Memory Aware Wireless Real Time Broadcast with Network Coding

Cheng Zhan and Fuyuan Xiao
School of Computer and Information Science, Southwest University, ChongQing, 400715 P.R.China.
Email: zhanc@swu.edu.cn, fyuanxiao@tom.com

Abstract —In this paper we study network coding based wireless broadcast scheduling problem in real time applications under the memory model, aiming at minimizing the number of transmissions. In the memory model, receiver has enough memory, and a receiver will buffer all received encoded packets and decode out their "wanted" packets when enough packets are received. We prove that the coding based scheduling problem under memory model is NP-hard. Based on the graph model and the matrix transformation, we propose an effective heuristic algorithm consisting of a two stage code construction. Simulation results show that our algorithm can significantly reduce the number of transmissions in most cases, which is an important performance metric in real time applications.

Index Terms—Network coding, broadcast scheduling, real time, memory

I. INTRODUCTION

In wireless applications, broadcasting data to multiple users is commonly used, such as satellite communications, WiFi networks, etc. By coding multiple original packets into a single coded packet, network coding can improve transmission efficiency and throughput over broadcast channels [1]-[3]. The works in [4], [5] studied coding based broadcast schemes for loss recovery that allow instantaneous decoding, where sender decides how to encode and transmit based on the cached information at receivers. These coding schemes are also known as Instantly Decodable Network Codes (IDNC). Previous work on IDNC focused on minimizing the completion time and decoding delay. However, there are few works considering the delay guarantee of data packets, which is an important aspect of quality of service.

Recent development of commercial wireless services has created large scale demands for real time applications such as video streaming or interactive gaming. These real time applications have a distinct characteristic: they have strict and urgent deadlines, and a packet is useless (or less useful) after a short amount of time [6], [7]. Inspired by the above observations, the work in [8] studied the coding schemes for real time applications based on IDNC and found a code that is instantly decodable by the maximum number of users to minimize the number of transmissions. However, in the existing IDNC scheme, receivers are forced to discard all packet combinations that cannot be immediately decoded, which means that the memory of receivers cannot be utilized efficiently. The two-user broadcast scenario with memory was analyzed in [9], it is shown that using the memory of receivers can increase the capacity region, however the work did not consider the encoding strategy for multiple users.

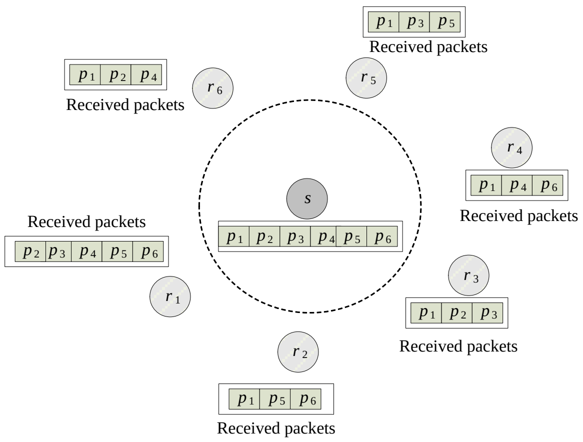


Fig. 1. An example

Consider a single hop broadcast scenario with a sender s and six receivers $r_1,r_2,\ldots,r_6$ as shown in Fig. 1. Suppose that s needs to send six packets $p_1,p_2,\ldots,p_6$ to the receivers. Due to overhearing or prior transmission, receivers already have some packets and inform these information to sender according to feedback. As shown in Fig. 1, r_1 has packets $\{p_2,p_3,p_4,p_5,p_6\}$, r_2 has packets $\{p_1,p_5,p_6\}$, r_3 has packets $\{p_1,p_2,p_3\}$, r_4 has packets $\{p_1,p_4,p_6\}$, r_5 has packets $\{p_1,p_3,p_5\}$, and r_6 has packets $\{p_1,p_2,p_4\}$.

