Computer Networks 105 (2016) 138–149

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Network coding for hop-by-hop communication enhancement in multi-hop networks

Peyman Pahlevani[a,*], Hana Khamfroush[b], Daniel E. Lucani[c], Morten V. Pedersen[c], Frank H.P. Fitzek[d]

[a] Institute of Advanced Studies in Basic Science, Iran
[b] Computer science department of Penn State University, United States
[c] Department of Electronic Systems, Aalborg University, Denmark
[d] TU Dresden, Germany

ARTICLE INFO

Article history:
Received 4 December 2015
Accepted 14 May 2016
Available online 26 May 2016

Keywords:
Wireless communication
Network coding
Routing protocols

ABSTRACT

In our recent study, we introduced the PlayNCool protocol that increases the throughput of the wireless networks by enabling a helper node to strengthen the communication link between two neighboring nodes and using random linear network coding. This paper focuses on design and implementation advantages of the PlayNCool protocol in a real environment of wireless mesh networks. We provide a detailed protocol to implement PlayNCool that is independent from the other protocols in the current computer network stack. PlayNCool performance is evaluated using NS-3 simulations and real-life measurements using Aalborg University's Raspberry Pi test-bed. Our results show that selecting the best policy to activate the helper node is a key to guarantee the performance of PlayNCool protocol. We also study the effect of neighbor nodes in the performance of PlayNCool. Using a helper in presence of active neighbors is useful even if the channel from helper to destination is not better than the channel between sender and destination. PlayNCool increases the gain of end-to-end communication by two-fold or more while maintaining compatibility to standard wireless ad-hoc routing protocols.

1. Introduction

Traditional routing protocols use a single path from a source node to a destination node in wireless mesh networks. The routing protocol finds the next hop to route a packet towards its destination based on different metrics, such as the number of hops and the round-trip delay. This approach is similar to the routing protocols in wired networks and it fails to exploit the broadcast nature of the wireless channel. The nodes, which are using the wireless medium, are able to overhear transmitted packets to/from other nodes. The overhearing of a packet provides interesting capabilities to nodes in the network, e.g., allowing them to forward the received packets opportunistically. Recently, Opportunistic Routing protocol (OR) exploits broadcast nature of the wireless channel to increase throughput of the communication between source and destination. In this approach, the source broadcasts a packet and all the neighboring nodes may overhear the transmitted packet and forward it to the destination. ExOR provides an implementation of an opportunistic routing protocol [1], with the caveat that nodes in the network needed to coordinate their actions to avoid redundant transmissions. To address this problem, MORE [2], CCACK [3], and GeoCode [4] exploit random linear network coding (RLNC) to decrease the amount of coordination needed between nodes. Using RLNC approaches, each intermediate node transmits coded packets. Since each coded packet is generated by making a linear combination of the received packets using randomly drawn coding coefficients from a finite field, the probability of conveying redundant data is significantly reduced. Although these approaches increase the network performance by introducing novel network coding based routing protocols, they are not exploiting the existing routing protocols already deployed in wireless networks, a fact that may hinder their use in real systems. The goal of our work is inherently different: we advocate for the use of PlayNCool, a network coded protocol that is independent of the system's routing protocol, thus allowing us to exploit existing routing protocols such as AODV [5], OLSR [6], and B.A.T.M.A.N [7] to select the best next hop. PlayNCool aims to exploit a helper node per link in the communication route, particularly when the quality of the link is poor. The helper node is a node not included in the communication path

* Corresponding author.
E-mail addresses: pahlevani@iasbs.ac.ir (P. Pahlevani), hkham@cse.psu.edu (H. Khamfroush), del@es.aau.dk (D.E. Lucani), mvp@es.aau.dk (M.V. Pedersen), frank.fitzek@tu-dresden.de (F.H.P. Fitzek).

http://dx.doi.org/10.1016/j.comnet.2016.05.012
1389-1286/

between a source and a destination, but it can improve the quality of a particular link by re-coding packets using RLNC and transmitting them to the destination of that particular link. This results in an increased reliability and throughput per link in the path and improving the end-to-end performance of the system.

There could be different reasons for node to help a communication between two nodes. For example, by strengthening of a communication for a weak link, the helper node can get rid of the retransmissions of the source node, and then transmits its own data with lower delay.

The key challenge is to maximize the effectiveness of the helper node, i.e., increase the probability of transmitting coded packets that are linearly independent of coded packets already at the destination. For this purpose, each helper *plays it cool* and avoids transmitting until it has gathered enough coded packets from the source. PlayNCool calculates the waiting time by considering the channel conditions as well as competition between nodes to access the channel. PlayNCool can use the channel condition information from the underlying routing protocol when available, e.g. B.A.T.M.A.N., or by exploiting PlayNCool's link quality discovery functionality otherwise, which uses feedback packets in PlayNCool to estimate packet losses. Our contributions are as follows.

- *Mathematical analysis and optimal solution:* We formulate the problem of finding the optimal time to enable a helper in a unicast scenario as a Markov Decision Process (MDP). The model assumes a link between two nodes in the presence of active users and a fair channel allocation per user. The MDP model allows us to determine an optimal solution for the problem. Having more active users will decrease the probability of accessing the channel for the current flow. A striking fact is that the presence of active users can significantly increase the gains of using a helper compared to the use of a direct link.
- *Heuristics based on local information and optimization:* We design PlayNCool, a heuristic to select and exploit neighbors to act as helpers for a specific link in a communication path computed by an underlying routing algorithm. PlayNCool considers only local information of channel quality and active neighbors for deciding when and for how long to allow the helper to generate RLNC packets and transmit. We show that PlayNCool's performance is near to the optimal MDP solution.
- *Implementation and simulation on NS–3:* We implement PlayNCool in NS–3 and tested its performance on deployments of up to 25 nodes. These results show a performance improvement of two to four fold compared to traditional routing in some scenarios using static routing protocol.
- *Measurements on testbed with Raspberry Pi devices:* We implement PlayNCool on Aalborg University's Raspberry Pi testbed and measured its performance in deployments across University buildings. These results confirm that significant throughput gains of up to two fold are achievable in practice. They also show that PlayNCool is particularly effective at maintaining a low number of linearly dependent packets transmitted by the helper node with minimal coordination and signaling.

The remainder of the paper is organized as follows. First, we describe the related work in Section 2. In Section 3, we illustrate our system model. Then, in Section 4 we describe the motivation and need for the PlayNCool protocol. In Section 5, an MDP model determines the optimal policy for using a helper node. Section 6 provides a design of the PlayNCool protocol to implement and evaluate it in wireless systems and Section 7 describes the numerical and measurement results of PlayNCool. Finally, Section 8 concludes the paper.

2. Related work

ExOR protocol [1] was the first protocol to exploit the broadcast nature of the wireless channel. In this protocol, the source broadcasts a packet and neighbors of the source receive it. The nodes run a protocol to find out which neighbors have received the packet. The closest node to the destination broadcasts the packet. Each node must coordinate the transmission with the other neighbors to avoid redundant transmissions. In MMOCR [8], the protocol relies on opportunistic forwarding on a channel with the least interference. Each node uses the Cumulative Interference Strength (CIS) as a metric to quantify different channel conditions.

Ahlswede et al. [9] introduced network coding to improve the performance of the networks by transmitting the combination of the packets. COPE [10] is the first network coding based unicast routing protocol where the relay node XORs the packets from different flows. FENC [11] introduced a new algorithm to reduce the complexity of COPE based approaches. MORE [2] is the first implementation of random linear network coding (RLNC) to decrease the amount of coordination needed between nodes. Each node calculates the transmission credit based on off-line calculation of the link quality in the network. Zhang et al. [12] introduced Optimized Multipath Network Coding (OMNC), which utilized MORE to work in a distributed fashion by assigning each node a broadcast rate in a distributed way. OMNC exploited the broadcast property of the wireless medium using network coding to adapt to lossy environment. CCACK [3] exploited a novel Cumulative Coded ACKnowledgment scheme that allows nodes to acknowledge in a network coded fashion to their upstream nodes in a simple way, resilience to loss, and with zero overhead. GeoCode [4] created multiple paths by choosing the nodes that are located inside a specified geographic area (e.g. ellipse) as relay nodes. The created paths may intersect each other at intermediate nodes, which use network coding to maximize the throughput. In [13], authors proposed a dynamic segmented network coding scheme to apply network coding into the Delay/Disruption Tolerant Network (DTN). SlideOR [14] is another MORE-based reliable multicast protocol that uses both intra-session and inter-session network coding to increase the performance of the networks. Khamfroush et. al investigated the optimal use of Network Coding for a multicast scenario that allows for cooperation between destinations to reduce the cost of multicast packet transmission [15].

3. System model

In traditional routing protocols, we consider a network that consists of source, destination, and a number of relays. Let us include an additional set of nodes, called helpers. The helpers are not originally part of the communication path, but can be chosen locally from the neighbor nodes to improve the throughput of specific communication links.

