WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 0000; **00**:1–16
DOI: 10.1002/wcm

Broadcasting with Hard Deadlines in Wireless Multi-hop Networks Using Network Coding

Pouya Ostovari[1], Abdallah Khreishah[2], and Jie Wu[1]

[1]Department of Computer & Information Sciences, Temple University, Philadelphia, PA 19122
[2]Department of Electrical & Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102

ABSTRACT

Broadcasting with network coding mixes packets to minimize the number of transmissions, which improves the energy efficiency of wireless networks. On the other hand, delaying the transmissions increases coding opportunities at intermediate nodes, but increases the delay of packets. In this paper, we consider these two contradicting factors and study the problem of minimizing the number of transmissions in wireless networks while meeting the deadline constraints. We show that this problem is NP-complete; therefore, we provide a heuristic to solve it. First, we construct broadcasting trees, each of them rooted at one source. We then specify overlapping conditions based on the constructed trees, to determine the number of transmissions each node has to perform without the deadline constraints. Then, we partition the set of packets such that coding is performed among the packets of the same partition, which does not result in deadline misses. Linear coding may not be applicable in some wireless networks because of its computational complexity. For these networks, we propose three XOR coding approaches which rely only on local neighborhood information. Simulation results show that our techniques not only reduce the number of transmissions, but also allow the majority of nodes to receive the packets on time. Copyright © 0000 John Wiley & Sons, Ltd.

KEYWORDS

Broadcasting, broadcast tree, linear network coding, local XOR coding, energy efficiency, deadline, partial dominant pruning, NP-completeness.

1. INTRODUCTION

Broadcasting is used frequently in wireless networks to disseminate control messages and data in many applications. Flooding is the simplest broadcasting method in wireless networks, where each node forwards the received packets to its neighbors. Clearly, flooding is not an efficient way to broadcast, due to the unnecessary and redundant transmissions it causes. To perform broadcasting efficiently, many works have targeted decreasing the number of transmissions. These works can be classified into two main categories: probabilistic and deterministic approaches.

In the probabilistic methods, each node forwards the received packets with a given forwarding probability [1, 2, 3]. This probability should be chosen carefully, so that all nodes are able to receive the packets with the restricted number of transmissions. On the other hand, the deterministic approaches use the network topology and neighborhood information to select some forwarding nodes that are responsible for forwarding the received packets. Connected dominating sets (CDS) [4] and pruning approaches [5] belong to this category.

With network coding [6, 7], intermediate nodes mix packets using mathematical operations, which reduces the number of transmitted packets. Network coding can be combined with both the probabilistic and deterministic approaches to improve the transmission efficiency in wireless networks.

Prepared using wcmauth.cls [Version: 2010/07/01 v2.00]

Most of the works on network coding focus on maximizing the throughput [8, 9], achieving fairness [10], or minimizing the energy consumption [11, 2]. While these metrics are important, delay and deadline metrics have received less attention from the community. In this paper, we take a different look at the network coding problem by studying the problem of *energy-efficient broadcasting in wireless networks subject to deadline constraints*. Network coding reduces energy consumption, but increases the delay of packets, as the intermediate nodes need to wait until receiving all of the packets before coding. Thus, it is crucial to specify the degree of network coding, such that the packets can meet the deadlines. Network coding has been studied with deadlines in [12, 13], but all of these studies are for single-hop coding and broadcast channels.

In this paper, we consider multihop wireless networks, and we have the following contributions.

- We show that the problem of minimum energy broadcast, subject to the deadline constraints, is NP-complete.
- We use linear network coding and propose a three-phase heuristic for the problem.
- For the networks with limited computational power, such as sensor networks, we propose a local XOR network. To increase the energy efficiency of network coding, we propose three methods, Velocity-based Waiting Time (VWT), Random Waiting Time (RWT), and Proportional Distribution of Waiting time (PDWT) methods, to compute the waiting times of the packets at relay nodes, such that no packet misses its deadline. These waiting times increase the chance of coding the packets, which decreases the number of transmissions, and increases the energy efficiency of network coding.
- We conduct simulations to show the benefits of our proposed schemes in terms of meeting the deadlines and energy efficiency.

The remainder of this paper is organized as follows: In Section 2, we provide the necessary background about network coding. Section 3 contains the problem definition, settings, and motivation. In Section 4, we propose our deadline-aware energy-efficient broadcast heuristic using linear network coding. We extend the proposed method in Section 4 to increase its efficiency. In Section 6, we propose our deadline-aware local XOR coding schemes. We evaluate the proposed methods through simulations in Section 7. Section 8 concludes the paper.

2. RELATED WORK AND BACKGROUND

Network coding can be classified into local and global coding. In local network coding, each relay node decodes the received coded packets, and it mixes the native (non-coded) packets, such that its neighbors can decode the coded packet using the packets in their buffer. This means that the next hops can decode the received coded packets immediately, and they do not need to wait to receive further packets to be able to decode the coded packets. On the other hand, in global network coding, the intermediate nodes do not perform decoding; they just code the coded packets again without considering the status of their neighbors. In this approach, when a receiver node receives a coded packet, it cannot decode the packet immediately, and it has to wait to receive a sufficient number of packets to be able to decode the coded packet. Usually, local network coding protocols use XOR coding, and global protocols perform random linear coding. Linear network coding is introduced in [7], as it is shown to achieve the capacity for the single multicast session problem. A useful algebraic representation of the linear network coding problem is provided in [14].

In linear network coding, each node generates and sends a linear combination of the received packets over a finite field. In Fig. 1 (a), the relay node has three incoming links. Packets p_1, p_2, and p_3 can be any linear combination of the source packets, and the output packets are also linear coded packets over a finite field. When a node receives an innovative packet, it stores this packet in its packet buffer, and the corresponding coefficients vector in its coefficients buffer. An innovative packet is a received packet, such that its coefficient vector increases the rank of the matrix formed by the received coefficient vectors. In other words, an innovative packet is a linearly independent packet to the previously received packets. Each forwarder node continues this process. Assume that K single packets are coded together. When a destination node receives K linearly independent coded packets, it will be able to decode all of the coded packets and retrieve all of the single packets. The decoding process

Wirel. Commun. Mob. Comput. 0000; **00**:1–16
DOI: 10.1002/wcm
Prepared using wcmauth.cls

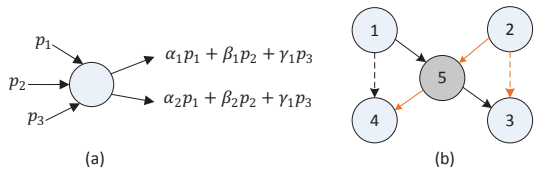


Figure 1. (a):Linear coding. (b) Local coding.

is done using Gaussian elimination for solving a system of linear equations. In [15], it is shown that selecting the coefficients, in a distributed manner at random, achieves the capacity asymptotically with respect to the finite field size.

In local network coding, each node has local information about the received packets by its neighbors. Based on this information, the relay node decides which packets should be coded together, such that all of the neighbors will be able to decode the coded packet using the packets in their buffers. Assume that in Fig. 1 (b) nodes 1 and 2 want to send their respective packets, p_1 and p_2 to nodes 3 and 4, respectively. Here, node 5 is the relay node. Using two-hop information, node 5 knows that nodes 3 and 4 can overhear nodes 2 and 1, respectively. If node 5 does not use network coding, it has to send two packets. However, this node can code packets p_1 and p_2 to send coded packet $P = p_1 \oplus p_2$, since each of the destination nodes can overhear one of the packets directly. Node 3 can recover packet p_1 by performing $P \oplus p_2$, and node 4 can recover p_2 by performing $P \oplus p_1$.

Local XOR coding does not achieve optimality compared to global linear coding, but it has some advantages over linear network coding. First, the computational complexity of coding and decoding processes in local XOR coding is much less than in global linear coding. Thus, for the nodes with limited computational power, such as sensor networks, local XOR coding is more attractive. Next, linear network coding has more overhead than local coding because of the coefficient vectors. In the rest of the paper, we refer to local XOR coding and global linear coding as local coding and linear coding, respectively.

