

# Network Coding for Mobile Devices - Systematic Binary Random Rateless Codes

Janus Heide

Dept. of Electronic Systems

Aalborg University

Email: speje@es.aau.dk

Morten V. Pedersen

Dept. of Electronic Systems

Aalborg University

Email: mvpe@es.aau.dk

Frank H. P. Fitzek

Dept. of Electronic Systems

Aalborg University

Email: ff@es.aau.dk

Torben Larsen

Dept. of Electronic Systems

Aalborg University

Email: tl@es.aau.dk

**Abstract**—In this work we consider the implementation of Random Linear Network Coding (RLNC) on battery constrained mobile devices with low computational capabilities such as sensors, mobile phones and Personal Digital Assistants (PDAs). It is non-trivial to create an efficient implementation of RLNC which is needed to ensure high throughput, low computational requirements and energy consumption. As a consequence there does not, to the best of our knowledge, exist any such implementation for mobile device that allow for throughput close to what can be achieved in e.g. Wireless Local Area Network (WLAN).

In this paper we propose to base RLNC on the binary Galois field and to use a systematic code. We have implemented this approach in C++ and Symbian C++ and achieve synthetic encoding/decoding throughput of up to 40/30 MB/s on a Nokia N95-8GB mobile phone and 1.5/1.0 GB/s on a high end laptop.

**Index Terms**—Mobile devices, Network coding, Reliable Multicast.

## I. INTRODUCTION

A large body of existing literature [1] treats the theoretical benefits of Network Coding (NC). However, the costs of implementing NC in terms computational overhead, memory consumption or network usage is often not considered. In this work we consider the implementation of RLNC on mobile battery constrained devices with low computational capabilities, such as sensors, mobile phones or PDAs. The computations performed using RLNC is based on finite fields arithmetic also known as Galois fields. From a coding perspective the field size,  $q$ , used should be large to ensure that coded packets are linearly independent, additionally increasing the size of the field elements is advantageous as this reduces the number of operations needed to code a certain amount of data [2]. However as the field size grows it becomes difficult to construct an efficient implementation of the necessary operations. Although the attention towards practical implementation of NC has been increased [3], [4], [5], some issues remains to be solved in order to pave the way for successful deployment of NC.

In [6], [7] implementation problems are investigated with focus on the computational complexity of coding operations. An implementation using log and anti-log tables is constructed and evaluated. Throughput up to 11 MB/s is measured on a 3.6 GHz dual core Intel P4 Central Processing Unit (CPU), but performance decreases rapidly as the number of packets that are coded together increases. Different optimization techniques are described and incorporated. Generated network

overhead as a consequence of utilizing NC is commented and sought to be minimized. Furthermore the concepts of *aggressiveness* and *density* and its impact of performance are presented. In [8] a coding throughput of 44Mb/s is measured using an implementation based on a simple full-size look-up table. This is achieved on an 800 MHz Intel Celeron CPU when 32 packets,  $g=32$ , of 1500 B are coded together. The observed throughput is approximately 10 times higher than that reported in [6] at similar settings. The authors conclude that deployment is not a problem, however it is mentioned that throughput for  $g=32$  is the best case, unfortunately no performance for  $g>32$  is presented. In [9] tables are not used for Galois field multiplication, instead the authors implement a loop based approach. In combination with Single Instruction, Multiple Data (SIMD) instructions this approach provides a performance increase of over 500% compared to a baseline implementation. The implementation is further optimized through parallelization and encoding speeds up to 43 MB/s is measured on a 2.8 GHz quad core Intel P4 CPU. Although these results show promising coding throughputs most were achieved at maximum CPU utilization, which is not acceptable on non-dedicated machines. Additionally these results are not in general valid for mobile phones or wireless sensors, as the hardware resources available on such devices are considerably lower than those of the desktop computers used to obtain these results. In [2] we present the results of a basic log and anti-log table implementation running on a Nokia N95 with a 332 MHz ARM 11 CPU. We achieved the maximum coding throughput of 117 KB/s for a  $GF(2^{16})$ , which indicates that before NC can be deployed on mobile devices, more efficient algorithms or hardware acceleration is needed.

In this work we propose to use the binary field to reduce the computational complexity of RLNC and to use a systematic [10] approach to reduce the amount of coding needed. Furthermore we provide an analysis of this approach for different generation sizes and compare its performance to network coding with a higher field size.