TABLE I: PACKET INFORMATION AT RECEIVERS

	p_1	p_2	p_3	p_4	p_5	p_6
r_1	1	0	0	0	0	0
r_2	0	1	1	1	0	0
r_3	0	0	0	1	1	1
r_4	0	1	1	0	1	0
r_5	0	1	0	1	0	1
r_6	0	0	1	0	1	1

The buffered packet information at the receivers is shown in Table I, where 0 indicates that the corresponding packet is already received at the receivers

Manuscript received July 7, 2015; revised December 8, 2015.
This work was supported by the Fundamental Research Funds for the Central Universities (No. SWU115002, No. XDJK2015C104).
Corresponding author email: zhanc@swu.edu.cn
doi:10.12720/jcm.10.12.997-1003

and 1 represents that the corresponding packet is needed at the receivers.

According to the feedback information, s needs 6 transmissions by the traditional way, i.e. transmitting $p_1,p_2,\ldots,p_6$ at each time slot respectively. However, by the encoding strategy introduced in [8], all needed packets can be recovered at all receivers by transmitting the following 4 encoded transmission packets $p_4\oplus p_5$, $p_2\oplus p_4$, $p_3\oplus p_5$ and $p_1\oplus p_6$ in sequence. For example, with $p_4\oplus p_5$, r_2 and r_5 can recover p_4, r_4 and r_6 can recover p_5. With $p_2\oplus p_4$, r_2 can recover p_2 according to $p_4\oplus(p_2\oplus p_4)$ using p_4 which is recovered before.

If each receiver can buffer its received encoded packets, it can accumulate the useful information to recover all its needed packets, which can reduce the number of transmission packets. Under such a strategy, for the example given in Fig. 1, we only need to transmit the following 3 encoded packets, namely, $p_1\oplus p_2\oplus p_6$, $p_3\oplus p_5\oplus p_6$ and $p_4\oplus p_5$. When r_5 receives the first encoded packet, the packet is useless for r_5 because r_5 needs both p_2 and p_6. If r_5 can buffer this received packet, when it receives the second encoded packet, it can recover p_6 since it already has p_3 and p_5. After p_6 is recovered at r_5, r_5 can use the first encoded packet to recover p_2 since it has p_6 and p_1 by now.

Inspired by the above examples, our aim in this paper is to determine the encoding strategy at the sender to minimize the number of transmissions under the receiver model which is referred to *memory model*. In the memory model, receiver has enough memory, and a receiver will buffer all received encoded packets and decode out their "wanted" packets when enough packets are received. Our contributions are summarized as follows:

- We study the coding based scheduling problem under memory model and prove that the scheduling problem is NP-hard.
- Based on graph model and matrix transformation, we propose an effective heuristic encoding algorithm using a two stage code construction scheme.
- Simulation results show that the proposed transmission strategy can significantly reduce the number of transmissions in most cases.

The remainder of this paper is organized as follows. In Section II, we will give the problem statement. The coding based scheduling problem under memory model is proved to be NP-Hard in Section III. In Section IV, we will propose our encoding algorithm based on the graph model. Decoding algorithm will be proposed in Section V. Simulation results will be shown in Section VI. Finally, we will conclude the paper in Section VII.

II. Network Model and Problem Description

Consider a single hop wireless broadcast scenario where there are a sender s and n receivers $R=\{r_1,r_2,\ldots,r_n\}$. s needs to transmit m packets $P=\{p_1,p_2,\ldots,p_m\}$ to the n receivers. Due to the overhearing or prior transmissions, receivers may have some packets in their caches. Each receiver only needs a subset of packets in P since it already had some packets in their caches. Set $N(r_i)$ denotes the packets needed at receiver r_i and set $H(r_i)$ denotes the packets already had at receiver r_i, so we have $N(r_i)\bigcap H(r_i)=\varnothing$, $N(r_i)\bigcup H(r_i)=P$. We assume that time is slotted, and at each time slot, the sender transmits one coded packet.

The problem is that given the set of stored packets $H(r_i)$ at the receiver r_i, the set of packets $N(r_i)$ needed by the receiver r_i, $1\le i\le n$, how to encode and transmit packets in each time slot to minimize the number of transmissions under memory model. In the memory model, a receiver will buffer all received encoded packets and decode them when enough encoded packets are received. This model is suitable for the nodes which have enough memory size such as mesh network and vehicular network nodes. Such an encoding decision problem is referred to as Memory Encoding (ME) problem. In this paper, we consider XOR coding since it is easy to be implemented with trivial overhead.