Fig. 1b illustrates a basic topology that a relay R_i transmits the coded packets to the next relay R_{i+1}, using a helper H_{i+1}. The helper H_{i+1} is chosen among neighbors where X represents all the number of neighbors and N_k represents an individual neighbor node between two relay nodes. When a helper is selected, the number of neighbor would be $X-1$. The packet loss probabilities between R_i and H_{i+1}, H_{i+1} and R_{i+1}, and R_i and R_{i+1} are represented by ϵ_1, ϵ_2, and ϵ_3 , respectively.

Fig. 1a illustrates a communication path from a source node to a destination node using multiple relay and helper nodes. R_0 represents the source node and R_n represents the destination node for a route with n-hop relays. The source, relays, and helpers transmit linear combinations of the packets of their buffer using RLNC. R_0 generates coded packets by linear combinations of generations of g packets using coefficients drawn uniformly at random from the el-

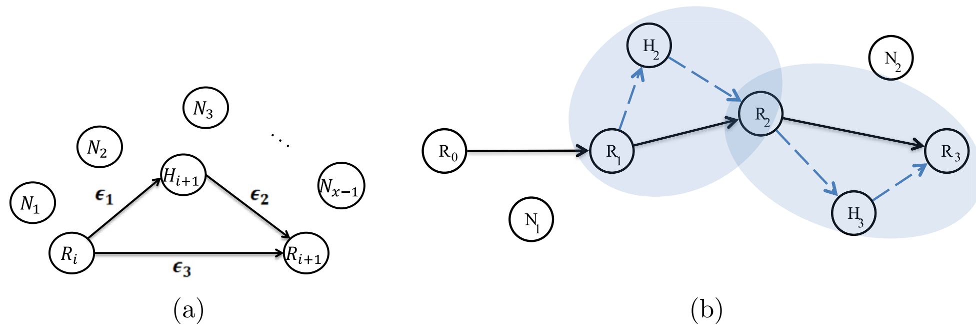
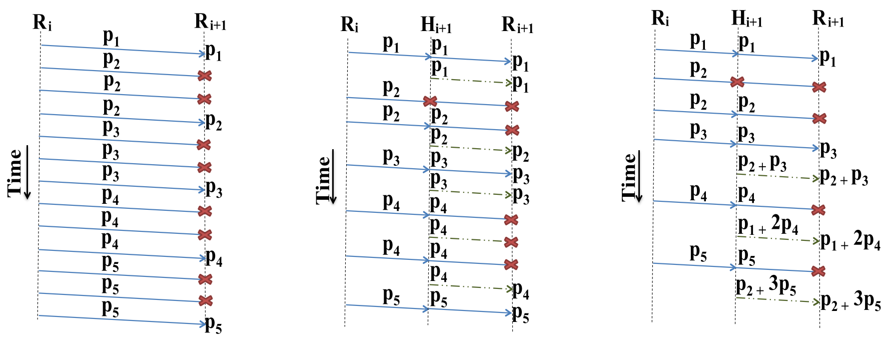


Fig. 1. (a) PlayNCool basic topology (b) PlayNCool approach. Grey areas illustrate local optimization with only one helper for a link.



(a) R_i transmits 13 packets. (b) R_i transmits 7 packets and H_{i+1} transmits 5 packets. (c) R_i transmits 5 packets and H_{i+1} transmits 3 packets.

Fig. 2. (a) R_i uses only direct link to transmit packets. (b) R_i exploits a helper but the helper only repeats the received packet. (c) the helper codes the packets in a smart way.

ements of the finite field of size q, i.e., GF(q). We assumed that q is large enough so that any RLNC packet received from the R_0 is independent from previously received packets with very high probability. However, the transmissions between H_{i+1} and R_{i+1} can be linearly dependent because they may share common linear combinations. The helper H_{i+1} accumulates the coded packets by overhearing transmissions from R_i. When it has accumulated enough coded packets, it generates RLNC packets by re-coding, i.e., by creating linear combinations of the buffered coded packets, and transmits them to R_{i+1}. At this point, both R_i and H_{i+1} continue to transmit coded packets to R_{i+1} until the R_{i+1} acknowledges that it has all g Degrees of Freedom (DOF) .[1] At this point, the R_i stops transmitting of this generation and starts transmitting a new generation.

4. Motivation

Let us use an example of transmitting five packets to illustrate the potential and caveats of using a helper node and re-coding at helper to increase the throughput performance between two nodes. Fig. 2a shows that the link between R_i and R_{i+1} is weak and, as a consequence, R_i transmits 13 times to deliver 5 packets to the R_{i+1}. While maintaining the same quality in the direct link, Fig. 2b considers the use of helper H_{i+1} with a better connection to the R_{i+1}. Let us assume that H_{i+1} simply repeats each packet to R_{i+1}. By using H_{i+1}, most of the packets transmitted from H_{i+1} are received successfully in R_{i+1}. However, R_{i+1} receives a lot of duplicated packets, i.e., the transmissions of H_{i+1} are not always useful. For example, p_1 and p_3 packets are received twice at R_{i+1}. In fact, the total number of transmissions in the system remains high, 12 transmissions in the example.

[1] Degrees of freedom corresponds to the number of independent linear combinations received or available to a node in the network.

Re-coding at H_{i+1} with RLNC with this particular loss pattern and with such an active helper does not bring a reduction of the number of transmissions. In general, it would bring an improvement over pure repetitions, but can still be quite wasteful. The reason is that H_{i+1} needs to build up its knowledge by listening to R_i transmissions. If it is too eager to transmit, it will reduce the impact of each transmission. We shall illustrate this in our measurements. Reducing the activity of H_{i+1} is an option, but being too cautious would also reduce its potential throughput benefits. Given this trade-off, it is clear that to make H_{i+1} truly useful, we need to not only to allow RLNC re-coding at H_{i+1} but to have a protocol that controls when and how many coded packets to send.

Fig. 2c provides an example of H_{i+1} waiting until it accumulates a certain number of coded packets before it starts to transmit. This translates in a total of nine transmissions in the network to convey five packets to the R_{i+1}. Although we considered a helper with a good link quality to R_{i+1}, our analysis and measurements will also show that using H_{i+1} is beneficial even when this link quality is poor.

5. Optimal MDP solution to the problem

In this section, we model the problem as an MDP problem, specifically a stochastic shortest path (SSP) [16] problem. We consider a basic topology, as shown in Fig. 1a, in the presence of $X-1$ neighbors that also use the same channel to transmit data packets. We assume a fair and adaptive Time Division Multiple Access (TDMA) medium control and all nodes have the same priority for channel access. Adaptive TDMA means that the list of nodes transmitting in each round can be updated when new nodes become active. The packet transmission cost is the number of time slots that a node needs to wait until it captures the channel plus the number of time slots is used to send packets. At each time slot,

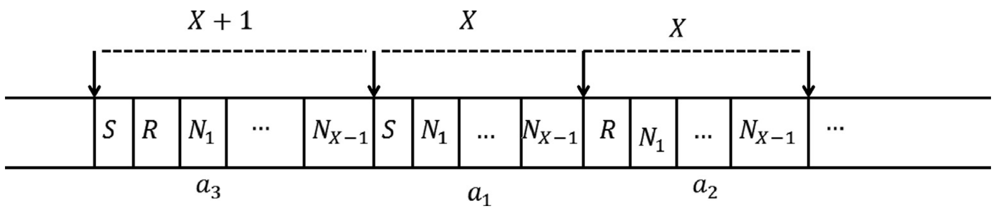


Fig. 3. Cost (required time slots) of three key actions.

the process is in a state s. By choosing an action a in the state s, the process moves to a new state s'. The process will be terminated when the R_{i+1} receives the generation. The states, possible actions, and transition probability are defined in the following [17].

5.1. State definition:

Each state is defined by a triplet $s(i_1, i_2, c)$, where i_1 is the number of DOFs at H_{i+1}, i_2 is the number of DOFs at R_{i+1}. c is the number of shared DOFs between R_{i+1} and H_{i+1}, i.e., the dimension of the common knowledge between R_{i+1} and H_{i+1}.

5.2. Possible actions:

We define actions a_1, a_2, a_3, $and a_4$ as possible ways of transmitting a packet in the network of Fig. 1a as follows. Action a_1: broadcast from R_i to R_{i+1} and H_{i+1}. Action a_2: unicast from H_{i+1} to R_{i+1}. Action a_3: first, broadcast from R_i to R_{i+1} and H_{i+1}, then unicast from H_{i+1} to R_{i+1} in two consecutive time slots. Action a_4: do not transmit.