This work is organized in the following sections. Section III describes the scenario under investigation. Section II presents an analysis of the proposed network coding approach. In Section IV we introduce our implementation using  $GF(2)$  and coding throughputs obtained with this implementation. The final conclusions are drawn in Section V.

## II. SCENARIO & SOLUTION

We consider a scenario where a source  $s$  wants to reliably transmit the same data to one or more nearby sinks  $t_1, t_2, \dots, t_N$  via a wireless link. This basic scenario is given in Figure 1.

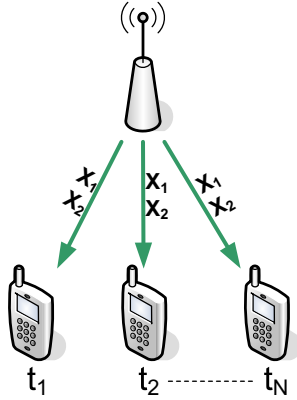


Fig. 1: One source  $s$  transmitting data to  $N$  receivers  $t_1, t_2, \dots, t_N$ .

As all receivers are requesting the same data, broadcast provides an efficient utilization of the wireless channel, as all packets are delivered to all nodes simultaneously. To ensure that the links are reliable some form of retransmission is required, to correct packet losses at the individual nodes. In current networks this is done by letting the individual nodes acknowledge received packets. Thus retransmissions of multiple different packet sets are required unless all nodes lose exactly the same packets.

To ensure reliability with a low overhead we will instead use RLNC to correct packet errors. This is done by transmitting the data in two stages, in the first stage the source,  $s$ , transmits all packets uncoded. This makes sense as all packets received by the individual nodes will contain useful new information. In the second stage we wish to correct packet losses which have occurred during the first stage. Due to the uncorrelated nature of packet losses the nodes will now hold disjoint sets of packets. Therefore to maximize the number of nodes for which a packet is useful, the source will create and send random linearly combinations of the original data. By using this approach one coded packet carries information which can potentially correct different errors at different nodes simultaneously. To retrieve the full data set a node now has to receive as many linear independent coded packets as it lost during the first stage.

## III. ANALYSIS

In this section we analyze single-source multiple-sinks reliable transmission using Markov chains. The main objective is to determine the expected number of transmissions  $E[t_x]$  needed for transmitting a packet from a source  $s$  to  $N$  sinks  $t_1, t_2, \dots, t_N$ . We assume an i.i.d. Packet Error Probability (PEP)  $p$  and consider unicast, broadcast, pure and systematic

network coding. We note that the analysis also holds for unicast and RLNC when the channels are not independent, but that broadcast will perform better if the erasures are correlated. Unicast is not designed for this type of transmission however it is interesting as a reference.

The state machine of a single sink that receive one packet is illustrated on Figure 2. Either the node has received the packet and is in state zero, or it has not received the packet and is in state one. The number of the state thus indicates the number of erroneous or missing packets.

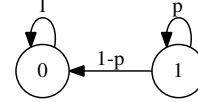


Fig. 2: Markov chain for a single node receiving a single packet

In the transmission matrix state zero corresponds to column zero and state one to column one. When  $s$  transmits the packet  $t_1$  will receive it with probability  $1 - p$  and not receive it with probability  $p$ . This yields the transition matrix  $\mathbf{T}$ . When transmission commences the node have not received the packet and thus is in state one. It is convenient to define a matrix that indicates the starting probabilities,  $\mathbf{S}$ .

$$\mathbf{T} = \begin{bmatrix} 1 & 0 \\ 1-p & p \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

### A. Unicast

The probability distribution of a sink after  $i$  transmissions from  $s$  is given by the last row in  $\mathbf{P}^i$ , in this case row one denoted by  $\mathbf{P}_{(1,:)}^i$ , where.

$$\mathbf{P}^i = \mathbf{S} \times \mathbf{T}^i \quad (1)$$

The probability that the sink has received the packet after  $i$  transmissions is  $\mathbf{P}_{(1,0)}^i$  which is the index row one, column zero in  $\mathbf{P}^i$ . Thus the probability that the sink has not received the packet is  $1 - \mathbf{P}_{(1,0)}^i$ . In unicast transmission is performed serially and thus we multiply with the number of sinks  $N$ .

