. IEEE/ACM Trans, Network. (TON). **16**(3), 497–510 (2008)
13. S Rayanchu, S Sen, J Wu, S Banerjee, S Sengupta, Loss-aware network coding for unicast wireless sessions: Design, implementation, and performance evaluation. ACM SIGMETRICS Performance Eval. Rev. **36**, 85–96 (2008)
14. D Nguyen, T Nguyen, X Yang, in *IEEE Packet Video*. Multimedia wireless transmission with network coding (Lausanne, Switzerland, 2007), pp. 326–335
15. X Xiao, Y Lu-Ming, W Wei-Ping, Z Shuai, in *4th IEEE International Conference on Circuits and Systems for Communications*. A wireless broadcasting retransmission approach based on network coding (Shanghai, China, 2008), pp. 782–786
16. D Nguyen, T Tran, T Nguyen, B Bose, Wireless broadcast using network coding. IEEE Trans. Veh. Technol. **58**(2), 914–925 (2009)
17. S El Rouayheb, A Sprintson, C Georghiades, On the index coding problem and its relation to network coding and matroid theory. IEEE Trans. Inf. Theory. **56**(7), 3187–3195 (2010)
18. S Sorour, S Valaee, An adaptive network coded retransmission scheme for single-hop wireless multicast broadcast services. IEEE/ACM Trans. Network. (TON). **19**(3), 869–878 (2011)
19. S Sorour, S Valaee, in *IEEE Global Telecommunications Conference (GLOBECOM)*. Minimum broadcast decoding delay for generalized instantly decodable network coding (Miami, FL, 2010), pp. 1–5
20. S Sorour, S Valaee, in *IEEE International Conference on Communications (ICC)*. On minimizing broadcast completion delay for instantly decodable network coding (Cape Town, South Africa, 2010), pp. 1–5
21. S Sorour, S Valaee, in *IEEE International Symposium on Information Theory Proceedings (ISIT)*. On densifying coding opportunities in instantly decodable network coding graphs (Cambridge, MA, 2012), pp. 2456–2460
22. X Yan, R Yeung, Z Zhang, in *IEEE International Symposium on Information Theory (ISIT)*. The capacity region for multi-source multi-sink network coding (Nice, France, 2007), pp. 116–120
23. A Eryilmaz, A Ozdaglar, M Medard, in *40th Annual Conference on Information Sciences and Systems*. On delay performance gains from network coding (Princeton, NJ, 2006), pp. 864–870
24. C Wang, On the capacity of wireless 1-hop intersession network coding—a broadcast packet erasure channel approach. IEEE Trans. Inf. Theory. **58**(2), 957–988 (2012)
25. R Yeung, N Cai, Network error correction, part I: Basic concepts and upper bounds. Commun. Inf. Syst. **6**, 19–36 (2006)
26. N Cai, R Yeung, Network error correction, part II: Lower bounds. Commun. Inf. Syst. **6**, 37–54 (2006)
27. R Koetter, F Kschischang, Coding for errors and erasures in random network coding. IEEE Trans. Inf. Theory. **54**(8), 3579–3591 (2008)
28. M Siavoshani, C Fragouli, S Diggavi, in *IEEE International Symposium on Information Theory*. Noncoherent multisource network coding (Toronto, ON, 2008), pp. 817–821
29. H Shojania, B Li, in *Proceedings of the 18th international workshop on Network and operating systems support for digital audio and video*. Random network coding on the iPhone: fact or fiction? (ACM, Wiliamsburg, VA, 2009), pp. 37–42
30. FHP Fitzek, MV Pedersen, J Heide, M Médard, in *Proceedings of the 5th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. Network coding: applications and implementations on mobile devices (ACM, Bodrum, Turkey, 2010), pp. 83–87
31. S Chieochan, E Hossain, Network coding for unicast in a WiFi hotspot: promises, challenges, and testbed implementation. Comput. Netw: Int J Comput Telecommunications Netw. **56**(12), 2963–2980 (2012)

32. SYR Li, RW Yeung, N Cai, Linear network coding. IEEE Trans. Inf. Theory. **49**(2), 371–381 (2003)
33. R Karp, in *Proceedings of a symposium on the Complexity of Computer Computations*. Reducibility among Combinatorial Problems (The IBM Research Symposia Series Plenum Press 1972, Yorktown Heights, NY, 1972), pp. 85-103
34. J Moon, L Moser, On cliques in graphs. Israel J. Math. **3**, 23–28 (1965)
35. G Grimmett, C McDiarmid, in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 77. On colouring random graphs (Cambridge Univ Press, 1975), pp. 313–324
36. B Bollobás, P Erdös, in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 80. Cliques in random graphs (Cambridge Univ Press, 1976), pp. 419–427
37. R Knopp, P Humblet, in *IEEE International Conference on Communications, Seattle*, vol. 1. Information capacity and power control in single-cell multiuser communications (Seattle, WA, 1995), pp. 331–335
38. E Sondik, The optimal control of partially observable Markov processes over the infinite horizon: discounted costs. Operat. Res. **26**(2), 282–304 (1978)
39. LP Kaelbling, ML Littman, AR Cassandra, Planning and acting in partially observable stochastic domains. Artif. Intell. **101**(1–2), 99–134 (1998)
40. A Humyn, Maximal Cliques (2010). [http://www.mathworks.com/matlabcentral/fileexchange/19889]

doi:10.1186/1687-6180-2013-25
Cite this article as: Cohen *et al.*: **Coded unicast downstream traffic in a wireless network: analysis and WiFi implementation.** *EURASIP Journal on Advances in Signal Processing* 2013 **2013**:25.

Submit your manuscript to a SpringerOpen journal and benefit from:

► Convenient online submission
► Rigorous peer review
► Immediate publication on acceptance
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com