5.3. Transition probabilities:

The possible states to which state (i_1, i_2, c) can transit to with non–zero probability depends on the action that we choose and also the total knowledge ($K = i_1 + i_2 - c$) that is available to both R_{i+1} and H_{i+1} at time t. We define $I_{x \in X}$ as an indicator function, which is one when $x \in X$ and zero otherwise. Considering the fact that the state of the network does not change when either the packet is lost or the received packet is not innovative at R_{i+1} and H_{i+1}, we can calculate the transition probability. The non–zero transition probabilities for four possible actions are summarized as follows:

Action a_1**(source broadcast):** There are different state transitions when R_i is broadcasting. We will explain the unexpected cases and the remaining cases can be studied via combinatorial arguments. If the packet is received in both R_{i+1} and H_{i+1} successfully, depending on the total knowledge, the state can transit to different states. If the total knowledge is less than g then the common knowledge between H_{i+1} and R_{i+1} will be increased by one because both H_{i+1} and R_{i+1} have received the same DOF. If the total knowledge is g the common knowledge will be increased by two. For example, assume that $g = 3$ and H_{i+1} has received $p_1, p_1 + p_2$, and R_{i+1} has received $p_2 + p_3$. The network state is $s = (2, 1, 0)$. Now, R_i broadcasts $p_1 + p_3$ and both H_{i+1} and R_{i+1} receive this packet. In this case, the common knowledge is increased by two and the system state is $s' = (3, 2, 2)$. Moreover, if the H_{i+1} has g DOFs, then any new coded packet sent by the R_i adds one DOF to the R_{i+1} and increases the common knowledge by one. This is because H_{i+1} already has all DOFs to decode the original packets and the common knowledge is equal to the number of DOFs at R_{i+1}. All possible transitions with non–zero probabilities are summarized briefly as follows:

- If $K < g,\ i_1 < g,\ i_2 < g$:
  In this case when both H_{i+1} and R_{i+1} receive the coded packet from R_i, the common knowledge will be increased by one.
- If $K = g,\ i_1 < g,\ i_2 < g$:
  In this case when both H_{i+1} and R_{i+1} receive the coded packet from R_i, the common knowledge will be increased by two.
- If $K = g,\ i_1 = g,\ i_2 \neq g$:
  In this case when R_{i+1} receives the coded packet from R_i, the common knowledge will be increased by one.
- If $i_2 = g$: $P_{(i_1, i_2, c) \to (i_1, i_2, c)} = 1$.

Action a_2 **(unicast from** H_{i+1} **to** R_{i+1}**):** If the number of DOFs at H_{i+1} is greater than the common knowledge, then the packet transmitted by H_{i+1} increases the number of DOFs by one at R_{i+1} under our high field size assumption. If the number of DOFs in the H_{i+1} is equal to the common knowledge between H_{i+1} and R_{i+1}, the received packet from H_{i+1} will increase the number of DOFs at R_{i+1}.

Action a_3 **(first broadcast, then unicast from** H_{i+1} **to** R_{i+1}**):** This action contains two successive phases, which includes a combination of a_1 and a_2 taking place at the same transmission round. First we use action a_1 to transit from state s to a new state $\hat{s}$ with probability $p_{s \to \hat{s}}$. Then, we use action a_2 to transit from $\hat{s}$ to s' with probability $p_{\hat{s} \to s'}$. Due to independent erasure channel assumption, the transition probability of moving from state s to state s' using action a_3 is calculated as $p_{s \to s'} = p_{s \to \hat{s}} \times p_{\hat{s} \to s'}$.

Action a_4**(do not transmit):** $P_{(i_1, i_2, c) \to (i_1, i_2, c)} = 1$.

5.4. Cost function

It is assumed that transmission of one packet takes one time slot for a neighbor. Because of the active neighbors, when R_i or H_{i+1} send a packet, they must wait for $X - 1$ time slots to get a new time slot to transmit their packets again. When R_i and H_{i+1} use action a_3 and both transmit in two successive time slots, then the number of time slots that is used is $X + 1$ in that transmission round. On the other hand, if only one transmits the number of slots in a round is X. Fig. 3 shows the cost of actions a_1, a_2, a_3. This leads to

$$C(s, a_j, s') = \begin{cases} X, & \forall s \in S \mid s \neq (i_1, g, c),\ j \in 1, 2 \\ (X + 1), & \forall s \in S \mid s \neq (i_1, g, c),\ j = 3 \\ D, & for\ s = (i_1, g, c),\ j \in 1, 2, 3, \\ D, & \forall s \in S \mid s \neq (i_1, g, c),\ j = 4, \\ 0, & \text{if } s = (i_1, g, c),\ \ j = 4, \end{cases} \quad (1)$$

where $C(s, a_j, s')$ is the cost of transition from state s to state s' by choosing action a_j and S is the set of all possible states. D is an arbitrary large number that is much greater than X. By defining large D, we guarantee that the MDP does not choose any one of the actions a_1, a_2, a_3 when R_{i+1} has received all DOFs which is one of the absorbing states. Instead, it chooses action a_4 that has the minimum cost. Moreover, it will not choose a_4 when the DOF of R_{i+1} is still not g. This leads to stopping the process at the absorbing states. We define a set of states of the form (i_1, g, c) as the absorbing states, where i_1 can change from zero to g.

5.5. Optimization algorithm

We can formulate the problem of cost minimization as a stochastic shortest path (SSP) problem that is a special case of an MDP, which can model decision based stochastic dynamic systems with a terminating state. The different possible situations that the system could encounter are modelled as states $s \in S_T$, where S_T denotes the state space of SSP. In each state s, the system must choose an action a_j from an action space $A(s) \subset A$ that is possible in state s that will impose an immediate cost $C(s, a_j, s')$ to the system, where A denotes the action space of the SSP problem. The cost of a transition from state s to state s' is a scalar that depends on s, the selected action a_j, and s'. In the SSP formulation, the expected cost, $\bar{C}(s, a_j)$, is calculated as $\bar{C}(s, a_j) = \sum_{s' \in S_T} P_{s \to s'}(a_j) C(s, a_j, s')$, where $P_{s \to s'}(a_j)$ represents the probability of system moving from state s to state s' once action a_j is taken. The terminating condition of the system can be thus represented as a zero-cost absorbing state s_{abs}. A policy $\pi = [\pi(s)]$ is a mapping from $S_T \to A$ that associates a given action to each of the states. The optimal policy π^* of an SSP problem is the one that minimizes the cumulative mean cost until the absorbing state is reached. The algorithms solving SSPs define a value function $V_\pi(s)$ as the expected cumulative cost until absorption, when the system starts at state s and follows policy π. It can be recursively expressed for all $s \in S_T$ as

$$V_\pi(s) = \bar{C}(s, \pi(s)) + \sum_{s' \in S(s, a_j)} P_{s \to s'}(\pi(s)) V_\pi(s'), \tag{2}$$

where $S(s, a_j)$ represents the set of possible states that system in state s can transit to with non-zero transition probabilities when action a_j is taken, i.e., $S(s, a_j) = \{s' \mid P_{s \to s'}(a_j) > 0\}$. Consequently, the optimal policy at state s can be defined as [18,19]

$$\pi^*(s) = \arg\min_{a_j \in A(s)} \left\{ \bar{C}(s, a_j) + \sum_{s' \in S(s, a_j)} P_{s \to s'}(a_j) V_{\pi^*}(s') \right\}.$$

The optimal policy of an SSP can be computed using well-known policy iteration and value iteration algorithms [20]. To solve our optimization problem and determine the optimal policy for minimizing the packet transmission cost, we assume a Genie system (GS), meaning that each node in the network has perfect knowledge of the system state. We drop this assumption for our practical schemes.

6. PlayNCool protocol

In this section, first we describe the idea behind the PlayNCool protocol. Then, we describe the PlayNCool protocol in detail by illustrating the possible actions of each node.

6.1. PlayNCool heuristic

The PlayNCool scheme uses a simple heuristic to transmit packets opportunistically. The key question is: when has the helper H_{i+1} accumulated enough coded packets? We define p as the number of overheard packets in H_{i+1} before it starts transmitting the coded packets. The helper H_{i+1} calculates the p value only from erasure probability of the link. Later, we will describe how the helper node measures the erasure probability.

The value of p should be large enough to guarantee that H_{i+1} transmissions are innovative[2] for R_{i+1} with high probability. If p is too small, H_{i+1} may transmit linearly dependent packets. If p is too large, H_{i+1} starts transmitting too late, which means R_{i+1} may have received most of the DOFs from R_i and the usefulness of H_{i+1} would be limited [21,22].

The total number of transmitted packets from R_i is split into two parts. First, the number of transmissions before H_{i+1} is activated (r) and second, the number of transmissions after H_{i+1} is activated (k). In our heuristic, we assumed that the channel is allocated to the nodes equally, which is a valid assumption for TDMA. Therefore, the number of transmissions from R_i and H_{i+1} is equal to k after the helper is activated. Hence, the total number of transmissions from R_i and H_{i+1} to R_{i+1} is as follow:

$$T_x = 2 \cdot k + r. \tag{3}$$

By considering the error probability between R_i and H_{i+1}, p is given as:

$$p = (1 - \epsilon_1) \cdot r. \tag{4}$$

Based on high field size assumption, we assume that the relay should receive g coded packets in total from H_{i+1} and R_i to decode a generation. Thus,

$$g = r \cdot (1 - \epsilon_3) + k \cdot (1 - \epsilon_2) + k \cdot (1 - \epsilon_3). \tag{5}$$

We divide r into two cases. In the first case, which is called r_a, the rate of incoming innovative packets to H_{i+1} is higher than the rate of outgoing packets from H_{i+1}, i.e., $(1 - \epsilon_1) \cdot \epsilon_3 > 1 - \epsilon_2$. Therefore, H_{i+1} starts transmitting when it has received the first innovative coded packet. The number of transmissions until H_{i+1} receives an innovative packet is $r_a = \frac{1}{(1-\epsilon_1)\cdot\epsilon_3}$, i.e., $p = 1/\epsilon_3$.