$$E[t_x] = N \cdot \sum_{i=0}^{\infty} 1 - \mathbf{P}_{(1,0)}^i \quad (2)$$

### B. Broadcast

In broadcast the probability distribution for one node is the same as for unicast but all nodes receive packets in parallel. Thus the probability that all sinks have received the packet after  $i$  transmissions is the probability that one node have received the packet to the power of the number of sinks,  $(\mathbf{P}_{(1,0)}^i)^N$ .

$$E[t_x] = \sum_{i=0}^{\infty} 1 - \left( \mathbf{P}_{(1,0)}^i \right)^N \quad (3)$$

### C. Pure Network Coding

In network coding data to be transferred from the source to the sinks is divided into packets of length  $m$ . The number of original packets over which encoding is performed is typically referred to as the batch size or generation size and denoted  $g$ . Thus the  $g$  original data packets of length  $m$  are arranged in the matrix  $\mathbf{M} = [\mathbf{m}_1 \mathbf{m}_2 \dots \mathbf{m}_g]$ , where  $\mathbf{m}_i$  is a column vector.

In pure network coding to generate one coded data packet  $\mathbf{x}$ ,  $\mathbf{M}$  is multiplied with a randomly generated vector  $\mathbf{g}$  of length  $g$ ,  $\mathbf{x} = \mathbf{M} \times \mathbf{g}$ . In this way we can construct  $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_{g+r}]$  that consists of  $g+r$  coded data packets and  $\mathbf{G} = [\mathbf{g}_1 \mathbf{g}_2 \dots \mathbf{g}_{g+r}]$  that contains  $g+r$  randomly generated encoding vectors, where  $r$  is redundant packets. In order for a sink to successfully recreate the original data packets, it must receive  $g$  linear independent coded packets and encoding vectors. All received coded packets are placed in the matrix  $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1 \hat{\mathbf{x}}_2 \dots \hat{\mathbf{x}}_g]$  and all encoding vectors are placed in the matrix  $\hat{\mathbf{G}} = [\hat{\mathbf{g}}_1 \hat{\mathbf{g}}_2 \dots \hat{\mathbf{g}}_g]$ . The original data  $\mathbf{M}$  can then be decoded as  $\hat{\mathbf{M}} = \hat{\mathbf{X}} \times \hat{\mathbf{G}}^{-1}$ .

All operation are performed over a Galois field of size  $q$ . Thus the probability that the source generates a linear dependant combination becomes an important factor. This probability depends on  $q$  and  $g' = g - \tilde{g}$ , where  $\tilde{g}$  is the number of linear independent packets received by the sink and  $g'$  thus is the number of needed packets at the sink. The following bound for linear independence is assumed in an alternative form in [11], [12] and is said to hold when  $q$  is high .

$$P \leq 1 - \frac{1}{q^{g'}} \quad (4)$$

We provide the following intuitive interpretation, where  $\tilde{\mathbf{G}}$  is a matrix of dimension  $\tilde{g} \times g$  that contains all received linear independent encoding vectors at the sink. The problem of drawing an encoding vector  $\mathbf{g}_i$  that is linear independent of all other  $\tilde{g}$  linear independent rows in  $\tilde{\mathbf{G}}$  is equal to drawing a linear independent combination of length  $g'$ . This is because the degrees of freedoms of any  $\mathbf{g}_i$  can be reduced to at most  $g'$ , alternatively at most  $g'$  indices of any  $\mathbf{g}_i$  are non zero when all pivot elements in  $\tilde{\mathbf{G}}$  is subtracted from  $\mathbf{g}_i$ . For the remaining sequence of length  $g'$  to be dependent it must consist of all zeros which have the probability  $\frac{1}{q^{g'}}$ .