From another perspective, network coding can be classified into intra- and inter-session network coding. Intra-session network coding uses the diversity of the wireless links, and codes packets from the same sessions, to address the packet loss problem and to provide reliability. In contrast, inter-session network coding mixes the packets from different sessions (sources) to solve the bottleneck problem and reduce the number of transmissions.

The MORE [16] and CCACK [17] are two opportunistic routing methods that use intra-session network coding to provide reliability in unicast and multicast applications. Later, the authors in [18] improved MORE and solved its Crying Babies problem. In [19] and [20], network coding is applied to decrease the number of required retransmissions due to packet loss in one-hop broadcasting over packet-erasure channels. In single-hop broadcasting over unreliable links, each destination node might lose some of the packets that are received by the other receivers. The idea in [19] is to code as many packets as possible together in each round of retransmissions, as to reduce the total number of transmissions.

The work in [21] uses intra-session network coding for the dissemination of data from a source node to the sensor nodes in wireless sensor networks. The R-Code method [22], constructs a minimum spanning tree, based on the reliability of the links, in which each non-leaf node works as a relay node. In R-Code, each parent node is responsible for delivering a sufficient number of intra-session linear coded packets to its children nodes, until they are all capable of decoding the coded packets..

The COPE method is a practical forwarding architecture for multiple unicast sessions [8, 9]. The nodes in COPE opportunistically listen to the transmissions and the relay nodes that are intersection of unicast flows use the advantage of inter-session network coding to reduce the number of transmissions. The authors in [2] show that, for fixed networks, inter-session network coding can, at most, offer a constant factor of benefits in terms of energy efficiency. They also propose a probabilistic network coding-based broadcasting algorithm. The work in [23] combines the partial dominant pruning (PDP) forwarding approach [5], which is a deterministic approach, with network coding. The algorithm uses local, two-hop topology information, and makes use of opportunistic listening to reduce the number of transmissions. Using network coding with directional antennas is considered in [11]. The work is based on the deterministic forwarding approach that uses directional CDS. It can be noted that the works that use the deterministic approach with network coding limit coding to XOR operations, and exploit only local coding opportunities.

3. SETTING AND MOTIVATION

We consider a multi-hop wireless network with multiple broadcast sessions, where a subset of the nodes are sources, and all of the nodes are destinations. Every packet has a deadline to reach each of the destinations. All of the nodes are synchronized, and all of the links are reliable. We assume that the nodes have multi-channel multi-radio capability. Thus, all of the nodes can transmit and receive simultaneously, and there is no conflict among the links. Also, we assume that each transmission takes one time slot to reach the next hop. Table I shows the set of symbols used in this paper.

Notation	Definition
s_i/d_j	The i-th source node/ The j-th destination node
T_i	Broadcasting tree rooted at source node s_i
p_i	Packet that originated at source node s_i
D_{ij}	Deadline of the packet p_i to be received by node d_j
u_i	The i-th node
t_i^j	Receiving time of a coded packet that contains packet p_i to node d_j
$\delta(u,v)$	The overlap-degree of node u to v
Δ_u	The maximum overlap-degree of node u
M	Total number of nodes
$N(u)$	The neighbors of node u, including node u.
R_{ij}	The remaining time of the packet p_i in node u_j.
E_{ij}	The extra time of packet p_i in node u_j.
H_{ij}	The maximum remaining hops of the longest branch from node u_j in tree T_i.
W_{ij}	The waiting time of the packet p_i in node u_j.
f_i	The number of outgoing flows from node u_j.
f_{ij}	The average value of the summation of f_k's of the branches from node u_j.
d	The diameter of the network.
G	The packet generation period.
t	The current time slot.

Table I. The set of symbols used in this paper.

Network coding reduces the number of transmissions, but increases the delay. The reason is that each node has to wait until it has received all of the incoming packets to code them together; the sending time of the coded packet should be at least the maximum arriving time of all of the received packets. In Fig. 2 (a), nodes 3, 6 and 5 are sources for packets p_3, p_6 and p_5, respectively. The deadline of the

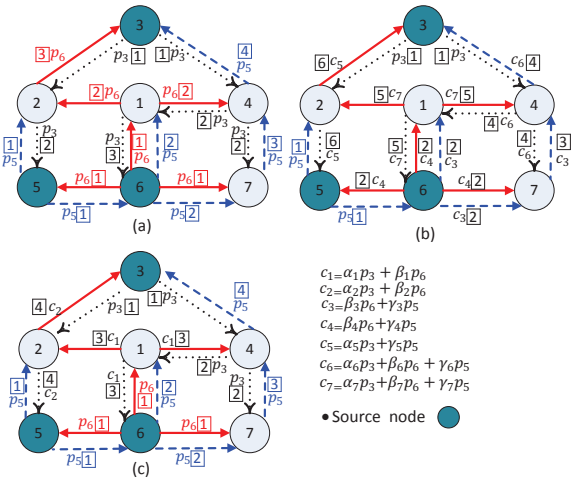


Figure 2. Broadcasting (a): without coding (b): coding p_3 and p_6 (c): coding all of the packets together.

packets p_3, p_6 and p_5 are time slots 5, 5 and 6, respectively. The sending time of each packet is shown in the box beside the packet. In this case, since there is no coding, all of the packets meet their deadlines, and the number of transmissions is 11. In Fig. 2 (b) all of the packets are coded together. Node 1 receives packets p_6, p_3 and p_5 at time slots 2, 3 and 5, respectively. Thus, in order to code these packets together, node 1 has to postpone the transmission of packets to time slot 5. Coding all of the packets together reduces the number of transmissions to 8, but causes a missed deadlines as node 3 receives packet p_6 after the packet's deadline. Therefore, we have to find another solution in which we reduce the number of the transmitted packets, while meeting the deadlines.

In Fig. 2 (c) we code only packets p_3 and p_6 together. The number of transmissions in Fig. 2 (c) is equal to 9, which is more than in Fig. 2 (b), but in Fig. 2 (c) there is no deadline miss. Therefore, an efficient solution for the problem of energy-efficient broadcasting with deadline constraints should be partitioning the set of packets such that coding the packets of each partition does not result missing a deadline. Thus, our problem becomes finding the set of partitions that minimize the total number of transmissions such that all of the packets meet their deadlines.

Wirel. Commun. Mob. Comput. 0000; **00**:1–16 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm
Prepared using wcmauth.cls

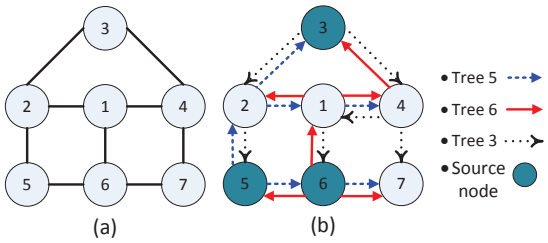


Figure 3. (a): A given topology. (b): Two broadcasting trees

4. DEALINE-AWARE LINEAR NETWORK CODING

In the appendix, we prove that the problem of energy-efficient broadcasting, subject to the deadline constraints, is NP-complete. Therefore, in this section, we propose a *deadline-aware network coding* (DANC) heuristic to solve the problem. For simplicity, we assume that each packet has the same deadline to reach all of the destinations. Our algorithm contains the following three phases:

- Constructing broadcasting trees: This phase ensures the decodability of the coded packets at the destination nodes. This phase is done once in the initializing phase.
- Partitioning the set of packets: The purpose of this phase is to guarantee meeting all the deadlines. This phase is done once in the initializing phase.
- Performing coding: In this phase, the relay nodes do the actual coding. This phase is repeated periodically.

By using broadcasting trees, each node receives enough linearly independent packets, so the nodes are able to decode the coded packets. If we allow all of the packets to be coded together, we can decrease the number of transmissions, but the delay increases, and some packets may miss their deadlines. Therefore, we partition the set of packets such that coding the packets of each partition does not result in deadline misses.