III. Memory Encoding

A. Memory Encoding Problem

In the memory model, receivers store packets which cannot be decoded immediately, and decode until enough encoded packets are received. Our Memory Encoding (ME) problem is to decide how to encode and transmit packets in each time slot to minimize number of transmissions with the assumption that receivers can decode until enough encoded packets are received. In the following, we will prove that ME problem is NP-hard by reducing the simultaneous matrix completion problem which is a well-known NP-complete problem, to ME problem.

A simultaneous matrix completion problem is defined as follows. Given a set of mixed matrices, each matrix contains a mixture of numbers and variables, and each particular variable can only appear once per matrix but may appear in several matrices. The objective of the simultaneous matrix completion problem is to find values for these variables such that all resulting matrices simultaneously have full rank. It was shown in [10] that simultaneous matrix completion problem over $GF(2)$ is NP-complete problem.

Theorem 1. The Memory Encoding problem is NP-hard.

Proof. Since a receiver can at most recover one original packet upon receiving an encoded (xor) packet, any receiver needs at least $l=max_{r_i\in R}|N(r_i)|$ encoded packets. In order to minimize the number of transmissions, we should construct at least l encoded packets from which the receivers can decode all the needed packets. Assume that the encoded packets are $p'_1,p'_2,\ldots,p'_l$, we can construct a $l\times m$ coefficient matrix $M_{l\times m}$ which contains 0,1 entries from the encoding vector of these packets. For receiver r_i, if $p_j \in N(r_i)$, we keep the j-th column in $M_{l\times m}$, otherwise, we remove it from $M_{l\times m}$. After such transformation, we get a $l\times l$ sub-matrix $M^i_{l\times l}$. $M^i_{l\times l}$ can be used to illustrate the encoded packets containing only the packets in $N(r_i)$. If $rank(M^i_{l\times l}) = l$, then r_i can decode all the needed packets in $N(r_i)$. Thus, given the matrix $M_{l\times m}$ whose elements are variables and sub-matrix set $\{M^i_{l\times l}\}, 1 \le i \le n$, we want to find a value assignment of $M_{l\times m}$ over $GF(2)$ to make sure that all matrices in $\{M^i_{l\times l}\}$ are full rank. Thus, we find that the memory encoding problem is equivalent to the simultaneous completion problem over $GF(2)$ which is NP-complete shown in [10], thus the Memory Encoding problem is NP -hard.

B. Graph Model

Given the information about the "Have" and "Need" sets of receivers, we form an IDNC graph as in [4]:

Definition 1. Given $R = \{r_1, r_2, ..., r_n\}$, $P = \{p_1, p_2, ..., p_m\}$, $N(r_i) \subseteq P$, $H(r_i) \subseteq P$, $N(r_i) \bigcap H(r_i) = \varnothing$, we construct a graph $G(V,E)$ as:

$$V = \{v_{ij} \mid \text{packet } p_j \text{ needed by } r_i\},$$

$$E = \{(v_{i_1 j_1}, v_{i_2 j_2}) \mid j_1 \ne j_2, p_{j_2} \in H(r_{i_1}), p_{j_1} \in H(r_{i_2})\} \bigcup \{(v_{i_1 j_1}, v_{i_2 j_2}) \mid j_1 = j_2\}.$$

For each packet $p_j \in N(r_i)$, there is a corresponding vertex $v_{ij} \in V(G)$. Table II shows the notations to be used in the constructed graph model and the proposed encoding algorithm. Fig. 2 is the corresponding graph of the aforementioned example in Section I.