However we consider cases where  $q$  is low and thus we need to estimate to what extend the bound holds for low values of  $q$ . To achieve this we have generated a large number, 100.000, square matrices of dimension  $g$  consisting of Galois elements with  $GF(2)$  and  $GF(2^8)$  and tested how many of these was linear independent. For given  $q$  and  $g$  the probability that a generated combination is linear dependent can be written as:

$$1 - \prod_{i=1}^g \left(1 - \frac{1}{q^i}\right) \quad (5)$$

Although the results in Table Ib does not necessarily guarantee equality the empirically obtained values is very close

$g$	calculated	empirical	$g$	calculated	empirical
2	62.5	62.6	2	3.92E-3	3.99E-3
4	69.2	69.3	4	3.92E-3	3.67E-3
8	71.0	70.9	8	3.92E-3	3.87E-3
16	71.1	71.2	16	3.92E-3	3.78E-3
32	71.1	71.0	32	3.92E-3	3.84E-3
64	71.1	71.1	64	3.92E-3	3.88E-3

(a)  $GF(2)$

(b)  $GF(2^8)$

TABLE I: Calculated and empirical determined probability of generating a linear dependent matrix of size  $g$ .

to the calculated and therefore this approximation, if any, is acceptable in the following analysis.

The state machine of a single sink receiving  $g$  packets has  $g+1$  states depicted in Figure 3.

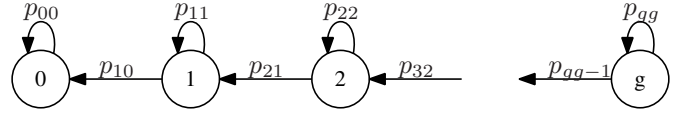


Fig. 3: Markov Chain for a single node that receives  $g$  packets.

The probability that a packet is useful at a sink is the probability it is received multiplied with the probability that it is independent

$$P_{i \rightarrow (i-1)} = (1-p) \left(1 - \frac{1}{q^i}\right)$$

The probability that it is not useful is therefore the probability that it was not received plus the probability that it was received but was dependent.

$$\begin{aligned} P_{i \rightarrow i} &= 1 - (1-p) \left(1 - \frac{1}{q^i}\right) = 1 - 1 + \frac{1}{q^i} + p - p \frac{1}{q^i} \\ &= p + (1-p) \frac{1}{q^i} \end{aligned}$$

These probabilities form the transition matrix  $\mathbf{C}$ .

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ (1-p)(1 - \frac{1}{q^1}) & p + (1-p)\frac{1}{q^1} & & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & (1-p)(1 - \frac{1}{q^g}) & p + (1-p)\frac{1}{q^g} \end{bmatrix}$$

TABLE II: Transmission matrix for coding.

$$\mathbf{Q}^i = \mathbf{S} \times \mathbf{C}^i \quad (6)$$

Thus the expected number of transmissions for one packet of the total  $g$  packets can be found by:

$$E[t_x] = \frac{1}{g} \sum_{i=0}^{\infty} 1 - \left(\mathbf{Q}_{(g,0)}^i\right)^N \quad (7)$$

#### D. Systematic Network Coding

Systematic RLNC consists of two phases. In the first phase all packets  $g$  in the generation are broadcasted uncoded. In the second phase coded packets are broadcasted and thus each node can be in  $g + 1$  one states, where state zero indicates that no packets are missing and state  $g$  that all packets in the generation are missing. Uncoded packets can be perceived as coded packets with a trivial encoding vector where a single element in the encoding vector  $\mathbf{g}_i$  is one and all other,  $g - 1$ , elements is zero. Thus we can generate an uncoded packet  $\mathbf{y}$  from its trivial encoding vector  $\mathbf{h}$ ,  $\mathbf{y} = \mathbf{M} \times \mathbf{h}$ . In this way we can construct  $\mathbf{Y} = [\mathbf{y}_1 \mathbf{y}_2 \dots \mathbf{y}_g]$  that consists of  $g$  uncoded data packets and  $\mathbf{H} = [\mathbf{h}_1 \mathbf{h}_2 \dots \mathbf{h}_g]$  that contains the  $g$  independent trivial encoding vectors. Furthermore we construct  $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_r]$  that consists of  $r$  coded data packet and  $\mathbf{G} = [\mathbf{g}_1 \mathbf{g}_2 \dots \mathbf{g}_r]$  that contains  $r$  randomly generated encoding vectors. For a sink to successfully recreate the original data packets, it must receive  $g$  linear independent packets and encoding vectors. Thus  $g$  received uncoded and coded packets are placed in the matrix  $[\hat{\mathbf{Y}}\hat{\mathbf{X}}] = [\hat{\mathbf{y}}_1 \hat{\mathbf{y}}_2 \dots \hat{\mathbf{x}}_{(g-i)} \hat{\mathbf{x}}_1 \hat{\mathbf{x}}_2 \dots \hat{\mathbf{x}}_i]$  and the  $g$  corresponding encoding vectors are placed in the matrix  $[\hat{\mathbf{H}}\hat{\mathbf{G}}] = [\hat{\mathbf{h}}_1 \hat{\mathbf{h}}_2 \dots \hat{\mathbf{h}}_{(g-i)} \hat{\mathbf{g}}_1 \hat{\mathbf{g}}_2 \dots \hat{\mathbf{g}}_i]$ . The original data  $\mathbf{M}$  can then be decoded as  $\mathbf{M} = [\hat{\mathbf{Y}}\hat{\mathbf{X}}] \times [\hat{\mathbf{H}}\hat{\mathbf{G}}]^{-1}$ .