In the second case, which is called r_b, we have $(1 - \epsilon_1) \cdot \epsilon_3 \leq 1 - \epsilon_2$. The number of received DOF in H_{i+1} should be at least equal to the number of transmitted DOF from H_{i+1}. Therefore,

$$r_b \cdot (1 - \epsilon_1) \cdot \epsilon_3 + k \cdot (1 - \epsilon_1) \cdot \epsilon_3 = k \cdot (1 - \epsilon_2). \tag{6}$$

Combining Eq. (5) and Eq. (6), allows us to calculate
$r_b = -g \cdot A(\epsilon_1, \epsilon_2, \epsilon_3)/E(\epsilon_1, \epsilon_2, \epsilon_3)$, where $A(a, b, c) = -1 + b + c - a \cdot c$ and $E(a, b, c) = (2 - c - b) \cdot (c - a \cdot c) - (1 - c) \cdot A(a, b, c)$.

The number of coded packets that need to be transmitted on the link from R_i is

$$B_s(r) = \frac{g + (1 - \epsilon_2) \cdot r}{2 - \epsilon_3 - \epsilon_2}, \tag{7}$$

where r is r_a (r_b) for case 1 (2).

We consider the effect of $X - 1$ active nodes in the completion time of transmission. There are X active nodes before H_{i+1} starts to transmit the packets. That is, R_i is expected to transmit one packet every X slots. Due to that, $r \cdot X$ is the expected completion time to transmit r packets. On the other hand, when H_{i+1} is transmitting, R_i is also transmitting. Therefore, in each $X + 1$ slots, R_i and H_{i+1} transmits one packet. The expected completion time is $T_H(X - 1) = r \cdot X + k \cdot (X + 1)$. The gain in the presence of $X - 1$ neighbors is defined as the completion time of transmission of a generation without using helper approach ($T_{WH}(X - 1)$) divided by the completion time of helper approach ($T_{WH}(X - 1)$).

$$\text{Gain} = \frac{T_{WH}(X - 1)}{T_H(X - 1)}. \tag{8}$$

6.2. PlayNCool protocol details

In this section, we describe the PlayNCool protocol. First, we describe a packet loss estimation protocol and then we illustrate the actions of each node to transmit a generation of packets.

[2] A coded packet is considered innovative when its coefficient vector is linearly independent of the coefficient vector of the coded packets that the node has already received from that generation.

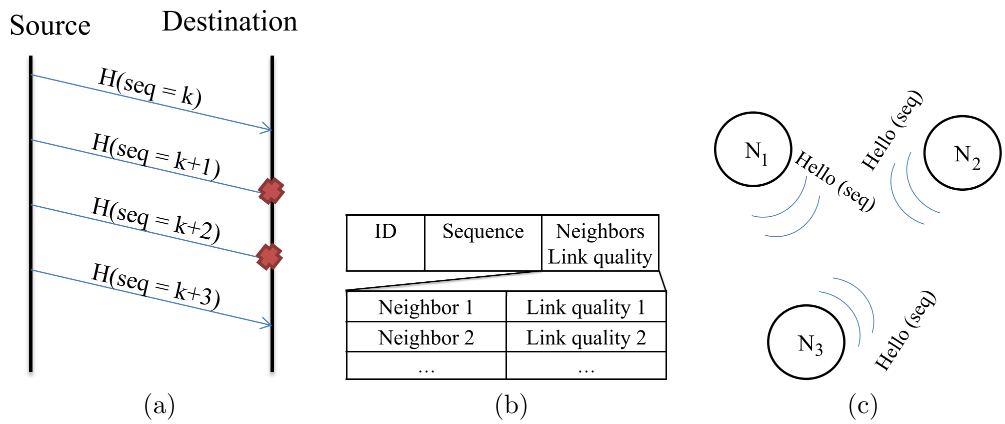


Fig. 4. (a) An example of link quality discovery protocol using sequence number in the *Hello* packets. (b) *Hello* packet format. (c) each node transmits *Hello* packets.

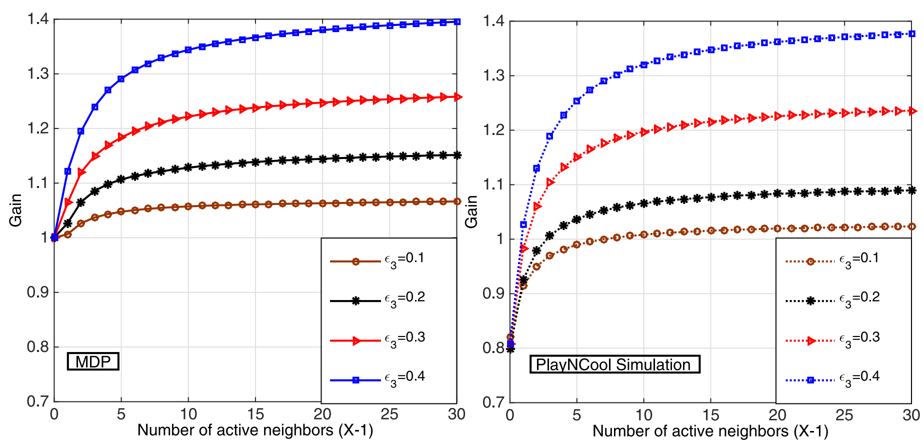


Fig. 5. Comparison between MDP, and PlayNCool simulation for $\epsilon_1 = 0.2, \epsilon_2 = 0.8, g = 10$ and different number of active neighbors.

6.2.1. Packet loss estimation protocol

To be able to calculate p and r values, H_{i+1} and R_i only need to estimate ϵ_1, ϵ_2, and ϵ_3 packet loss probabilities. Due to that, each node broadcasts a *Hello* packet to its neighbors periodically. This message includes an incremental sequence number, as shown in Fig. 4c. When a neighbor receives a *Hello* packet, it buffers the new received sequence number and then it compares the new sequence number with the last received sequence number and updates the number of lost packets as follows as

$$L(n) = L(n-1) + S_n - S_{n-1}, \tag{9}$$

which, $L(n)$ is the number of lost packets when the node receives n *Hello* packets from a neighbor, S_n is the sequence number in the message, and S_{n-1} is the last received sequence number. We define $L(1) = 0$. Each node calculates the error probability as follows:

$$e = \frac{L(n)}{S_n - S_1 + 1}, \tag{10}$$

where S_1 is the first received sequence number from a neighbor.

Each node updates the information about neighbors link quality by receiving a new packet. As an example in Fig. 4a, $S_n = k + 3$, $S_{n-1} = k$, therefore $L(n) = 2$ and $\epsilon = 0.5$. By using the Eq. 10, each node is able estimates the link quality from a neighbor. However, to estimate the link quality to a neighbor, all the neighbors should exchange the link quality. Due to that, each *Hello* packet includes estimated link quality as shown in Fig. 4b. ID header in the message format is a unique identifier for each node.

6.2.2. Relay actions

In this section we describe the sequence of the activities in R_i when the relay receives a new packet from a new generation.

Finding the best helper among the entire neighbors: The relay R_i receives the packets either from the previous relay R_{i-1} or from the network layer (when the relay is R_0). By receiving the first packet, R_i chooses H_{i+1}, which provides the most gain using Eq. (8), among the all neighbors.

Transmitting a request packet to $\mathbf{H_{i+1}}$: Once H_{i+1} is chosen, R_i transmits a request packet to H_{i+1} and activates it. The request packet activates the node to be a helper for the incoming generation. It includes information of the error probability of ϵ_1, ϵ_2, ϵ_3, generation number, and generation size.

Estimating the number of coded packets for transmission: By using the Eq. (7), R_i calculates the number of coded packets that needs to be transmitted to R_{i+1} using H_{i+1}.