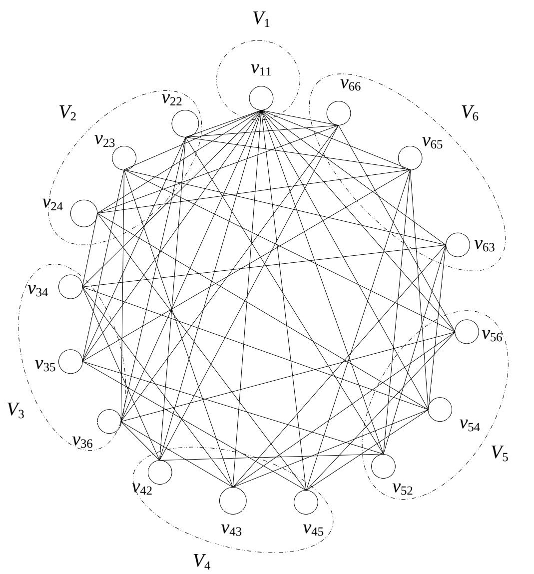


Fig. 2. A graph example

TABLE II: NOTATION

Symbol	Description
s	The sender
r_i	Receiver i
n	Number of receivers
$H(r_i)$	The set of packets already had at receiver r_i
$N(r_i)$	The set of packets needed at receiver r_i
P	The set of packets to be transmitted
p_j	The j-th packets to be transmitted
m	Number of packets to be transmitted
G	The constructed graph
$V(G)$	The vertex set of graph G
$E(G)$	The edge set of graph G
v_{ij}	A vertex corresponds to packet p_j needed by r_i in graph G
C	A clique in graph G

In Fig. 2, v_{11} represents that r_1 needs packet p_1. Since $p_1 \in H(r_6)$ and $p_6 \in H(r_1)$, there is an edge (v_{11},v_{66}). r_2 and r_6 need the same packet p_3, then there is an edge (v_{23},v_{63}).

It was shown in [4] that one encoded packet corresponding to a clique $C \subseteq V(G)$ can help some receivers recover their needed packets immediately. Intuitively, finding the clique containing maximum vertices will serve more receivers. The work in [8] used the maximum clique of IDNC graph to reduce the number of transmissions, under the assumption that instantly decoding is conducted at the receivers, i.e. once the encoded packet p' arrives at receiver r_i, if r_i cannot decode p' immediately, r_i just drops packet p'. However, if we allow receivers to store packets which cannot be decoded immediately, we can further reduce the number of transmissions. In our memory model, receivers store packets which cannot be decoded immediately, and decode until enough encoded transmission packets are received.

IV. ENCODING ALGORITHM

We propose an encoding algorithm which aims to satisfy all all needed packets at the receivers with minimum number of transmissions. Let $l_{max} = max_{r_i \in R} |N(r_i)|$, where $|N(r_i)|$ is the cardinality of $N(r_i)$. Since a receiver can at most recover one original packet upon receiving an encoded (xor) packet, any encoding algorithm needs at least l_{max} transmission packets. Therefore, l_{max} is the lower bound of the number of transmissions under the ideal case where all packets are successfully received by all receivers. The proposed encoding algorithm consists of two stages. At the first stage, we construct l_{max} encoded packets. Each encoded packet is the xor of some original packets, of which, at most one is selected from the needed packets of a receiver. With these l_{max} encoded packets, some receivers may not decode out all needed packets. Therefore, at the second stage, we need to append more necessary packets to complete the service.

At the first stage, we construct l_{max} encoded packets. According to the definition of G above, for any $i, 1 \le i \le n$, if $j \ne k$, $(v_{ij}, v_{ik}) \notin E(G)$. We can partition V into n subsets $\{V_1, V_2, \ldots, V_n\}$, where $V_i = \{v_{ij} \mid p_j \in N(r_i), 1 \le j \le m\}$ is the set of needed packets of receiver r_i. The main idea of our construction algorithm is that, we select a needed packet corresponding to a vertex in every V_i and xor them together as one encoded transmission packet. For every V_i, we select v_{ij_m}, $j_m = min\{j \mid v_{ij} \in V_i\}$. After encoded packet being decided, we delete v_{ij_m} in G. At the end of the construction algorithm, we get l_{max} encoded packets. This stage constructs a partial solution for Memory Encoding problem. The initial partial solution construction algorithm is given in Fig. 3.

```
l ← 0; P′ ← ∅;
while (V(G) ≠ ∅ ) do
    p′_l ← 0; Q(p′_l) ← ∅;
    for (i← 1 to n) then
        j_m ← min{j|v_ij ∈ V_i};
        if (p_{j_m} ∉ Q(p′_l)) then
            p′_l ← p′_l ⊕ p_{j_m};
            Q(p′_l) ← Q(p′_l) ∪ {p_{j_m}};
        end if
        delete v_{ij_m} in G;
    end for
    P′ ← P′ ∪ {p′_l};
    l ← l + 1.
end while
```

Fig. 3. Initial partial solution construction algorithm for memory encoding problem