In phase one the transition matrix for a single sink is identical to that of broadcast, however here we consider the transmission of  $g$  packets instead of one packet, thus the transition matrix have  $g + 1$  states:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ (1-p) & p & & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & (1-p) & p \end{bmatrix}$$

TABLE III: Transmission matrix for broadcasting.

The probability distribution of the first phase is the input to the transition matrix of the second phase and thus the probabilities in any of the two phases can be calculated as:

$$\mathbf{R}^i = \begin{cases} \mathbf{S} \times \mathbf{T}^i & \text{for } 0 \leq i \leq g \\ \mathbf{S} \times \mathbf{T}^g \times \mathbf{C}^i & \text{for } g < i \end{cases} \quad (8)$$

Thus the expected number of transmissions can be found by:

$$E[t_x] = \frac{1}{g} \sum_{i=0}^{\infty} 1 - \left( \mathbf{R}_{(g,0)}^i \right)^N \quad (9)$$

We note that the performance of systematic network coding is equal or better than that of pure network coding. For the trivial coded packets in systematic coding the probability of linear dependencies is zero, for pure network coding the probability is non-zero but very small, see Equation 5. For the non-trivially coded packets the probability of linear independence is identically for systematic and pure network coding.

#### E. Results

Here we compare the performance of RLNC with unicast and broadcast. We assume  $p=0.3$  as we have previously observed such  $p$  [13]. In Figure 4 the performance of unicast, broadcast and network coding at different settings is plotted for an increasing number of sinks.

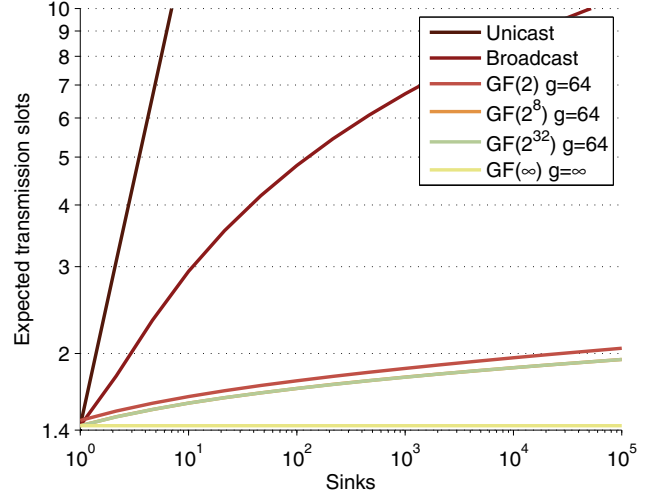


Fig. 4: Expected number of transmission per packet,  $p = 0.3$ .

Unicast is not designed to perform this type of transmission and therefore it is not surprising that it performs the worst. Broadcast performs better, however, as the number of sinks increases it suffers from the fact that all sinks must receive *all* original packets and thus retransmissions of packets become necessary. For RLNC this is not the case, instead all sinks only have to receive a number of *any* independent coded packets and can then decode the original data. When  $g$  is fixed RLNC with GF(2) performs the worst of the RLNC approaches and GF(2<sup>8</sup>) and GF(2<sup>32</sup>) performs the same, therefore we do not consider GF(2<sup>32</sup>) in the following.