We assume that the broadcasting operation is periodic; we run the first two phases only once, then the third phase runs periodically. Thus, the complexity for the first two phases is not a major issue (however it is polynomial), though their performance is important, because they decide the operations of the third phase.

4.1. Constructing Broadcasting Trees

We use broadcasting trees to broadcast the packets. A broadcasting tree is a spanning tree rooted at one source node to reach all of the other nodes. Fig. 3 (b) shows three broadcasting trees. If a node is a non-leaf node in more than one broadcasting tree, it has the opportunity to code the received packets in order to send fewer packets. Assuming that there are K sources, we will have K broadcasting trees, so each destination node receives K coded packets, each of them from a different broadcasting tree. In order to ensure the decodability of the packets at the destination nodes, the K received packets have to be linearly independent.

We define the overlap-degree of node u to v as the the number of trees that use link (u, v), and we represent it as $\delta(u, v)$. Also, we define the maximum overlap-degree of node u as $\Delta(u) = \max_v(\delta(u, v))$. In Fig. 3 (b), $\delta(6, 1) = 1$, $\delta(6, 5) = 1$, and $\delta(6, 7) = 2$, so $\Delta(6) = 2$. In order to guarantee decodability, node 6 has to send two linearly independent coded packets to node 7. Each of these packets has to contain both of the packets p_4 and p_6. To make sure that the coefficient vectors in the buffers of all of the nodes achieve full rank, the number of transmissions at node u has to be at least equal to $\Delta(u)$. Consequently, if we can reduce Δ of each node, we can reduce the total number of transmissions. If we select the coefficients in a distributed manner at random, destination nodes will receive K linearly independent coded packets with high probability, almost 1 [15].

Our heuristic sequentially constructs broadcasting trees, each of them rooted at a source node. First, this approach sorts the sources in increasing order of the deadline of their packets. Then, in each iteration, our algorithm starts from a new source and traverses the network using the BFS algorithm. During traversal, each node not in the tree selects a node in the tree as its parent, based on the following two rules:

- $Rule_1$: Node v selects the parent u that has the maximum number of effective neighbors.
- $Rule_2$: Node v selects the parent u, where selecting that node does not increase $\Delta(u)$.

Effective neighbors of node u are the neighbors that do not have a parent in the tree. While constructing the broadcasting trees, we give more priority to $Rule_1$ over $Rule_2$. The reason is that a node with the maximum

Algorithm 1 Constructing broadcasting trees

for each source node u in ascending order of deadlines **do**
  Add node u to tree $T(u)$
  while there is a node $\notin T(u)$ **do**
    Select the next node $v \notin T(u)$ using the BFS algorithm
    Select node $w \in T(u)$ as v's parent, based on $Rule_1$ and $Rule_2$
    Select w as the parent of its neighbors $\notin T(u)$

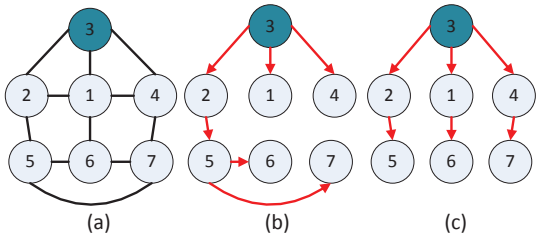


Figure 4. (a): A given topology. (b):Broadcasting tree constructed using $Rule_1$ and $Rule_2$. (c): Shortest path tree.

number of effective neighbors can cover more nodes by a single transmission. Algorithm 1 describes our algorithm.

Fig. 4 (b) shows a constructed broadcasting tree using our heuristic. The depth of nodes 6 and 7 in the constructed tree is 3, but the depth of the shortest path tree is 2. If the depth of a constructed broadcasting tree is more than the deadline of the packet of its source node, we will reconstruct that tree by adding a new rule to the algorithm. $Rule_3$: node u selects a parent with the minimum depth. We give more priority to this rule than the previous rules to guarantee meeting the deadline. The output of the new algorithm is a shortest path tree. The constructed tree is shown in Fig. 4 (c). The number of transmissions in Fig. 4 (b) is 3 and in Fig. 4 (c) is 4. Therefore, we cannot start from $Rule_3$, and we only use $Rule_3$ if we find that using $Rule_1$ and $Rule_2$ does not guarantee meeting the deadline.

Assume that in Fig. 3 we have constructed tree 5, and we want to construct tree 6. First, node 1 selects node 6 as its parent, and nodes 5 and 7 connect to node 6. Then, node 2 can select node 1 or 6 as its parent. Node 2 selects node 1, which has more (two) effective neighbors. If node 3 selects node 2, $\Delta(2)$ increases. Therefore, node 3 selects node 4 as the parent.

Algorithm 2 Partitioning the set of packets

Sort list of packets L in increasing order of deadlines
while $L \neq$ empty **do**
  $i \leftarrow i + 1$, Create new partition P_i
  Transfer the first packet of L to P_i
  for each packet p of L in ascending order **do**
    Using the RT algorithm, compute receiving times of the packets in $P_i \cup \{p\}$
    if no deadline misses **then**
      Delete p from L, Add p to P_i
Return $\{P_1, .., P_i\}$

4.2. Partitioning the Set of Packets

So far, we have discussed the first phase, which guarantees the decodability of the packets at the destination nodes. However, it does not guarantee meeting the deadlines. To prevent missing the deadlines, we have to decide which packets to code together. For this purpose, we use a greedy heuristic to partition the set of packets into different partitions, such that coding all of the packets of each set together does not result in deadline misses.

Our *Deadline-Aware Network Coding* (DANC) heuristic uses the constructed broadcasting trees. First, the algorithm sorts the list of the packets in increasing order of their deadlines (each packet belongs to the root of one tree). Then, the algorithm places the first packet of the list to the first partition. After that, the algorithm finds which packets can be added to the partition without causing deadline misses. The algorithm finds the remaining partitions using the same operation. The detailed algorithm is shown in Algorithm 2.

To compute the receiving times of the packets, we use the Receiving Time (RT) algorithm. First, for each relay node u, the RT algorithm finds the set of packets in partition P that node u is a relay node of. We represent this set as $R_P(u)$. Using the BFS algorithm, the RT algorithm traverses the trees of a given partition P simultaneously. If all of the traversal trees that their respective packets are in $R_P(u)$ have reached node u, the algorithm assigns the maximum arriving time of the trees, plus one (each transmission takes one time slot to reach the next hop), to the receiving time of the corresponding packets by the children nodes of node u.

In Fig. 2, the deadlines of the packet p_3, p_6 and p_5 are 5, 5 and 6, respectively. First, we add packet p_3 to partition P_1. Then we code packet p_6 with p_3 and compute the receiving times of the packets. The sending time of the

Wirel. Commun. Mob. Comput. 0000; **00**:1–16 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm
Prepared using wcmauth.cls

Algorithm 3 Performing coding

On receiving packet p by node u
if $u \in$ relay nodes of p **then**
Find the partition P such that $p \in P$
wait until receiving all of the packets $\in R_P(u)$
send $\Delta_P(u)$ random combination of the packets $\in R_P(u)$

packets are shown in Fig. 2 (c). Because all of the nodes receive both packets on-time, we add packet p_6 to partition P_1. Next, we code packet p_5 with the packets in P_1. Fig. 2 (b) shows the sending time of the packets. We cannot add packet p_5 to partition P_1, as nodes 1 and 5 receive packets p_3 and p_6 after their deadlines. Therefore, the partitions are $\{3,6\}$ and $\{5\}$.

4.3. Performing Coding

We extend $\Delta(u)$ to $\Delta_P(u)$. $\Delta_P(u)$ represents the maximum overlap-degree of node u for the packets in partition P. From the first two phases, each node u knows which packets it has to forward, and also it knows $\Delta_P(u)$ of each partition which specifies the number of transmissions of the coded packets of that partition. When a relay node u receives a packet p, it finds the partition P such that $p \in P$. Then, node u waits until it hase received all of the packets of partition P, so that the node is a relay node of $(R_P(u))$. Assuming that $\Delta_P(u) = m$, node u sends m random linear combinations of the packets. The relay nodes perform similar operations for the other partitions.