Generating coded packets and controlling the transmission rate: Then R_i starts generating the coded packets using the RLNC approach and stores them in the MAC queue. The MAC layer removes the coded packets and transmits them over the wireless channel. Once the coded packets are stored in the MAC queue for

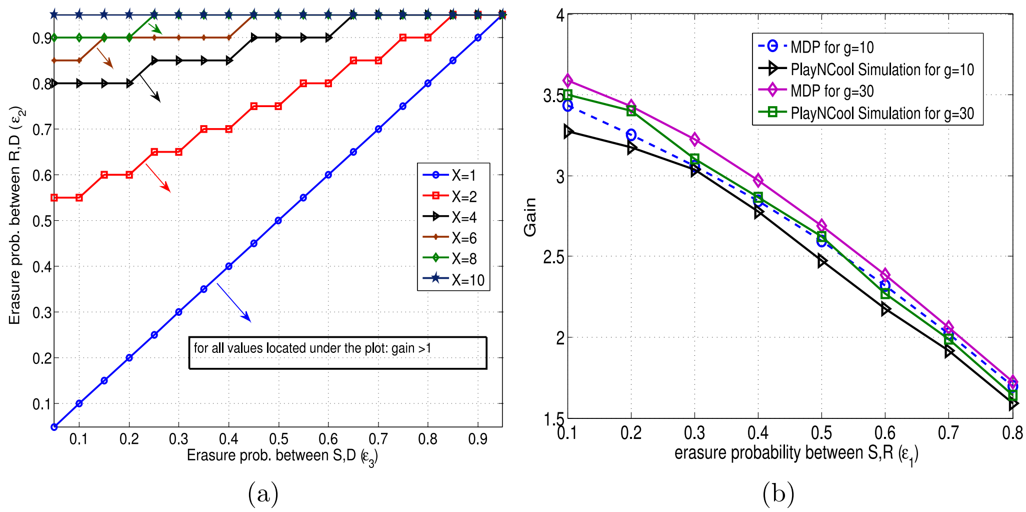


Fig. 6. (a) The map of possible area of getting benefit from using relay for $\epsilon_1 = 0.2, g = 10$ and different values of ϵ_3, ϵ_2, X: pairs of (ϵ_3, ϵ_2) under the curve of X provide gain $>$ 1, i.e., there is a gain of using the relay. (b) Gains of MDP and PlayNCool simulation for $\epsilon_3 = 0.8, \epsilon_2 = 0.3$, $X - 1 = 5$, and different values of ϵ_1 and g.

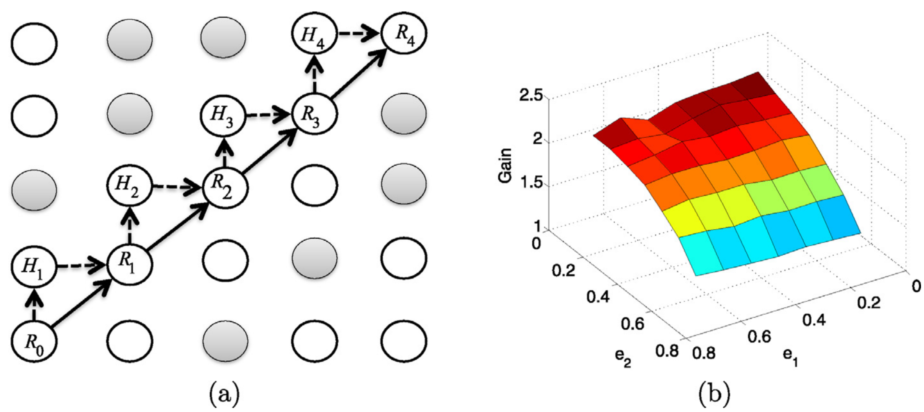


Fig. 7. (a) The meshed topology including 25 nodes. The grey nodes are generation load in the network. R_0 sends 12 generation to destination using PlayNCool. $g = 50$, $\epsilon_3 = 0.8$. (b) Gain of PlayNCool approach for different link quality in 5 × 5 mesh network. $\epsilon_3 = 0.8$, $g = 50$, and 8 nodes generates extra packets with 100KBps rate.

transmission, PlayNCool cannot remove them because the PlayNCool protocol is independent from the upper and the lower layers. R_i should control the generation of packets and it should not generate more than it knows, e.g., it should not generate 100 coded packets if it has only received two coded packets. R_i controls coded packets generation by a metric called *Budget*. When R_i receives one innovative packet, based on the link quality to the next hop, it increases the budget $B_r(t+1)$. The budget of $B_r(t+1)$ at time $t+1$ is

$$B_r(t+1) = B_r(t) + C_r^{(i+1)} - Y_r^{(i+1)}(t), \tag{11}$$

where the credit $C_r^{(i+1)}$ is the number of coded packet that needs to be generated in R_i by receiving a new coded packet and $Y_r^{(i+1)}(t)$ is the number of transmitted coded packets at time t.

When R_i sends a coded packet, H_{i+1} and R_{i+1} both may receive this new coded packet. Hence, the number of coded packets that needs to be transmitted from R_i until the R_{i+1} and H_{i+1} receive the coded packet is

$$C_r^{(i+1)} = (1 - \epsilon_3 \cdot \epsilon_1)^{-1}. \tag{12}$$

R_i generates coded packets when the budget is higher than one. When the budget is zero, it stops generating until the budget is increased.

Adding extra budget: When R_i has received all coded packets of a generation and transmission budget is zero, it adds extra budget and transmits more packets to R_{i+1}.

Transmitting an ACK packet to the previous relay: When R_{i+1} has received enough coded packets to decode a generation, it transmits an acknowledgment to R_i to stop receiving more coded packet of that generation.

Stop generating coded packets: R_i stops transmitting coded packets belonging to that generation.

6.2.3. Helper actions

In the following we explain the helper activities in details:

Receive a request packet from R_i: H_{i+1} starts receiving the coded packets from R_i when it has received a request packet from R_i. The request packet includes information about the error probability on the link and the generation ID.

Transmit a response packet to R_i: H_{i+1} transmits a response packet to R_i for each request packet. The response packet confirms that H_{i+1} is ready to transmit coded packets to R_{i+1} .

Estimating the number of coded packets that need to be transmitted: H_{i+1} calculates the number of coded packets that need to be transmitted from Eq. (5) as

$$k(r) = \frac{g - r \cdot (1 - \epsilon_3)}{2 - \epsilon_3 - \epsilon_2}, \tag{13}$$

where r is calculated for different cases (r_a and r_b).

Accumulates enough number of packets: H_{i+1} accumulates coded packets from R_i until it has received p coded packets. Then, H_{i+1} starts transmitting the coded packets to R_{i+1} meanwhile it is receiving the coded packet from R_i.

Increasing the budget by overhearing the source packets: Similar to R_i action, H_{i+1} should not generate more than it knows. By using budget $B_h(t+1)$, H_{i+1} controls the number of transmission of the packets. The budget of $B_r(t+1)$ at time $t+1$ is

$$B_r(t+1) = B_r(t) + C_r^{(i+1)} - Y_r^{(i+1)}(t), \tag{14}$$

the credit $C_r^{(i+1)}$ is the number coded packets needs to be generated in H_{i+1} per receiving a new innovative coded packet and $Y_r^{(i+1)}(t)$ is the number of packets generated by H_{i+1} at time t. As shown in Fig. 1a, when R_i transmits t coded packets, H_{i+1} receives $y = t \cdot (1-\epsilon_1)$ coded packets in the expectation. Consequently, when H_{i+1} receives one coded packet ($y = 1$) from R_i it increases its budget by a credit value equal to:

$$C_h^{(i+1)} = (1-\epsilon_1)^{-1}. \tag{15}$$

H_{i+1} generates coded packets when the budget is higher than zero.

Terminating the transmissions of the coded packets: R_{i+1} transmits an acknowledgement when it has received enough DOF to decode the generation. H_{i+1} finishes generating of coded packets when it receives the acknowledgement.

Generating additional coded packets: When H_{i+1} has transmitted the budget and it didn't receive any acknowledgement yet, H_{i+1} adds extra budget to generate more packets.

7. Performance evaluation

In this section, we show the numerical result of PlayNCool and MDP solution. First, we compare the PlayNCool heuristic with the MDP solution and evaluate PlayNCool's performance compared to the optimal approach. Then, we present the result of the PlayNCool protocol in the NS–3 simulator platform and, finally, we demonstrate the PlayNCool benefits by presenting its performance in a real implementation on Raspberry Pi devices. In this section, whenever we say source and destination, it means R_0 and R_n.

7.1. MDP and PlayNCool comparison

The C++ KODO library [23] is used to simulate the PlayNCool protocol and compare it with the optimal MDP solution. In our simulation, each node uses a fair TDMA to access the channel. The losses in the channel are synthetic and they are generated with a random variable having a Bernoulli distribution. The topology in this test includes R_0, R_1, H_1, and $X-1$ neighbors, as a consequence, X is total number of active neighbors together with H_1. In order to study the effect of different parameters of the network on the gain, we consider two scenarios: a) g and $X-1$ are fixed while ϵ_i is varied, b) ϵ_i and g are fixed while $X-1$ is varied.

In Fig. 5 we consider the case where $\epsilon_2 > \epsilon_3$, which was shown in [24] to require no helper to achieve optimal performance. As shown in Fig. 5 the gain of using helper can be larger than one if there are active neighbors in the system. By using even a small X, the gain of the helper solution is significant. This figure shows that when X is low, PlayNCool does not provide a good estimation of the gain until there is a large number of active nodes. However, when the X is large enough, PlayNCool performance and MDP performance are close. Fig. 5 demonstrates that by using a helper node, even a weak link between H_1 and R_1 decreases the completion time by around 40%.