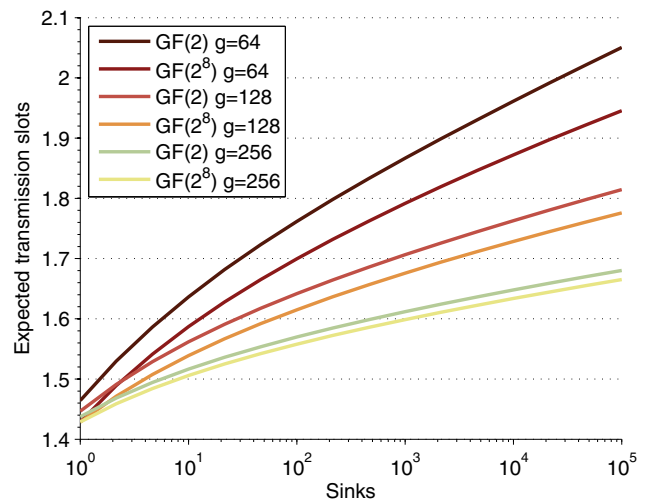


Fig. 5: Expected number of transmission per packet,  $p = 0.3$ .

In Figure 5 we see that network performance improves when either the field or generation size is increased. When the field size increases the probability of linear dependency decreases and when the generation size increase the uncertainty of how many packets will be lost at each sink decreases. Generally the benefit of doubling  $g$  is greater than going from GF(2) to GF( $2^8$ ) and additionally the performance benefit of moving from GF(2) to GF( $2^8$ ) decreases as  $g$  increases. The reason is that for high values of  $g$  linear dependency becomes less important because only the very last packet will have a low probability of being linear independent. Thus when  $g$  is increased the ratio of packet that have low probability of being linear independent decreases.

Thus if the throughput that can be achieved with GF(2) allows for a substantially higher value of  $g$  compared to an implementation with GF( $2^8$ ), it may be beneficial to use the small field in some cases.

#### IV. IMPLEMENTATION

To get a feel for the achievable encoding and decoding throughput we implemented RLNC based on GF(2). In GF(2) adding two packets simplifies to the XOR operation. Encoding a packet in GF(2) can be performed in two simple steps. First the encoding vector is generated as a random bit vector, where the indices in the vector corresponds to packets in the original data set i.e. index one corresponds to packet one. The second step is performed by iterating over the encoding vector and adding packets where the corresponding index in the encoding vector is 1. The following listing shows the encoding algorithm in pseudo code, where  $\mathbf{M}$  is the data buffer containing all original packets,  $\mathbf{g}$  is an encoding vector and  $\mathbf{x}$  is the resulting encoded packet.

```

1: procedure ENCODEPACKET( $\mathbf{M}, \mathbf{x}, \mathbf{g}$ )
2:    $\mathbf{x} = \mathbf{0}$ 
3:   for each bit  $b$  in  $\mathbf{g}$  do
4:     if  $b$  equal 1 then
5:        $i =$  position of  $b$  in  $\mathbf{g}$ 
6:        $\mathbf{x} = \text{XOR}(\mathbf{x}, \mathbf{M}[i])$ 
7:     end if
8:   end for
9: end procedure

```

Decoding is performed on the run in two steps with a slightly modified Gauss-Jordan algorithm. Note that this approach is different from what is typically done in implementations for higher field sizes where the encoding matrix is inverted and then subsequently multiplied with the data matrix. Thus the received data at the sink is always decoded as much as possible and the load on the CPU is distributed evenly. In the first step we reduce the incoming encoded packet by performing a forward substitution of already received packets. This is done by inspecting the elements of the encoding vector from start to end and thus determining which original packets the coded packet is a combination of. If an element is 1 and we have already identified a packet with this element as a pivot element we subtract that packet from the coded packet and continue the inspection. If an element is 1 and we have

not already identified a packet where this element is a pivot element we have identified a pivot packet and continue to the second stage of the decoding. Note that if we are able to subtract all information contained in the received encoded packet, it will contain no information useful to us and can be discarded.