5. EXTENSIONS

5.1. Deadlock Detection and Recovery

Since there is more than one source in our problem, it is likely that deadlock occurs in the network. When there is a circular waiting among the processes (nodes) to access the resources (packets), a deadlock happens. Fig. 5 (a) shows a deadlock between two trees. In this figure, nodes 3 and 6 are sources. Node 4 receives a packet from node 3 and waits to receive the other packet from node 1. On the other hand, node 1 waits to receive a packet from node 4. As a result, we have a deadlock in this network.

We resolve the deadlock problem in the partitioning phase. To address the deadlock problem, we use a distributed deadlock detection and recovery scheme [24]. We allow deadlocks to happen, then we resolve them. To resolve that deadlock, at least one node among the nodes that causes the deadlock has to forward a packet without waiting for other packets. In Fig. 2, node 4 can break the deadlock by forwarding packet 3.

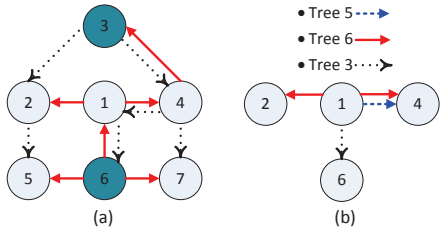


Figure 5. (a): A cycle between two trees. (b): Coding optimization.

Using convolutional codes [25], is another way to resolve the deadlocks. However, the complexity of convolutional codes limits their applicability. In our heuristic, we use linear coding which is less complex than convolutional codes, and can be implemented in a decentralized way. We also use deadlock detection and recovery to resolve the deadlocks.

5.2. Coding Optimization

Fig. 5 (b) shows a part of a network. Assume that the deadlines are set such that all of the packets can be coded together. Also, assume that node 1 has received packet p_5. In our former coding algorithm, node 1 has to wait to receive all of the packets. However, node 1 does not need to wait for other packets, and it can send packet p_5 immediately. Then, the next transmission covers packets p_3 and p_6. Therefore, node 1 can send the packets with the same number of transmissions and less delay. Only, we need to ensure that the sent packets are linearly independent and collectively cover all of the packets.

To reduce the coding delay, while preserving the number of transmissions required for each node u to be equal to $\Delta(u)$, we define the following rule. Node u can perform a transmission if for each child node v of u, at least one of the following conditions is true. $Condition_1$: the coded packet contains a new packet p_i, such that there is a link from nodes u to node v in tree T_i. $Condition_2$: node u has sent all of the packets that there is a link in their respective trees to node v. $Condition_3$: node u has transmitted less than $\Delta(u) - \delta(u, v)$ packets.

$Condition_1$ means that node u has a new packet for node v. When node v has received all of the necessary packets from node u, $Condition_2$ is true; node u does not need to send any more packets to node v. $Condition_3$ is related to the example in Fig. 5 (b). It means that $\Delta(u) - \delta(u, v)$ of the transmissions do not need to contain a new packet for node v. As a result, these transmissions can contain packets for only the children of u, such that the maximum overlap-degree of node u is to that node (node 4 in Fig. 5 (b)).

6. DEADLINE-AWARE LOCAL XOR CODING

6.1. Partial Dominant Pruning

The proposed deadline-aware linear network coding has two disadvantages. First, due to computational complexity of linear network coding, the DANC approach is not applicable to the nodes with limited computational power. Also, the first phase of the DANC method is a centralized approach. Therefore, in this section, we use local network coding instead of linear coding. The complexity and overhead of local coding is much less than that of linear coding. Also, all the phases of our proposed local network coding approaches are distributed. It should be noted that, in contrast with the proposed linear coding approach, our deadline-aware local network coding is applicable in both periodic and non-periodic broadcasting applications.

To prevent broadcast flooding, we can use global or local approaches. Since we want to address local network coding, we use the Partial Dominant Pruning (PDP) [5] broadcasting approach, which is a local method. However, in our local network coding methods, PDP can be replaced by other deterministic broadcasting methods. PDP is a local deterministic forwarding method. In PDP, each source node broadcasts its packet and selects a set of its one-hop neighbors as relay nodes, such that this set covers two-hop neighbors of the source. Each relay node performs the same process, and all of the nodes receive the broadcasted packet. This approach forms a tree from a source node to all other nodes.

We represent the set of neighbors of node u (including u) as $N(u)$, and the set of neighbors of $N(u)$ as $N(N(u))$ (nodes that are within two-hops from u). Assume that node

Algorithm 4 Coding

$P \leftarrow p_i$
for each remaining packet p_k in the queue **do**
  if all neighbors can decode $P \oplus p_k$ **then**
    $P \leftarrow P \oplus p_k$

u sends a broadcast packet to node v, and chooses this node as a relay node. Now, node v has to relay the packet and select a set of its one-hop neighbors as relay nodes to cover its two-hop neighbors. To minimize the number of transmissions, this set has to contain the minimum number of nodes. Nodes in $N(v)$ will receive the packet when node v broadcasts the packet, and nodes in $N(u)$ have already received it. Also, neighbors of common neighbors of nodes $N(u)$ and $N(v)$, will receive it. Therefore, node v has to select its relay nodes $R(u, v)$ from nodes in $B(u, v) = N(v) - N(u)$ to cover the nodes in $U(u, v) = N(N(v)) - N(u) - N(v) - N(N(u) \cap N(V))$. To find this set, a greedy set cover algorithm is used in [5]. At each step, the node in set B that covers the maximum number of nodes in U is added to the relay nodes. This process is repeated until all of the nodes of set U are covered.

In our problem, all of the nodes can be a source. All of the sources broadcast their packets based on the PDP algorithm, which is a local deterministic broadcasting approach. If a node is a relay node of more than one packet, it has opportunity to mix the received packets. In local network coding, if node u sends a coded packet P, a neighbor v of u should be able to decode the packet without waiting for further packets. In other words, each node u with a set of native packets p in its sending buffer seeks to find a subset of the native packets q to XOR. To decrease the number of transmissions, for each transmitted packet, node u has to maximize the number of neighbors which can decode a missing packet. In [23], it is proven that this problem is NP-complete. Therefore, a greedy algorithm is used to address this problem. This algorithm takes the packet p at the head of the sending queue and sequentially looks for other packets in the queue such that, if they are combined with p, all neighbors of node u will be able to decode the coded packet. If it finds such a packet, that packet is added to the set of coding packets. The procedure is described in Algorithm 4.

To increase coding opportunities, instead of sending the packets immediately, each forwarder node has to wait for a given time to receive more packets. Choosing the

Wirel. Commun. Mob. Comput. 0000; **00**:1–16 © 0000 John Wiley & Sons, Ltd.
DOI: 10.1002/wcm
Prepared using wcmauth.cls

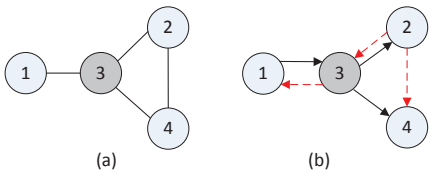


Figure 6. Waiting time and deadline.

appropriate waiting times is critical in this approach. Long waiting times can result in deadline misses, and short waiting times decrease coding opportunities. Fig. 6 (a) shows a given topology. Assume that nodes 1 and 2 are sources. Fig. 6 (b) shows the paths from the sources to the destinations. Assume that the sending time of p_1 and p_2 are 1 and 3, respectively. Also, assume that the deadline of p_1 and p_2 is time slot 6. Node 3 receives p_1 and p_2 at times 2 and 4, respectively. If we set the waiting time of p_1 at node 3 to 1, this node sends packet p_1 at time slot 3. Therefore, node 3 has to send two non-coded packets. On the other hand, if we set the waiting time to 4, node 4 receives p_1 at time slot 7, which is after the deadline. By choosing a waiting time of packet p_1 equal to 3, node 3 sends one coded packet instead of two non-coded packets.

To address the problem of choosing waiting times, we propose three approaches in the following sections.