In order to understand the effect of the active nodes in the efficiency of a helper, we illustrate the operating region where the helper provides benefits. This efficient operating region for the erasure probabilities of the links between R_0, R_1 (ϵ_3) and H_1, $R_1(\epsilon_2)$ is defined for each X value as the area under the curve (pointed by an arrow) in Fig. 6a. In other words, for different X, the helper provides gains for each pairs of (ϵ_3, ϵ_2) that are located under the curve. When there is no active neighbor ($X = 1$), Fig. 6a confirms result in [24] if $\epsilon_3 < \epsilon_2$ gain is less than one. Increasing the number of active neighbors leads to the wider efficient region of using a helper in such a way that even a single active neighbor, i.e., $X = 2$, expand in the efficient region significantly. For $X = 10$, essentially any values of (ϵ_3, ϵ_2) benefits from using H_1, as shown in Fig. 6a. In other words, the existence of active nodes makes the helper useful in a wider range of channel conditions.

We also consider the case where $\epsilon_2 < \epsilon_3$, $\epsilon_3 = 0.8$, $\epsilon_2 = 0.3$, and $X-1 = 5$. Fig. 6b shows the gain when ϵ_1 is changing and for both $g = 10$ and $g = 30$ packets. As shown in this figure by increasing ϵ_1 the gain is decreasing but it is always bigger than one. This means that regardless of the link quality between R_0 and H_1, the relay can benefit from H_1 to decrease the completion time. Also, Fig. 6b illustrates that by increasing the value of g, the difference between the gain calculated by the MDP and the simulation is decreased. This is because PlayNCool assumes that $H_!$ is always transmitting innovative packets to R_1. However, this is not always true as we have shown in the MDP analysis. By increasing the generation size, the probability of transmitting an innovative packets increases. Due to that, the gain of PlayNCool is closer to the MDP solution in this case.

7.2. NS–3 simulation

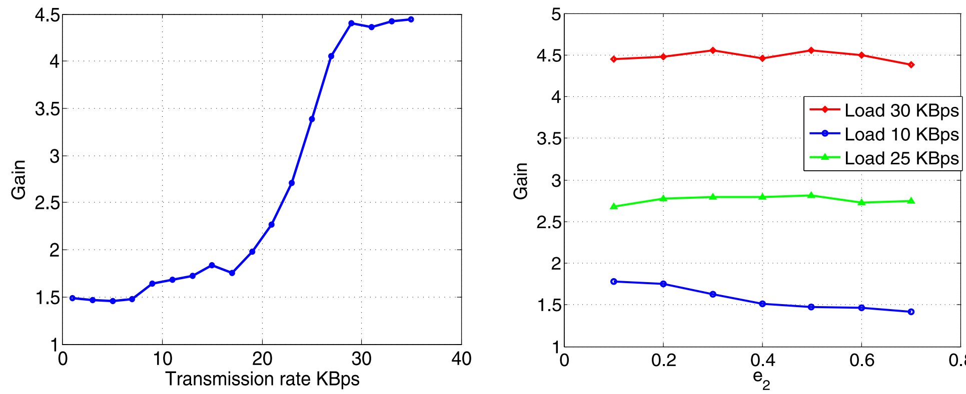
In our NS–3 simulation, we considered 5 relays (R_0, ..., R_4). R_0 sends a UDP flow to the R_4 using static routing protocol in IP layer. Moreover, the IEEE 802.11b [25] standard is used to transmit and access the channel. R_i used broadcast mode to transmit the packets to R_{i+1}. We applied the Random Propagation Delay Model and the Nakagami Propagation Loss Model. The PlayNCool layer is inserted between the MAC layer and IP layer in the NS–3 protocol stack.

In order to demonstrate the performance of the PlayNCool protocol, we evaluated the PlayNCool protocol in the wireless mesh network, including 25 nodes as shown in Fig. 7a. In this implementation, for simplicity, we have chosen a predefined helper between R_i and R_{i+1}. Each helper is in the range of the of R_i and it can transmit the packets to R_{i+1}. The active neighbor nodes, shown in gray color, are generating extra traffic to increase the competition between nodes to capture the channel. R_0 transmits 12 generations to the R_4 through 3 relays and 4 helpers. We define 50 packets in each generation ($g = 50$) and $\epsilon_3 = 0.8$ in this topology.

Results

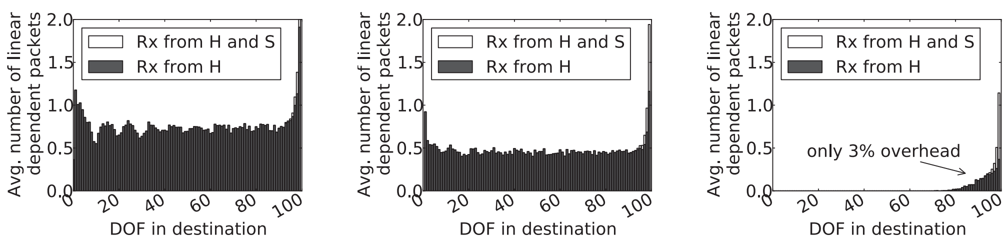
As shown in Fig. 7b, the maximum gain is 2.3 and the gain is mostly determined by ϵ_2. However, even when ϵ_2 is weak and close to 0.6 the gain is still significant and it is close to 1.5.

The effect of the load on the PlayNCool gain: As we discussed before, the competition between nodes to capture the channel plays an important role in gain. To study the effect of that, all the nodes around the main flow are generating traffic in the network. Fig. 8a shows the result for the different transmission rates when $\epsilon_1 = 0.4$, $\epsilon_2 = 0.4$, $\epsilon_3 = 0.8$. By increasing the transmission rate, the gain of the PlayNCool protocol increases and it stabilizes when the gain is equal to 4. The reason is that, having R_i and H_{i+1} active at the same time allows them to access the channel more frequently. When the rate reaches to the highest point, the gain of the PlayNCool protocol will be stabilized because the MAC protocol shares the channel between nodes equally and the channel is fully congested. Fig. 8b shows the effect of the ϵ_2 and different transmission rates on the gain. In the case of low transmission rates, by increasing ϵ_2 the gain will be decreased. However, when the load is high enough the gained obtained from competition is dominated and the effect of the ϵ_2 on the gain is minor.

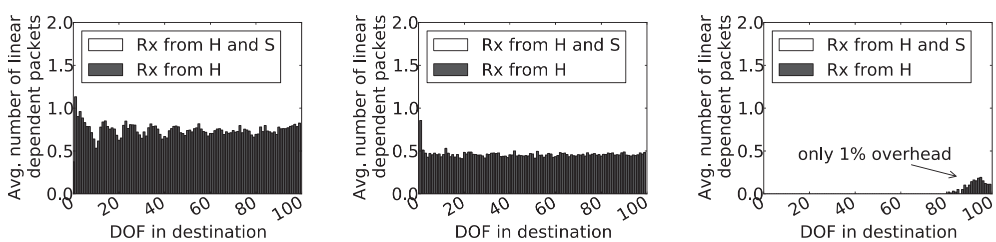


(a) (b)

Fig. 8. (a) The effect of the load on the gain in PlayNCool. $\epsilon_1 = 0.4$, $\epsilon_2 = 0.4$, $\epsilon_3 = 0.8$, $g = 50$. (b) The effect of the load and error probability of the helper and the relay (R_{i+1}) on the gain in PlayNCool. $\epsilon_1 = 0.3$, $\epsilon_3 = 0.8$, $g = 50$.



(a) Strategy:Standard GF(2) Overhead=76% (b) Strategy:recode-forward GF(2) Overhead=47%, (c) Strategy:PlayNCool GF(2) Overhead=3%



(d) Strategy:Standard GF(2^8) Overhead=73% (e) Strategy:recode-forward GF(2^8) Overhead=45% (f) Strategy:PlayNCool GF(2^8) Overhead=1%

Fig. 9. The average number of linearly dependent packets received in the destination from the source and the helper for certain DOF in the destination. The helper is using three different strategies including standard, recode-and-forward, and PlayNCool with different field sizes ($GF(2)$, $GF(2^8)$). $g = 100$, $\epsilon_1 = 0.3$, $\epsilon_2 = 0.1$, and $\epsilon_3 = 0.3$.

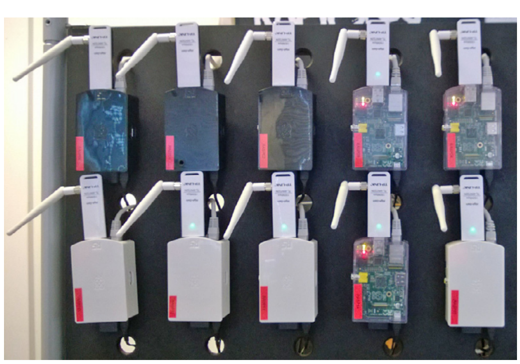
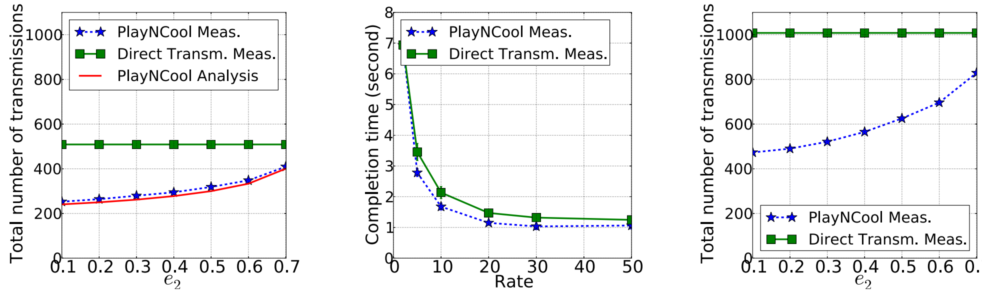


Fig. 10. The Raspberry Pi devices test-bed to take the channel quality.