In the second step we perform backward substitution with the newly identified pivot packet. We do this by subtracting the pivot packet from previously received packets for which the corresponding encoding vector indicates that the particular packet is a combination of the pivot packet. The following listing shows the decoding algorithm in pseudo code, where  $\hat{\mathbf{M}}$  is the packet decode buffer of packets received and decoded so far and  $\hat{\mathbf{G}}$  is the corresponding encoding vector buffer,  $\hat{\mathbf{x}}$  is a newly received encoded packet and  $\hat{\mathbf{g}}$  is the newly received encoding vector.

```

1: procedure DECODEPACKET( $\hat{\mathbf{M}}, \hat{\mathbf{G}}, \hat{\mathbf{x}}, \hat{\mathbf{g}}$ )
2:   pivotposition = 0
3:   pivotfound = false
4:   for each bit  $b$  in  $\hat{\mathbf{g}}$  do ▷ Forward Substitution
5:     if  $b$  equal 1 then
6:        $i =$  position of  $b$  in  $\hat{\mathbf{g}}$ 
7:       if  $i$ 'th packet is in  $\hat{\mathbf{M}}$  then
8:          $\hat{\mathbf{g}} = \text{XOR}(\hat{\mathbf{g}}, \hat{\mathbf{G}}[i])$ 
9:          $\hat{\mathbf{x}} = \text{XOR}(\hat{\mathbf{x}}, \hat{\mathbf{M}}[i])$ 
10:      else
11:        pivotfound = true
12:        pivotposition =  $i$ 
13:      end if
14:    end if
15:  end for
16:  if pivotfound equal false then
17:    Exit procedure ▷ The packet was linear dependant
18:  end if
19:  for each packet  $j$  in  $\hat{\mathbf{M}}$  do ▷ Backward Substitution
20:     $k = \hat{\mathbf{G}}[j]$ 
21:    if bit at pivotposition in  $k$  equal 1 then
22:       $\hat{\mathbf{G}}[j] = \text{XOR}(\hat{\mathbf{G}}[j], \hat{\mathbf{g}})$ 
23:       $\hat{\mathbf{M}}[j] = \text{XOR}(\hat{\mathbf{M}}[j], \hat{\mathbf{x}})$ 
24:    end if
25:  end for
26:   $\hat{\mathbf{G}}[\text{pivotposition}] = \hat{\mathbf{g}}$ 
27:   $\hat{\mathbf{M}}[\text{pivotposition}] = \hat{\mathbf{x}}$ 
28: end procedure

```

The algorithm can also be used unmodified in a systematic coding approach, in which case we only have to ensure that uncoded packets are treated as pivot packets. Based on these algorithms we have implemented a coding library designed to deliver high throughput and optimized through assembly and SIMD instructions. Subsequently the implementation was ported to Symbian to allow for tests on a mobile platform. All implementations are single threaded. We used the following platforms for the tests.

- 1) Nokia N95-8GB, ARM 11 332 MHz CPU, 128 MB ram, Symbian OS 9.2.

2) Lenovo T61p, 2.53 GHz Intel Core2Duo, 2 GB ram, Kubuntu 8.10 64bit.

We tested the performance of the implementation by encoding and decoding without transmission over any network at different generation sizes from 16 to 256. First a large file, 5 MB for the phone and 128 MB for the PC, was divided into packets of 1500 Bytes. These packets were then split into generations of size equal to the specified generation size. From each of these generations packets were generated and saved. Then packets from each generation were read and decoded until the original data was recreated. Encoding and decoding time were measured and from this the throughput was calculated.

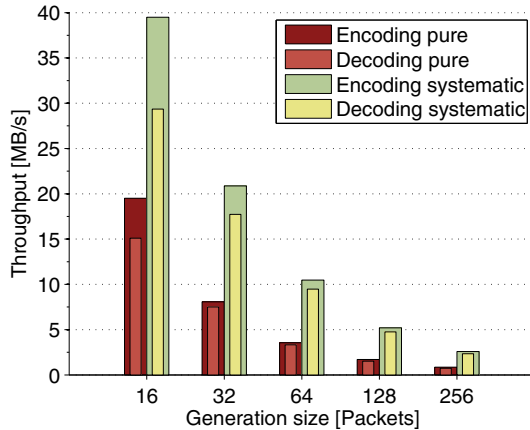


Fig. 6: Encoding and Decoding throughput on a mobile phone.