6.2. Velocity-Based Waiting Time

Our constraint is meeting all of the deadlines. Thus, a relay node u_j can postpone the transmission of a packet if it finds that, by postponing the transmission, none of the nodes will receive that packet after the deadline. To make sure that all of the next-hop nodes receive packet p_i on-time, node u_j has to consider the receiving time of the packet by the deepest leaf node. The deepest leaf node is the farthest node of all of the branches in tree T_i that node u_j is the root of. Based on the maximum remaining hops, node u_j can calculate the extra time, which is the maximum allowable waiting time, and can use a portion of this time as its waiting time.

The Velocity-based Waiting Time approach (VWT) contains two phases, the initialization and running phases. In the first phase, based on the PDP algorithm, each source node u_s sends a packet to construct a tree T_s. Then, for each tree T_i, each leaf node u_k sends feedback, which contains the length of the longest branch from node u_k. We call this value *the maximum remaining hops*, and we represent it by H_{ik}. For the leaf nodes, we set $H_{ik} = 0$,

Algorithm 5 Velocity (Initializing phase)

if Node u_j is a leaf node **then**
  Send feedback $H_{ij} = 0$
else
  On receiving a feedback from node u_k to node u_j
  Store H_{ik}
  if Received feedback from all children nodes **then**
    $H_{ij} \leftarrow \max(H_{ik}) + 1$, Forward H_{ij} to the parent node

since there is no remaining hop from node u_k. Node u_j collects the feedback from all of its children nodes. It adds 1 to the maximum received value, stores it as H_{ij}, and relays it to its parent. Based on the feedback, each relay node u_j knows the remaining hops of the longest branch in tree T_i. The pseudo-code of the feedback part of the initializing phase is shown in Algorithm 5.

In the running phase (Algorithm 6), when a relay node receives a packet, it computes the remaining time of that packet by subtracting the current time from the deadline. Then, it subtracts the number of maximum remaining hops from the remaining time to compute the extra time. At the end, using a velocity-based approach [26], it computes the waiting time of the packet. Velocity-based approach means that the waiting time of the packet is calculated based on both the deadline and the maximum remaining hops. When the waiting time of a packet expires, the node uses Algorithm 4 to code the packet with other packets in its buffer and transmit it. The relay node computes the waiting time using equation $W_{ij} = \lfloor \frac{E_{ij}}{H_{ij}} \rfloor$, where $E_{ij} = R_{ij} - H_{ij}$ and $R_{ij} = D_i - t$. Here, D_i and t are the deadline of the packet p_i and the current time, respectively. The remaining time of the packet p_i, when it is at node, u_j is represented as R_{ij}. We use H_{ij} to represent the maximum remaining hops from node u_j. Based on our assumption, the nodes have the multi-channel multi-radio capability, and each transmission takes one time slot. Therefore, H_{ij} is equal to the remaining transmission time. Therefore, E_{ij} represents the extra time of packet p_i at node u_j. The waiting time for packet p_i at node u_j is represented by W_{ij}.

The selected paths based on the PDP approach from nodes 1 and 4 to other nodes are shown in Figs. 7 (a) and 7 (b). Assume that 1 and 4 send packets p_1 and p_4 at time slots 1 and 3, respectively. Also, assume that $D_1 = 7$ and $D_4 = 8$. Nodes 2 and 3 receive packet p_1 at time slot 2. Node 2 is a relay node in only one tree. Thus, it does

Algorithm 6 Velocity (Running phase)

On receiving a packet P by node u_j
for each Native packet $p_i \in P$ **do**
  if $u_j \in Forwarders(p_i)$ **then**
    $R_{ij} = D_i - t$, $E_{ij} = R_{ij} - H_{ij}$
    $W_{ij} = \lfloor \frac{E_{ij}}{H_{ij}} \rfloor$, $Timer_i \leftarrow W_{ij}$

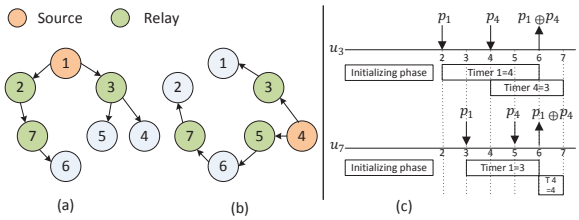


Figure 7. Velocity-based approach.

Algorithm 7 PDWT (Initialization phase)

if Node u_j is a leaf node **then**
  Send feedback $H_{ij} = 0$ and $f_{ij} = 0$
else
  On receiving a feedback from node u_k to node u_j
  store H_{ik} and f_{ik}
  if feedback has been received from all child nodes **then**
    $H_{ij} \leftarrow \max(H_{ik}) + 1$
    if $f_i > 1$ **then**
      $f_{i,j} \leftarrow mean(f_{ik}) + f_i$
    Forward H_{ij} and f_{ij} to the parent node

not have a coding opportunity, and forwards the received packet immediately. In contrast, node 3 is a forwarder node in both trees, so it computes the waiting time of the packet. At time slot 2, the remaining hops from node 3 in tree T_1 is 1. Therefore, $R_{1,3} = 7 - 2 = 5$, $E_{1,3} = 5 - 1 = 4$, and $W_{1,3} = 4$. Node 7 receives packet p_1 at time slot 3, and its waiting time $W_{1,7} = \frac{(7-3)-1}{1} = 3$. Node 3 and 7 receive packet p_4 at time slots 4 and 5, respectively. The waiting times of packet p_4 at nodes 3 and 7 are $W_{4,3} = \frac{(8-4)-1}{1} = 3$, and $W_{4,7} = \frac{(8-6)-1}{1} = 1$. Fig. 7 (c) shows that at nodes p_3 and p_7 the timers of packet p_1 expire after receiving packet p_7 and before the expiration of the timers of p_7. As a result, when the timer of p_3 is expired, nodes p_3 and p_7 check to see if they can mix packet p_1 with p_4. Since all of their neighbors can decode $p_1 \oplus p_4$, they send this coded packet.

6.3. Proportional Distribution of Waiting Time

There is a higher chance of coding at nodes that are relay nodes in more trees than other nodes, since they receive packets more frequently than other nodes. Also, the chance of mixing many packets together at these nodes is more than at other nodes. Therefore, in our second method, which is based on the Proportional Distribution of Waiting Time (PDWT), we distribute the extra time among different nodes in a way that is proportional to the number of outgoing flows that pass from these nodes. This means that if node u is a relay node of more flows than node v, we assign more waiting time to node u than to node v. We represent the number of outgoing flows from node u_i as f_i.

Similar to the previous approach, the new approach has the initialization and the running phases. The feedback part of the initialization phase is described in Algorithm 7. The only difference between the initialization phase of the PDWT and the VWT approaches is in the feedback. Similar to the previous approach, after constructing all of the trees, the leaf nodes of each tree T_i send back the number of remaining hops to their parent. They also send the number of outgoing flows that pass from them. When a parent node u_j receives all of the feedback from its children nodes in tree T_i, it records the average received value of f_{ik}'s. We represent the average received value from the children nodes of node u_j in tree T_i as f_{ij}. If f_j is more than one, it means that u_j is a coding node. In this case, node u_j adds f_j to f_{ij}, and if f_j is less than one, it does not change f_{ij}. Then, this node forwards f_{ij} to its parent. Also, node u_j forwards H_{ij}. This process is repeated until the source node of tree T_i receives this feedback. After the initialization phase, each node u_j knows H_{ij} and f_{ij}.