7.3. Implementation on Raspberry Pi devices

In this section, we present the PlayNCool results on the Raspberry Pi devices. The Raspberry Pi devices [26] are credit-card-sized computers intended for teaching computer science at school. Raspberry Pi devices are using the Ubuntu operating system and TL-WN722N WiFi devices to transmit data in wireless channel [27]. All nodes are using IEEE 802.11 standard and 2412 MHz frequency to transmit and receive data packets in broadcast mode. The source broadcasts UDP flow to the destination. The measurement are taken in the Department of Electronic Systems at Aalborg University. PlayNCool is implemented on the application layer of the Raspberry Pi devices using broadcast sockets. Our test-bed is shown in Fig. 10. An ad hoc network including three nodes is considered as shown in Fig. 1a. The helper is located in the range of the source and it can transmit the packets to the destination. We used both synthetic and non-synthetic loss to evaluate the performance of the PlayNCool.



(a) (b) (c)

Fig. 11. (a) Total number of transmissions for basic topology. $\epsilon_3 = 0.8$, $\epsilon_1 = 0.4$. (b) completion time for different transmission rates for basic topology. $\epsilon_1 = 0.5$, $\epsilon_2 = 0.3$, $\epsilon_3 = 0.8$. (c) Total number of transmission when when the topology includes 5 nodes of R_0, R_1, R_2, H_1, H_2. R_0. $\epsilon_1 = 0.5$, $\epsilon_2 = 0.3$, $\epsilon_3 = 0.8$.

7.3.1. The importance of the PlayNCool protocol

As we discussed before, re-coding at the helper reduces the transmission of linear dependent packets from the helper. However, applying re-coding alone does not reduce the transmission of linearly dependent packets significantly and it needs a proper protocol to transmit re-coded packets in a wise way.

To understand the benefit of PlayNCool strategy, we introduce standard and Re-code-and-Forward strategies to activate the helper node. In the standard strategy, by receiving the first coded packet from the source, the helper starts re-coding and transmitting the coded packets. In this strategy, there is no control on the number of transmuted packets from the helper. On the contrary, the helper in the Recode-and-Forward strategy, re-codes and forwards a packet whenever it receives only one new packet from the source. In other words, receiving a packet from the source gives transmission credit of one packet to the helper.

In Fig. 9a, the helper is using standard strategy to transmit packets. the destination receives 76% overhead. It receives on average 176 coded packets before being able to decode the original 100. The Recode-and-Forward strategy is better than the standard strategy but the destination is still receiving 47% overhead per generation. On the contrary, the PlayNCool strategy transmits only 0.03% overhead shown in Fig. 9c. The reason is that the helper does not transmit coded packets until it has accumulated enough coded packets to transmit to the destination.

The field size has an impact on the overhead of linear dependent packets. Figs. 9a and 9 d are using $GF(2)$ and $GF(2^8)$ respectively for standard strategy. As shown in these figures, the destination receives 76% overhead when the source and the helper use GF(2) to code the packets. On the other hand, when they use $GF(2^8)$, the destination receives 73% overhead. The reason is that when the destination has received most of the DOF and the source and the helper are using GF(2) to code the packets, the probability of selecting a new DOF is low.

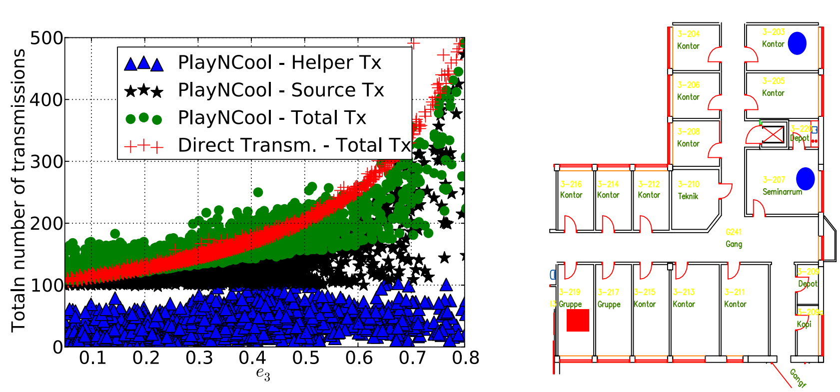
7.3.2. PlayNCool performance on Raspberry Pi devices

Our topology includes three nodes R_0, R_1, H_1. Fig. 11a compares the implementation and theoretical results of PlayNCool in terms of the total number of transmissions when $g = 100$, $\epsilon_3 = 0.8$, $\epsilon_1 = 0.4$, while ϵ_2 is varying. The result confirms that the PlayNCool heuristic estimation is close to the real implementation. Moreover, the total number of transmissions of PlayNCool is significantly less than a direct transmission from source to destination.

Fig. 11b illustrates the completion time of the PlayNCool measurement and the Direct Transmission measurement in different rates. The completion time of both approaches decreases by increasing the transmission rate and stabilizes in the high rate. PlayNCool requires less time to complete the transmission because because both helper and source are active at the same time.

Fig. 11c compares the total number of transmissions when the topology includes 5 nodes of R_0, R_1, R_2, H_1, H_2. R_0 transmits coded packets to R_1 and R_1 re-codes the coded packets and forwards them to the R_2. Both R_0 and R_1 exploit a helper (H_1, H_2) to fortify their direct link to the next hop.

As a second measurement round, we no longer introduce synthetic losses but rely only on losses introduced by the wireless channel. The measurement is taken in the building of Department of Electronic Systems at Aalborg University as in Fig. 12b. The red point is R_0 (source), located in the first floor, and blue points are R_1 and H_1, located in the second floor. We put H_1 in three



(a) (b)

Fig. 12. (a) The number of transmitted packets for different nodes when $0 \le e_2 \le 0.2$ and $0.2 \le e_1 \le 0.4$. (b) The Raspberry Pi devices deployment in a Aalborg University building.

Table 1
The correlation between helper and destination for different helper positions.

Position	Correlation
near to R_1	0.33166
in the middle	0.08937
near to R_0	0.05780

different places (close to R_0, close to $R1$, and in the middle of R_0 and R_1). Fig. 12a shows the implementation results when $0 \leq \epsilon_2 \leq 0.2$ and $0.2 \leq \epsilon_1 \leq 0.4$. When the error between R_0 and R_1 is higher than the 0.4, the total number of transmissions of the PlayNCool approach is less than the total number of transmissions of Direct Transmission. However, there are some cases that the PlayNCool approach is not efficient. The reason is that the value of loss correlation from R_0 to H_1 and R_1 has a significant impact on the gain. Having the high value of the packet loss correlation makes H_1 inefficient because H_1 receives the same DOFs as R_1 with high probability(Table 1).

8. Conclusions

In this paper, we introduced a routing independent protocol, called PlayNCool, to increase the performance of the wireless networks. PlayNCool exploits a local helper to fortify the gain of the individual link. The advantage of using a local helper is that it can use local information available in a specific link. In particular, the link quality between two relays and between relay and helper are the key factor to determine the gains provided by the PlayNCool. Besides, we showed that using a helper in the presence of active neighbors is useful even if the channel from helper to destination is not better than the channel between sender and destination. Our NS–3 simulations showed that PlayNCool increases the end-to-end gain by factor of two to four folds in the wireless mesh network. The implementation and measurements using Aalborg University's Raspberry Pi testbed proved that the PlayNCool protocol decreases the total number of transmissions and the completion time. Our future work will focus on the using more helpers and also the effect of geographical position of the helper in the gain of the PlayNCool.

Acknowledgment

This work was partially financed by the Green Mobile Cloud project granted by the Danish Council for Independent Research (Grant No. 10-081621) and Portuguese Foundation for Science and Technology (FCT) under grant SFRH/BD/72961/2010.