For example, suppose that sender s needs to transmit packets $P=\{p_1,p_2,p_3,p_4,p_5,p_6\}$, r_1 needs $\{p_1,p_2\}$, r_2 needs $\{p_1,p_3\}$, r_3 needs $\{p_1,p_4\}$, r_4 needs $\{p_2,p_3\}$, r_5 needs $\{p_2,p_4\}$, r_6 needs $\{p_3,p_4\}$. Accordingly, $H(r_i) = P \setminus N(r_i), 1 \le i \le 6$. From the definition of graph G, we get the vertex subsets $V_1=\{v_{11},v_{12}\}$, $V_2=\{v_{21},v_{23}\}$, $V_3=\{v_{31},v_{34}\}$, $V_4=\{v_{42},v_{43}\}$, $V_5=\{v_{52},v_{54}\}$, $V_6=\{v_{63},v_{64}\}$. The initial partial solution is $P' = \{p_1 \oplus p_2 \oplus p_3, p_2 \oplus p_3 \oplus p_4\}$.

Given P' constructed at the first stage, we need to find out whether P' is enough and what else are needed for completing the service. At the second stage, we append necessary packets based on the initial partial solution such that all receivers can recover their needed packets.

For receiver r_i, when it receives an encoded packet p'_k, let P'_k be the set of packets used to encode p'_k, we define a receiving vector $v_{ik}=(a_1,a_2,\ldots,a_m)$, $a_j=1$ if $p_j \in N(r_i)$ and $p_j \in P'_k$, else $a_j=0$, $1 \le i \le n, 1 \le j \le m$. Assume that r_i needs k packets and it has received l encoded packets, $l \ge k$. Let $N(r_i) = \{p_{i_1}, p_{i_2}, \ldots, p_{i_k}\}$. r_i can construct a matrix $M^i_{l\times m}$ based on receiving vectors. If $p_j \in N(r_i)$, we keep the j-th column in $M^i_{l\times m}$, otherwise, we remove it from $M^i_{l\times m}$. After such transformation, we get a $l \times k$ sub-matrix $M^i_{l\times k}$.

The following transformation can find out whether the j-th needed packet at r_i can be recovered or not. We apply Gaussian elimination on $M^i_{l\times k}$, if $l>k$, there exists some all-zero rows. After randomly deleting some all-zero rows, we can transform $M^i_{l\times k}$ from a $l \times k$ matrix to a $k \times k$ upper triangular matrix $M^i_{k\times k} = (a_{tj})_{k\times k}$. Based on $M^i_{k\times k}$, considering the set $J=\{j \mid a_{tj}$ is the first non-zero element of row $t\}$. From the following lemma, we know that if $j_1 \notin J, 1 \le j_1 \le k$, the j_1-th needed packet at r_i cannot be recovered.

Lemma 1. Consider n variables $x_1, x_2, \ldots, x_n$ and n equations, $M_{n\times n}$ is the coefficient matrix. After Gaussian elimination on M, $J=\{j \mid a_{tj}$ is the first non-zero element of row $t\}$, if $j_1 \notin J$, we cannot solve for x_{j_1}.

Proof. After the Gaussian elimination, M is transformed to an upper triangular matrix M'. $J=\{j \mid a_{tj}$ is the first non-zero element of row $t\}$, if $j_1 \notin J$, then a_{tj_1} is not the first non-zero element of row t, $1 \le t \le n$. That means $a_{j_1 j_1} = 0$ if M' is an upper triangular matrix.

$$\begin{pmatrix} \ddots & & & & \\ & a_{j_1-1,j_1-1} & & & \\ & & 0 & a_{j_1,j_1+1} & \\ & & & 0 & \\ 0 & & & & \ddots \end{pmatrix}$$

According to the upper triangular matrix M', we cannot solve for x_{j_1}.