When a relay node receives a packet in the running phase, it uses the following equation to compute the waiting time of the packet (Algorithm 8):

$$W_{ij} = \lfloor \frac{E_{ij} \times f_j}{f_{ij}} \rfloor \quad (1)$$

Fig. 8 shows a part of tree T_1 and the outgoing flows from each node. We do not show the complete topology and all of the trees for brevity. In this example, after constructing all of the trees, node 6 sends $f_6 = 0$ as feedback. Node 4 receives this feedback. This node has two outgoing flows. Therefore, it stores and sends $f_{1,4} = 0 + f_4 = 2$. Node 3 has one outgoing flow, so it is not a coding node. Thus, it does not change the receiving feedback and sends $f_{1,3} = 2$. Then, node 2 forwards $f_{1,2} = 2 + 3 = 5$. This process is repeated for nodes 10 and 11. Node 7

DOI: 10.1002/wcm
Prepared using wcmauth.cls

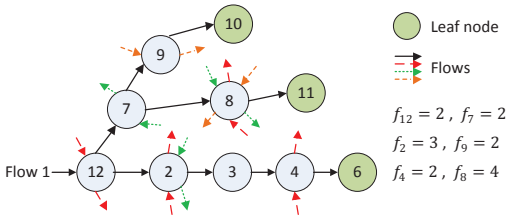


Figure 8. (a): Velocity-based approach (b):Proportional distribution of waiting time.

Algorithm 8 PDWT (Running phase)

On receiving a packet P by node u_j
for each Native packet $p_i \in P$ **do**
if $u_j \in Forwarders(p_i)$ **then**
$R_{ij} = D_i - t$, $E_{ij} = R_{ij} - H_{ij}$
$W_{ij} = \lfloor \frac{E_{ij} \times f_j}{f_{ij}} \rfloor$, $Timer_i \leftarrow W_{ij}$

Algorithm 9 Random (Running phase)

On receiving a packet P by node u_j
for each Native packet $p_i \in P$ **do**
if $u_j \in Forwarders(p_i)$ **then**
$R_{ij} = D_i - t$, $E_{ij} = R_{ij} - H_{ij}$, $Z_{ij} = \lfloor \frac{E_{ij} \times f_j}{f_{ij}} \rfloor$
$W_{ij} \leftarrow rand(\frac{Z_{ij}}{2}, \frac{Z_{ij}+E_{ij}}{2})$, $Timer_i \leftarrow W_{ij}$

receives two feedbacks $f_{1,8} = 4$, and $f_{1,9} = 2$. This node sends $f_{1,7} = \frac{4+2}{2} + 2 = 5$ to node 12. At the end, node 12 computes and sends $f_{1,12} = \frac{5+5}{2} + 2 = 7$. Assume that $E_{1,12} = 7$. In this case, $W_{1,12} = \lfloor \frac{7 \times 2}{7} \rfloor = 2$, so $E_{1,7} = 5$. Thus, $W_{1,7} = \lfloor \frac{5 \times 2}{5} \rfloor = 2$ and, $W_{1,8} = \lfloor \frac{3 \times 4}{4} \rfloor = 3$.

6.4. Random Waiting Time

The third proposed method is Random Waiting Time (RWT). The initialization phase of this approach is similar to the PDWT method. In this approach, each node computes a range of waiting time values and selects a random value from this range. To prevent deadline misses, node u_j cannot postpone the transmission of packet p_i more than E_{ij}. Also, it is not logical to use E_{ij} as the upper bound of the range, since it is possible to use the entire waiting time at the first nodes, which results in a much smaller waiting time at the remaining nodes. Therefore, we use $\frac{Z_{ij}+E_{ij}}{2}$ as the upper bound. Here, Z_{ij} represents the waiting time of packet p_i at node u_j that is computed by the PDWT approach. On the other hand, we use $\frac{Z_{ij}}{2}$ as the lower bound of the waiting time range. In conclusion, in the RWT approach, the initialization phase is similar to the PDWT approach. Then, in the running phase, each node uses Equation 1 to compute Z_{ij}, which is similar to the PDWT approach. Next, the node selects a random waiting time within the range of $(\frac{Z_{ij}}{2}, \frac{Z_{ij}+E_{ij}}{2})$. Algorithm 9 shows the running phase of this method.

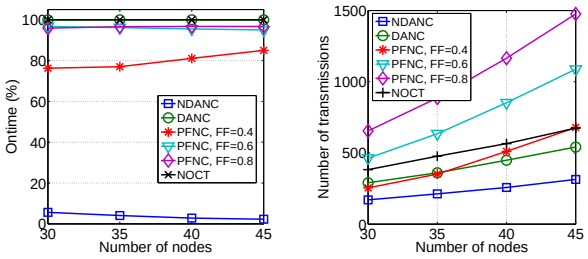
6.5. Unreliable Links

As mentioned in Section 3, we assume that the links are reliable, and addressing the problem of unreliable links is beyond the scope of this work. However, we give some hints that can be used to extend out proposed methods to address the case of unreliable links. In the proposed local network coding approaches, we use the maximum remaining hops H_{ij} from node u_j to calculate the remaining time E_{ij}. When the links are unreliable, each one-hop transmission might take more than one time slot. As a result, instead of the maximum hop counts H_{ij}, the expected total number of required time slots should be used. The expected required time slot for successful one-hop transmissions from node u_i to u_j is equal to $\frac{1}{r_{ij}}$, where r_{ij} is the reliability of the link between node u_i and u_j. The expected delivery time of a path C is equal to the summation of the expected required time slots of each link in the path C. A similar idea can be used to extend the proposed linear network coding approach.

7. SIMULATION RESULTS

First, we study the performance of the proposed linear network coding approach, the DANC (Deadline-Aware Network Coding) method. Then, we compare the proposed deadline-aware local network coding heuristics. Also, we will compare the performance of the linear network coding to the proposed local network coding. To the best of our knowledge, there are just two all-to-all broadcast methods that use inter-session network coding to reduce the number of transmissions. We compare the DANC method with the proposed method in [2], which uses random linear network coding. Moreover, we compare our local network coding methods with the work in [23] which, similarly to ours, applies local XOR coding.

As mentioned in the related work section, lots of intra-session methods have been proposed for unicast and multicast applications to provide reliability. Also, there are many inter-session network coding approaches for multiple unicast flows. However, none of them



(a) (b)

Figure 9. (a): Packets received on-time. (b): Total number of transmissions.

works for all-to-all broadcasting (or multiple broadcasting) applications.

7.1. Simulation Setting

We implemented a simulator in the MATLAB environment, since working with vectors, which are the core part of network coding, is easy in MATLAB. We assume that the nodes are synchronized and the nodes have multi-channel multi-radio capability. Therefore, there is no interference between the links. Based on this assumption, we implemented a simple MAC layer. We construct PDP and broadcasting trees, and perform network coding at the network layers. We evaluate the methods on 100 random topologies. The nodes are randomly scattered in a square field, $40m \times 40m$, and the communication range is $9m$. We perform our simulation on random topologies with 30, 35, 40 and 45 nodes, and with random deadlines in the range of $1.5r$ and $2r$, where r is the diameter* of the network. We also run the last simulation on a large network with 100 nodes to compare our local XOR and linear network coding methods. We assume that each node is a source of a broadcast packet. The plots in this paper are based on the average output of 100 runs.

7.2. Proposed Deadline-Aware Linear Coding Approach

In addition to the DANC approach, we simulate a *Non Deadline-Aware Network Coding (NDANC)* approach. In the NDANC approach, we put all of the packets in the same partition, and allow all of the packets to be coded together. We evaluate the DANC and NDANC approaches by comparing the number of transmissions, the on-time received packets, and the decodable packets. To find the number of decodable packets, we ignore the deadlines and compute the number of received packets that can be decoded at the destination nodes.

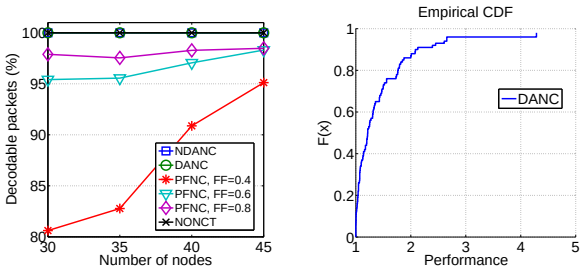
We compare our proposed methods with the *Probabilistic Forwarding with Network Coding (PFNC)* approach in [2], which uses random linear network coding. In [2], when a node receives an innovative packet, it sends a coded packet, of the innovative packets it has in its buffer, with a given probability. The value of this probability is called the *Forwarding Factor* (FF). We also simulate a deterministic, non-coding protocol. In this protocol, we use broadcasting trees to broadcast the packets, and we call it the *Non-Coding Tree* (NONCT) approach in the plots.