References

[1] S. Biswas, R. Morris, ExOR: Opportunistic multi-hop routing for wireless networks, SIGCOMM, 2005.
[2] S. Chachulski, M. Jennings, S. Katti, D. Katabi, MORE: A network coding approach to opportunistic routing, SIGCOMM, 2007.
[3] D. Koutsonikolas, C.-C. Wang, Y. Hu, CCACK: Efficient network coding based opportunistic routing through cumulative coded acknowledgments, IEEE INFOCOM, 2010.
[4] H. Khamfroush, D. Lucani, J. Barros, GeoCode: A geographic coding-aware communication protocol, in: International IEEE Conference on Intelligent Transport System (ITSC), 2011.
[5] C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1997, pp. 90–100.
[6] T. Clausen, P. Jacquet, Optimized Link State Routing Protocol (OLSR), 2003.
[7] A. Neumann, C. Aichele, M. Lindner, S. Wunderlich, Better approach to mobile ad-hoc networking, IETF Internet-Draft, 2008.
[8] W. Jingrong, W. Muqing, L. Bo, W. Dongyang, Opportunistic cooperative routing in multi-radio multi-channel wireless sensor networks, in: Communications Workshops (ICC), 2013 IEEE International Conference on, 2013, pp. 276–280, doi:10.1109/ICCW.2013.6649243.
[9] R. Ahlswede, N. Cai, S.-Y. Li, R.W. Yeung, Network information flow, IEEE Trans. Inf. Theor. 46 (4) (2000) 1204–1216, doi:10.1109/18.850663.
[10] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, J. Crowcroft, XORs in the air: practical wireless network coding, SIGCOMM Comput. Commun. Rev. 36 (4) (2006) 243Ãû254.
[11] P. Pahlavani, V. Derhami, A.M.Z. Bidoki, Fenc: Fast and efficient opportunistic network coding in wireless networks., in: TIIS, 2011, pp. 52–67.
[12] X. Zhang, B. Li, Optimized multipath network coding in lossy wireless networks, IEEE J. Selected Areas Commun. 27 (5) (2009) 622–634, doi:10.1109/JSAC.2009.090605.
[13] D. Zeng, S. Guo, J. Hu, Reliable bulk-data dissemination in delay tolerant networks, Parallel Distributed Syst., IEEE Trans. 25 (8) (2014) 2180–2189, doi:10.1109/TPDS.2013.221.
[14] P. Li, S. Guo, S. Yu, A. Vasilakos, Codepipe: An opportunistic feeding and routing protocol for reliable multicast with pipelined network coding, in: INFOCOM, 2012 Proceedings IEEE, 2012, pp. 100–108, doi:10.1109/INFCOM.2012.6195456.
[15] H. Khamfroush, D. Lucani, P. Pahlevani, J. Barros, On optimal policies for network-coded cooperation: Theory and implementation, Selected Areas Commun., IEEE J. 33 (2) (2015) 199–212, doi:10.1109/JSAC.2014.2384291.
[16] Mausam, A. Kolobov, Planning with Markov Decision Processes: An AI Perspective, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2012.
[17] H. Khamfroush, P. Pahlevani, D.E. Lucani, M. HundebÈ'll, F. Fitzek, On the coded packet relay network in the presence of neighbors: Benefits of speaking in a crowded room, in: IEEE International Conference on Communications, 2014.
[18] S. Sorour, S. Valaee, On minimizing broadcast completion delay for instantly decodable network coding, in: Communications (ICC), 2010 IEEE International Conference on, 2010, pp. 1–5, doi:10.1109/ICC.2010.5502758.
[19] N. Aboutorab, P. Sadeghi, S. Sorour, Enabling a tradeoff between completion time and decoding delay in instantly decodable network coded systems, Commun., IEEE Trans. 62 (4) (2014) 1296–1309, doi:10.1109/TCOMM.2014.021614.130172.
[20] M.L. Puterman, Markov Decision Processes—Discrete Stochastic Dynamic Programming, John Wiley & Sons, Inc., New York, NY, 1994.
[21] P. Pahlevani, D. Lucani, M.V. Pedersen, F. Fitzek, Playncool:opportunistic network coding for local optimization of routing in wireless mesh networks, in: Proceedings of Globecom Workshops, 2013.
[22] P. Pahlevani, D. Lucani, M.V. Pedersen, F. Fitzek, Network coding to enhance standard routing protocols in wireless mesh networks, in: Proceedings of Globecom Workshops, 2013.
[23] M.V. Pedersen, J. Heide, F.H. Fitzek, KODO: An open and research oriented network coding library, Workshop on Network Coding App. and Pro. (NC-Pro), Spain.
[24] X. Shi, M. Médard, D.E. Lucani, Whether and where to code in the wireless packet erasure relay channel, IEEE Journal on Selected Areas in Communications 31 (8) (2013) 1379–1389.
[25] IEEE, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, 1999, (IEEE Standard 802.11).
[26] Raspberry pi, URL http://www.raspberrypi.org/.
[27] A. Paramanathan, P. Pahlevani, S. Thorsteinsson, M. HundebÈ'll, D. Roetter, F. Fitzek, Sharing the Pi: testbed description and performance evaluation of network coding on the Raspberry Pi, IEEE Vehicular Technol. Conf. Proc. (2014).

Peyman Pahlevani is an Assistant Professor in the department of Computer Science and Information Technology of Institute for Advanced Studies in Basic Science (IASBS), Iran. He received his PhD in Wireless Communication from Aalborg University, Denmark in 2014. Prof. Pahlevani was a visiting researcher at the Computer Science Department of University of California, Los Angeles (UCLA). Moreover, He has been collaborated with different institutes and universities such as MIT, and Porto universities. Beside his academic backgrounds, he extended his skills to solve practical challenges in the area of video communication in his previous carrier, as a Researcher Engineer at AIRTAME company. His research interests are in the area of wireless communication, network coding and its applications, vehicular communications, cooperative networking, and video streaming over WiFi links. Dr. Pahlevani has also served as TPC member for international conferences and as reviewer for high impact journals, such as IEEE Transactions on Vehicular Technology.

Hana Khamfroush is a Postdoctoral scholar in the department of computer science and electrical engineering of Penn State University, USA. She received her PhD in Telecommunications engineering from University of Porto, Portugal in 2014, where she was a member of the Instituto de Telecomunicações (IT) from Nov 2009 to Nov 2014. Her research interests lie in the general areas of communications and networks, network coding, interdependent networks, network recovery and network tomography. Dr. Khamfroush was a visiting researcher at the Department of Electronic Systems of Aalborg University. Dr. Khamfroush has also served as TPC member for international conferences and as reviewer for high impact journals, such as, IEEE Journal of Selected Areas in Communications, IEEE Transactions on Communications, and Elsevier Journal.

Daniel E. Lucani is an Associate Professor in the department of Electronic Systems, University of Aalborg, Denmark. He was an Assistant Professor at the Faculty of Engineering of the University of Porto and a member of the Instituto de Telecomunicações (IT) from April 2010 to July 2012 before joining Aalborg University. He received his B.S. (*summa cum laude*) and M.S. (with honors) degrees in Electronics Engineering from Universidad Simón Bolívar, Venezuela in 2005 and 2006, respectively, and the Ph.D. degree in Electrical Engineering from the Massachusetts Institute of Technology (MIT) in 2010. His research interests lie in the general areas of communications and networks, network coding, information theory and their applications to highly volatile wireless sensor networks, satellite and underwater networks, focusing on issues of robustness, reliability, delay, energy, and resource allocation. Prof. Lucani was a visiting professor at MIT. He is the general co-chair of the 2014 International Symposium on Network Coding (NetCod2014) and was the general co-chair of the Network Coding Applications and Protocols Workshop (NC-Pro 2011). Dr. Lucani has also served as TPC member for international conferences and as reviewer for high impact journals, such as, IEEE Journal of Selected Areas in Communications, IEEE Transactions on Information Theory, IEEE Transactions on Communications, IEEE Transactions on Wireless Communications, and IEEE/ACM Transactions on Networking.

Morten V. Pedersen received his PhD in 2012. He is currently working as a post doctoral researcher at the Department of Electronic Systems at Aalborg University. Where his main focus is working on low complexity network coding algorithms and cooperative networking protocols. In 2010 he received the Forum Nokia Champion award. He co-founded the start-up company Steinwurf ApS in 2011, creating industrial grade high performance content distribution systems.

Frank H. P. Fitzek is a Professor and chair of the communication networks group at Technische Universität Dresden coordinating the 5G Lab Germany. He received his diploma (Dipl.-Ing.) degree in electrical engineering from the University of Technology - Rheinisch-Westfälische Technische Hochschule (RWTH) - Aachen, Germany, in 1997 and his Ph.D. (Dr.-Ing.) in Electrical Engineering from the Technical University Berlin, Germany in 2002 and became Adjunct Professor at the University of Ferrara, Italy in the same year. In 2003 he joined Aalborg University as Associate Professor and later became Professor. He co-founded several start-up companies starting with acticom GmbH in Berlin in 1999. He has visited various research institutes including Massachusetts Institute of Technology (MIT), VTT, and Arizona State University. In 2005 he won the YRP award for the work on MIMO MDC and received the Young Elite Researcher Award of Denmark. He was selected to receive the NOKIA Champion Award several times in a row from 2007 to 2011. In 2008 he was awarded the Nokia Achievement Award for his work on cooperative networks. In 2011 he received the SAPERE AUDE research grant from the Danish government and in 2012 he received the Vodafone Innovation price. In 2015 he was awarded the honorary degree "Doctor Honoris Causa" from Budapest University of Technology and Economy (BUTE). His current research interests are in the areas of wireless and mobile 5G communication networks, mobile phone programming, network coding, cross layer as well as energy efficient protocol design and cooperative networking.