With Lemma 1, we can easily identify which packet of $N(r_i)$ cannot be recovered and determine the necessary packets to be transmitted. If $j_1 \notin J, 1 \le j_1 \le k$, $p_{i_{j_1}}$ will be sent as a transmission packet from the sender. The appending process of the second stage is given in Fig. 4.

```
Q′ ← ∅;
for (i ← 1 to n) do
    l ← |P′|;
    k ← |N(r_i)|; (N(r_i) = {p_{i_1}, p_{i_2}, ..., p_{i_k}})
    Construct matrix M^i_{l×m} and M^i_{l×k}
    based on receiving vectors of r_i;
    Do Gaussian elimination on M^i_{l×k} to make
    an upper triangular matrix M^i_{k×k} = (a_tj)_{k×k};
    J ← {j|a_tj is the first non-zero element of row t};
    for (j ← 1 to k) do
        if (j ∉ J) then
            Q′ ← Q′ ∪ {p_{i_j}};
            break;
        end if
    end for
end for
```

Fig. 4. Appending algorithm for memory encoding problem

Combining results of the first stage and the second stage, we can get the encoded packets from $P' \cup Q'$, where P' is the set of encoded packets from stage 1 and Q' is the set of original packets from stage 2. Considering the above example, at the second stage, based on P', we can find out that all receivers can recover their needed packets expecting r_4. Based on the receiving vectors of r_4, v_{41}=(0,1,1,0),v_{42}= (0,1,1,0), r_4 can construct matrix M

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

After Gaussian elimination, we get the upper triangular matrix

$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

We find that J={1}. Since $N(r_4)$={p_2,p_3}, and $2 \notin J$, we get $Q' = \{p_3\}$. Finally, the set of packets to be transmitted is $P' \cup Q' = \{p_1 \oplus p_2 \oplus p_3, p_2 \oplus p_3 \oplus p_4, p_3\}$.

V. Decoding Algorithm

Similarly, each receiver can determine how to decode each needed packet by the operations of Gauss-Jordan elimination. Upon receiving l encoded packets and with its received original data packets, each receiver can construct a matrix as follows.

- Suppose that a receiver r_i receives l encoded packets, denoted by $p'_1, p'_2, \ldots, p'_l$. For each p'_j, the packet head can specify which original packets are encoded. P'_j denotes the original packets set used to encode p'_j. With such packet head information, we can construct a m-dimension vector v = (a_1,a_2,…,a_m) where a_k is 1 if $p_k \in P'_j$, otherwise a_k is 0.
- For each received original data packet p_j where $p_j \in H(r_i)$, we can also construct a m-dimension vector v=(0,…,a_j,…,0) where only a_j = 1.

As a result, r_i constructs a $(l + |H(r_i)|) \times m$ matrix M, where $|H(r_i)|$ is the cardinality of $H(r_i)$. For each row of M, we maintain the correspondent packet information in an array B.

For example, if the vector correspondent to p'_j is row k_1 of M, we set $B[k_1]$ = p'_j. Similarly, if the vector correspondent to $p_j \in H(r_i)$ is row k_2 of M, we set $B[k_2]$ = p_j.

We then conduct Gauss-Jordan elimination on M to transform M to an $m \times m$ diagonal matrix by xoring rows and exchanging rows. We record the decoding operation by changing the packet information in B correspondingly.

- If we xor row j_1 to row j_2, then $B[j_2] = B[j_2] \oplus B[j_1]$.
- If we exchange row j_1 with row j_2, we exchange the information in $B[i_1]$ with the information in $B[i_2]$.

After M is transformed to an $m \times m$ diagonal matrix, $B[j]$ records how to decode and recover needed packet p_j. For the example given in Section IV, the set of transmitted packets is $\{p'_1, p'_2, p'_3\}$, where $p'_1 = p_1 \oplus p_2 \oplus p_3$, $p'_2 = p_2 \oplus p_3 \oplus p_4$, $p'_3 = p_3$, $N(r_1)$={p_1,p_2}. After r_1 has received p'_1, p'_2, p'_3, it can construct M and B as follows:

$$M = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \ B = \begin{pmatrix} p_3 \\ p_4 \\ p'_1 \\ p'_2 \\ p'_3 \end{pmatrix}$$

After the Gauss-Jordan elimination,

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \ B = \begin{pmatrix} p'_1 \oplus p'_2 \oplus p_4 \\ p_3 \oplus p'_2 \oplus p_4 \\ p_3 \\ p_4 \end{pmatrix}$$