In the first experiment, we compare the number of packets received on-time. As is shown in Fig. 9 (a), the DANC method guarantees meeting the deadlines. Also, in the NONCT approach, all nodes receive the packets on-time because there is no coding. The number of on-time received packets increases as we increase the forwarding factor since, by increasing FF, each node forwards more packets. Because all of the packets are coded together in the NDANC, the delay increases as the number of nodes increases; as a result, more packets miss their deadlines. As we increase the number of nodes, the density and the degree of the nodes increases, which results in a larger number of transmissions in the PFNC approaches. Consequently, the ratio of the on-time received packets increases in the PFNC approaches.

Fig. 9 (b) shows the total number of transmissions for different approaches. Both of our heuristics have fewer transmissions than the PFNC approach. The reason is that the redundant transmissions are removed in our methods. Only the PFNC approach with $FF = 0.4$ has less transmissions than the DANC approach in the case of 30 nodes, but in this case, the number of packets received on-time with the PFNC approach is more than 20% less than that of the DANC approach. The DANC method has more transmissions than the NDANC approach since, in the NDANC approach, we code all of the packets together. As it is anticipated, the number of transmissions of the NONCT approach is more than the DANC and NDANC approaches, and the NDANC approach has the fewest number of transmissions.

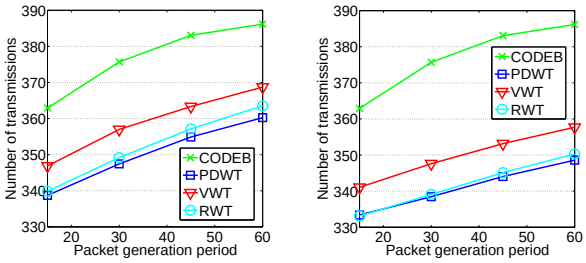
*The diameter of the network is the distance in terms of hop count between the farthest nodes of the network.

DOI: 10.1002/wcm
Prepared using wcmauth.cls



(a) (b)

Figure 10. (a): Total number of decodable packets. (b): Empirical CDF of performance DANC, compared with PFNC approach with forwarding factor 0.4.



(a) (b)

Figure 11. Effect of packet generation period on the number of transmissions- M=30. (a):$D \in (d, 1.5 \times d)$. (b):$D \in (1.5 \times d, 2 \times d)$

In the next experiment, we ignore the deadline constraints and compare the number of decodable packets at the nodes. Fig. 10 (a) shows that in the DANC, NDANC, and NONCT approaches, all of the nodes can decode all of the packets, as using broadcasting trees guarantees decodability. In contrast, some of the received packets in the PFNC approach are not decodable. By increasing the forwarding factor, due to the increase in the number of transmissions, the number of decodable packets in the PFNC approach increases.

Fig. 10 (b) shows the performance of the DANC method. For each simulation run, we calculate the ratio of the number of on-time received packets in the DANC method and in the PFNC approach with a forwarding factor of 0.4, and we show the empirical CDF of the result. It can be seen that the performance of the DANC approach is always better than that of the PFNC approach. In about 20% of the cases, the performance of the DANC approach is more that two times that of the PFNC approach.

7.3. Proposed Deadline-Aware Local Coding Approach

We compare our proposed local network coding methods with the network coding method in [23], called CODEB. CODEB constructs PDP as the broadcasting structure. The relay nodes in this method use inter-session network to reduce the number of transmissions. CODEB sets a random waiting time for the packets in the case of delay tolerant networks. For the case of non-delay tolerant networks, there is no waiting time for the packets, but coding is still performed. We represent the packet generation period by G. This means that each node in every G time slots generates one packet. Therefore, G is inversely proportional to that of the packet generation rate. We vary the packet generating period from $\frac{M}{2}$ to $2 \times M$. Here, M is the number of nodes.

In the first experiment, we study the effect of packet generation period on the number of transmissions. In Fig. 11 (a) the deadlines are in the range of 1 to 1.5 times the diameter of the network. The number of transmissions of the methods increases as we increase G. The reason is that, by increasing the generation period of the packets, the expected number of received packets at each time slot decreases. Thus, the coding opportunity decreases. In Fig. 11 (b) we change the deadlines to be in the the range of 1.5 and 2 times the diameter. It can be inferred from this figure that the difference between our proposed methods and the CODEB method increases as we increase the generation period of the packets.

In the next experiment, we study the effect of deadlines on the number of transmissions. When the deadlines of the packets are in the range of 0.5 to 1 times the diameter of the network, the extra time of the most of the packets at relay nodes is near zero. Thus, the number of transmissions of all of the methods are close, which can be seen in Figs. 12 (a) and (b). In this case, all of the methods have misses, but since they are exactly the same, we exclude this case. The number of transmissions is inversely proportional to the the deadlines. The reason is that by increasing the deadlines, the packets have more waiting time, which increases the chance of coding. The number of transmissions of the CODEB method is fixed, since relay nodes forward the received packets immediately, without considering the deadlines. It can be inferred from Fig. 12 (a) that the PDWT method has the lowest number of transmissions compared to the other approaches.

The period of generating packets in Fig. 12 (b) is half of the period in Fig. 12 (a). The number of transmissions in

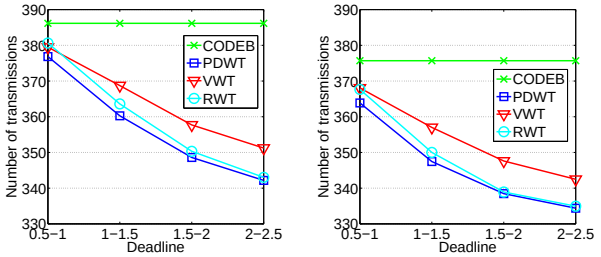


Figure 12. Effect of deadlines on the number of transmissions- M=30. (a):$G = 60$. (b):$G = 30$.

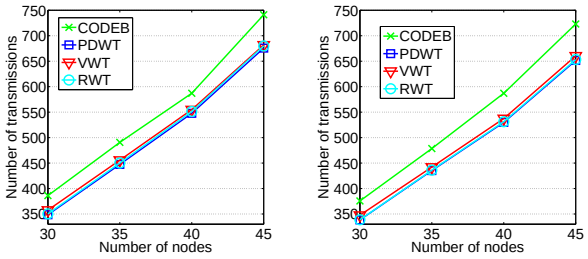


Figure 13. Effect of number of nodes on the number of transmissions- $D \in (1.5 \times d, 2 \times d)$. (a): $G = M$. (b): $G = 2 \times M$

Fig. 12 (b) is less than that in Fig. 12 (a). The reason is that the relay nodes have more chances of coding the received packets.

Figs. 13 (a) and (b) show the effect of the number of nodes on the number of transmissions. The number of transmissions of all of the methods increases as we increase the number of nodes, since the number of generated packets increases. The packet generation period in Fig. 13 (b) is half of that in Fig. 13 (a); the number of transmissions is less than that in Fig. 13 (a).

Fig. 14 (a) shows the number of transmissions in the DANC and PDWT approaches. We set the deadlines randomly in the range of $1.5r$ and $2r$. This figure shows that linear network coding is up to 27% more efficient than local network coding. The CDF of the performance (in terms of number of transmissions) of the DANC approach over the PDWT method in the case of 100 nodes is shown in Fig. 14 (b). It can be inferred from this figure that the performance of our linear network coding method (DANC) is always more that that of our local XOR coding (PDWT).

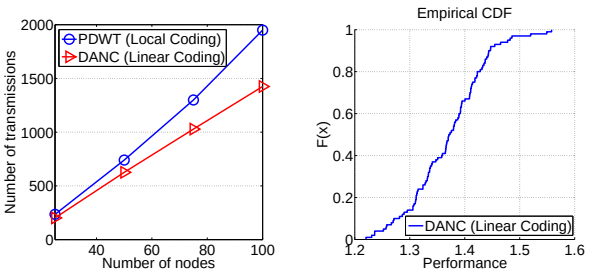


Figure 14. Linear network coding VS. local XOR network coding.