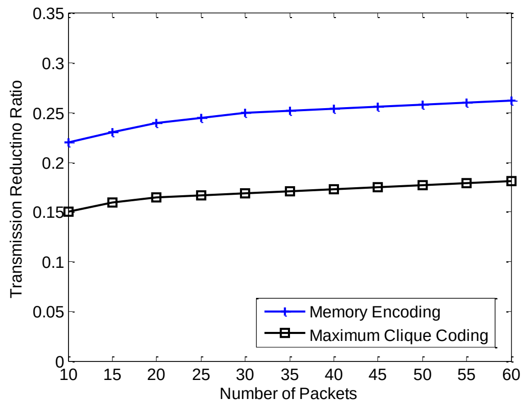
From the information in B, we know that r_1 can recover p_1 by computing $p'_1 \oplus p'_2 \oplus p_4$, and recover p_2 according to $p_3 \oplus p'_2 \oplus p_4$.

VI. Simulation Results

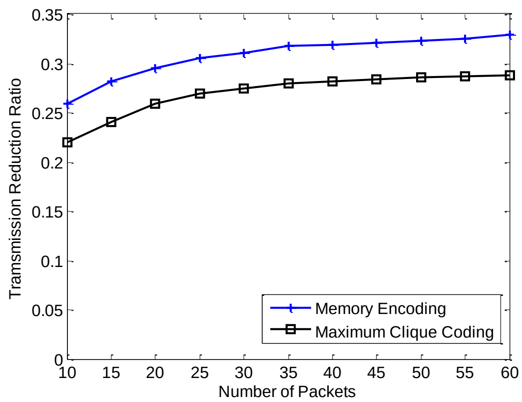
The simulation scenario consists of a sender and n receivers. The sender needs to send m packets which we denote as packet set P to n receivers, according to the overhearing or the prior transmission, every receiver has already stored some packets. The needed packets are randomly selected from the m packets with probability ρ, and the stored packets set $H(r_i) = P \setminus N(r_i), 1 \le i \le n$. We are interested in the number of transmissions which is an important performance metric in the real time applications. In order to study the impacts of number of packets m and the number of receivers n on the network coding gain, we use the transmission reduction ratio $\beta = \frac{Num_{nocoding} - Num_{coding}}{Num_{nocoding}}$ as performance metric where $Num_{nocoding}$ is the number of transmissions without network coding, and Num_{coding} is the number of transmissions with network coding. We compare transmission reduction ratio of maximum clique scheme in [8] with our memory encoding scheme.

With regard to the impacts of number of packets m and the number of receivers n on the network coding gain, similar impacts can be observed for both the case when ρ is the same for all receivers and the case when ρ is different for different receivers. We only report the case when ρ is the same for different receivers in this paper to avoid redundancy.

Fig. 5 shows the impact of number of packets m on the network coding gain which is measured by the transmission reduction ratio for n=5. In Fig. 5(a), ρ is uniformly distributed in [0.2,1]. In Fig. 5(b), ρ is uniformly distributed in [0.5,1]. As shown in Fig. 5, for fixed ρ, the probability that a needed packet of a receiver is stored by another receiver and vice versa increases with increasing *m*. The reason is that the stored packets at each receiver will be randomly distributed in a larger set of original data packets while *m* increases. Thus, the network coding gain becomes larger. Due to the same reason, if we compare the results in Fig. 5(a) with the results in Fig. 5(b), it is found that when ρ becomes larger, the network coding gain increases faster with increasing *m*.



(a) ρ in [0.2, 1]

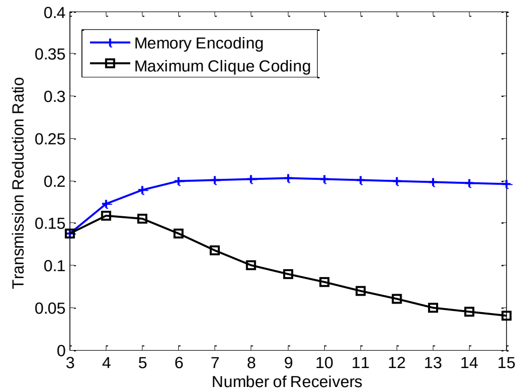


(b) ρ in [0.5, 1]