8. CONCLUSION

In this paper, we study the problem of energy-efficient broadcasting with deadline constraints. We prove that this problem is NP-complete. Thus, we propose a deadline-aware heuristic to solve this problem. We use the concept of broadcasting trees to select forwarder nodes. Our DANC heuristic classifies the packets into sets, such that coding all of the packets of each set does not result in a deadline miss. Our heuristic also works for the case when packets do not have deadline constraints. In wireless networks with periodic broadcasting, our protocol computes the coding decision once, and based on that, each node determines its responsibility in future rounds. Also, for the networks with limited computational power, we provide three heuristics to compute the local waiting time of the packets at relay nodes to improve the efficiency of local network coding without missing deadlines.

ACKNOWLEDGMENT

This research was supported in part by NSF grants ECCS 1128209, CNS 1138963, CNS 1065444, and CCF 1028167.

REFERENCES

1. Y. Sasson, D. Cavin, and A. Schiper, “Probabilistic broadcast for flooding in wireless mobile ad hoc networks,” in *IEEE WCNC 2003*, vol. 2, Mar 2003, pp. 1124–1130.

2. C. Fragouli, J. Widmer, and J. L. Boudec, "Efficient broadcasting using network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 450–463, Apr 2008.
3. D. Dubhashi, O. Häggström, L. Orecchia, A. Panconesi, C. Petrioli, and A. Vitaletti, "Localized techniques for broadcasting in wireless sensor networks," *Algorithmica*, vol. 49, no. 4, pp. 412–446, 2007.
4. J. Wu and H. Li, "On calculating connected dominating sets for efficient routing in ad hoc wireless networks," in *Proc. International Workshop Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999.
5. W. Lou and J. Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *IEEE Transactions on Mobile Computing*, vol. 1, no. 2, pp. 111– 122, Apr-Jun 2002.
6. R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.
7. S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.
8. S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," in *ACM SIGCOMM*, 2006.
9. ——, "XORs in the air: practical wireless network coding," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 243–25, 2006.
10. A. Khreishah, C. Wang, and N. Shroff, "Cross-layer optimization for wireless multihop networks with pairwise intersession network coding," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 5, pp. 606–621, 2009.
11. S. Yang and J. Wu, "Efficient broadcasting using network coding and directional antennas in MANETs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 2, pp. 148–161, Feb 2010.
12. X. Li, C.-C. Wang, and X. Lin, "Throughput and delay analysis on uncoded and coded wireless broadcast with hard deadline constraints," in *IEEE INFOCOM*, 2010.
13. C. Zhan and Y. Xu, "Broadcast scheduling based on network coding in time critical wireless networks," in *IEEE International Symposium on Network Coding*, June 2010.
14. R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782– 795, Oct 2003.
15. T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.
16. S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *ACM SIGCOMM*, 2007.
17. D. Koutsonikolas, C. Wang, and Y. Hu, "Efficient network coding based opportunistic routing through cumulative coded acknowledgments," *IEEE/ACM Transactions on Networking*, vol. 19, no. 5, pp. 1368–1381, Feb 2011.
18. D. Koutsonikolas, Y. Hu, and C. Wang, "Pacifier: High-throughput, reliable multicast without"crying babies"in wireless mesh networks," in *IEEE INFOCOM*, 2009, pp. 2473–2481.
19. D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 2, pp. 914–925, 2009.
20. W. Fang, F. Liu, Z. Liu, L. Shu, and S. Nishio, "Reliable broadcast transmission in wireless networks based on network coding," in *IEEE INFOCOM Workshops (INFOCOM WKSHPS)*, 2011, pp. 555–559.
21. W. Xiumin, W. Jianping, and X. Yinlong, "Data dissemination in wireless sensor networks with network coding," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, 2010.
22. Z. Yang, M. Li, and W. Lou, "R-code: Network coding based reliable broadcast in wireless mesh networks with unreliable links," in *IEEE GLOBECOM*, 2009.
23. L. Li, R. Ramjee, M. Buddhikot, and S. Miller, "Network coding-based broadcast in mobile ad-hoc networks," in *IEEE INFOCOM*, May 2007.
24. J. Wu, *Distributed System Design*. CRC Press, 1999.
25. E. Erez and M. Feder, "Convolutional network codes for cyclic networks," in *Proc. NETCOD 2005 Workshops*, 2005.
26. C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He, "Rap: A real-time communication architecture

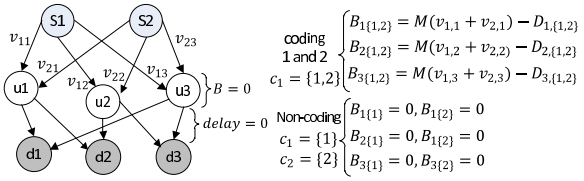


Figure 15. A reduction from an instance of a vector packing problem to an energy-efficient broadcast with deadline constraints. (B_{jc} represents the decoding delay for the packets of the set c at node d_j.)

for large-scale wireless sensor networks," in *Proc. of IEEE RTAS*, 2002, pp. 55–66.

A. COMPLEXITY

Theorem 1

The problem of energy-efficient broadcasting, subject to the deadline constraints, is NP-complete.

Proof

In order to show that the problem is NP-complete, we need to show that it is NP and NP-hard.

It is easy to show that this problem is NP; if we are given the topology, the set of sources and destinations, and the energy consumption at every node, we can verify in polynomial time, using the BFS algorithm, that these parameters solve the problem.

In order to show that it is NP-hard, we need to provide a polynomial time reduction from a well known NP-complete problem. We choose the *vector packing problem* as the known NP-complete problem. In vector packing, we have K vectors, each with N positive integers. The i-th vector can be represented by $V_i = [v_{i1}, \ldots, v_{iN}]$. We also have identical bin vectors. Each bin vector contains N integers and can be represented by $[b_1, \ldots, b_N]$. The problem is packing the vectors in as few bins as possible. The constraint is that the sum of the vectors in each bin cannot exceed the size of the bin. Formally, the problem can be described as minimizing L, subject to: $\sum_{i \in l} v_{ij} \leq b_j, \quad \forall l \in \{1, \ldots, L\}, \forall j \in \{1, \ldots, N\}$, where $i \in l$ means that the i-th vector is packed in the l-th bin.

The reduction is as follows. For every instance of the vector packing problem, we generate an instance of our problem according to the following rules. First, we place K sources, N intermediate nodes $u_i, i \in \{1, \ldots, N\}$, and N destination nodes in the graph. We connect each source to all intermediate nodes and each intermediate node to K different destinations, such that each destination node has K input links from K different intermediate nodes. Then, we set the delay of the link between s_i and u_j to $v_{ij}, \forall i, j$, the delay for the links connecting the u and d nodes to zero, and all of the transmission costs to zero except for the u nodes, where we set the cost to 1, per sent packet.

Let $M = \frac{\max_{ij}(v_{ij})}{\min_{ij}(v_{ij})}$ and let D_{jc} represent the delay for receiving the $|c|$ linearly independednt packets from the set c at node j when we choose to code the packets in the set c. We set the decoding delay for the packets of the set c at node d_j as $M \sum_{i \in c} v_{ij} - D_{ic}$. Note that, due to the multiplication by M, the decoding delay is always ≥ 0. Therefore, the total delay at node d_j to receive and decode the packets in the set c would be $M \sum_{i \in c} v_{ij}$. Also, the total cost of transmissions is proportional to the number of partitions where coding is allowed. This is due to setting the cost of all transmissions to zero, except the cost of transmissions at the u nodes.

After doing this reduction, if we set the deadline of packet p_i to reach d_j to Mb_j, it is easy to see that the vector packing problem is solvable iff the minimum cost deadline-aware problem is solvable on the constructed graph. □

Fig. 15 shows a reduction from a vector packing problem with three 2-dimensional vectors to an energy-efficient broadcasting problem with deadline constraints.

DOI: 10.1002/wcm
Prepared using wcmauth.cls