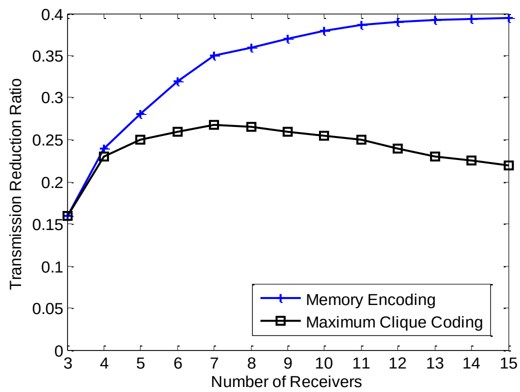
Fig. 5. Transmission reduction ratio vs. The number of packets

Fig. 6 shows the impact of the number of receivers *n* on the network coding gain. The transmission reduction ratio increases first and then decreases with increasing *n*. The probability that a needed packet of some receivers is stored by other receivers and vice versa is related to both ρ and *n*. Given ρ, the probability increased first and then decreased with increasing *n*. Thus, the network coding gain also increased first and then decreased with increasing *n*.

From Fig. 5 and Fig. 6 we can also see that using coding can reduce about 30% of transmissions. The memory encoding scheme performs better than maximum clique coding. The reason is that using memory encoding, sender does not discard the encoded packets which cannot be decoded immediately, and leaves it for later use. Thus sender can further reduce the number of transmissions, which is an important performance metric in real time applications.



(a) ρ in [0.2, 1]



(b) ρ in [0.5, 1]

Fig. 6. Transmission reduction ratio vs. The number of receivers

VII. Conclusions

In this paper, we focus on network coding based broadcast scheduling in wireless networks with memory model and aim at minimizing the number of transmission packets. According to a two stage code construction based on graph model and matrix transformation, effective heuristic algorithm is proposed in this paper. Simulation results show that our algorithm significantly reduce the number of transmissions in most cases, which is an important performance metric in real time applications. The network coding gain of memory model excels IDNC scheme which is consistently better than the no coding scheme.

Acknowledgment

This work was supported by the Fundamental Research Funds for the Central Universities (No. SWU115002, No. XDJK2015C104).

References

[1] C. Fragouli, J. Widmer, and J. Le Boudec, "Efficient broadcasting using network coding," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 450-463, 2008.

[2] P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive network coding for broadcast channels," in *Proc. NetCod*, Lausanne, 2009, pp. 80-85.

[3] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Tran. Vehicular Tech.*, vol. 58, no. 2, pp. 914-925, Feb. 2009.

[4] S. Sorour and S. Valaee, "Completion delay minimization for instantly decodable network coding with limited feedback," in *Proc. ICC*, Kyoto, 2011, pp. 1-5.

[5] N. Aboutorab, P. Sadeghi, and S. Sorour, "Enabling a tradeoff between completion time and decoding delay in instantly decodable network coded systems," *IEEE Trans. Commun.*, vol. 62, no. 4, pp. 1296-1309, Mar. 2014.

[6] K. S. Kim, C. P. Li, and E. Modiano, "Scheduling multicast traffic with deadlines in wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 2193-2201.

[7] A. Le, L. Keller, C. Fragouli, and A. Markopoulou, "MicroPlay: A networking framework for local multiplayer games," in *Proc. SIGCOMM MobiGames Workshop*, Helsinki, 2012, pp. 155-160.

[8] A. Le, A. S. Tehrani, A. G. Dimakis, and A. Markopoulou, "Instantly decodable network codes for real-time applications," in *Proc. International Symposium on Network Coding* (*NetCod*), 2013, pp. 1-6.

[9] M. Heindlmaier and C. Blochl, "The two-user broadcast packet erasure channel with feedback and memory," in *Proc. IEEE Workshop on Network Coding, Theory, and Applications* (*NetCod*), 2014.

[10] N. J. A. Harvey, D. R. Karger, and S. Yekhanin, "The complexity of matrix completion," in *Proc. SIAM-ACM Symp. Discrete Algorithms* (*SODA*), 2006, pp. 1103-1111.

Cheng Zhan was born in ChongQing, China, in 1985. He received the Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, in 2011. He is currently in the School of Computer and Information Science, Southwest University. His research interests include network coding, wireless network, and distributed storage.

Fuyuan Xiao was born in Guangxi Province, China, in 1986. She received the Ph.D. degree from the Kumamoto University, Japan, in 2014. She is currently in the School of Computer and Information Science, Southwest University. Her research interests include optimization algorithm, data analysis, and vehicular network.