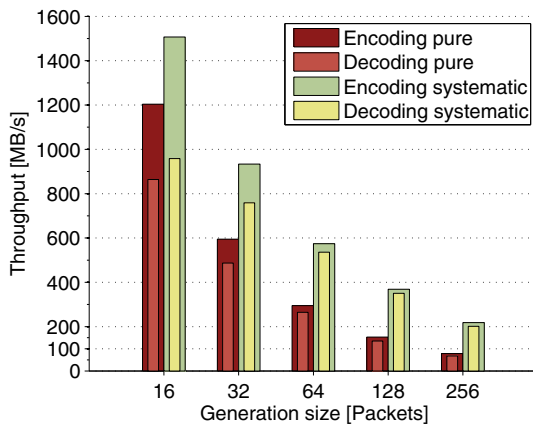


Fig. 7: Encoding and Decoding throughput on a laptop.

Both encoding and decoding throughput is considerably higher than any other reported result known to the authors and the throughput is approximately a first order function of the generation size. Note that for the systematic approach 70% of the packets were uncoded and 30% coded. In a real wireless scenario the ratio of coded vs. uncoded packets depends on the error probability of the link. In the pure approach all packets are coded, thus the throughput for the pure approach is equal to the worst case throughput of the systematic approach.

## V. CONCLUSION

In this paper we have proposed to base RLNC on the binary Galois field in order to decrease the computational complexity. Additionally we have proposed techniques for reducing the amount of coding needed, which can help to increase throughput and decrease energy consumption of NC implementations. The proposed approach have been analyzed from a network point of view. The approach have been implemented and we have demonstrated that high encoding and decoding throughputs can be achieved on both mobile phones and laptops.

## VI. ACKNOWLEDGEMENT

We would like to thank Ralf Koetter for an interesting discussion in the Biergarten and Muriel Medard for the discussion on network coding and cooperative wireless networks. Furthermore, we would like to thank Nokia for providing technical support and mobile phones. Special thanks to Mika Kuulusa, Gerard Bosch, Harri Pennanen, Nina Tammelin, and Per Moeller from Nokia. This work was partially financed by the X3MP project granted by Danish Ministry of Science, Technology and Innovation.

## REFERENCES

- [1] "The network coding home page." [https://hermes.lnt.e-technik.tu-muenchen.de/DokuWiki/doku.php?id=network\\_coding:bibliography\\_for\\_network\\_coding](https://hermes.lnt.e-technik.tu-muenchen.de/DokuWiki/doku.php?id=network_coding:bibliography_for_network_coding). Extensive (250+) but incomplete list of publications related to Network Coding.
- [2] J. Heide, M. V. Pedersen, F. H. Fitzek, and T. Larsen, "Cautious view on network coding - from theory to practice," *Journal of Communications and Networks (JCN)*, 2008.
- [3] J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Medard, "Codecast: a network-coding-based ad hoc multicast protocol," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 13, no. 5, pp. 76–81, October 2006.
- [4] D. Nguyen, T. Nguyen, and B. Bose, "Wireless broadcasting using network coding," in *Third Workshop on Network Coding, Theory, and Applications*, January 2007.
- [5] L. Popova, A. Schmidt, W. Gerstacker, and W. Koch, "Network coding assisted mobile-to-mobile file transfer," in *Australasian Telecommunication Networks and Applications Conference (ATNAC)*, December 2007.
- [6] M. Wang and B. Li, "How practical is network coding?," *Quality of Service, 2006. IWQoS 2006. 14th IEEE International Workshop on*, pp. 274–278, June 2006.
- [7] M. Wang and B. Li, "Lava: A reality check of network coding in peer-to-peer live streaming," in *INFOCOM*, pp. 1082–1090, 2007.
- [8] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 169–180, 2007.
- [9] H. Shojania and B. Li, "Parallelized progressive network coding with hardware acceleration," *Quality of Service, 2007 Fifteenth IEEE International Workshop on*, pp. 47–55, June 2007.
- [10] M. Xiao, T. Aulin, and M. Médard, "Systematic binary deterministic rateless codes," in *Proceedings IEEE International Symposium on Information Theory*, 2008.
- [11] D. E. Lucani, M. Stojanovic, and M. Médard, "Random linear network coding for time division duplexing: When to stop talking and start listening," *CoRR*, vol. abs/0809.2350, 2008. informal publication.
- [12] A. Eryilmaz, A. Ozdaglar, and M. Medard, "On delay performance gains from network coding," *Information Sciences and Systems, 2006 40th Annual Conference on*, pp. 864–870, March 2006.
- [13] J. Heide, M. V. Pedersen, F. H. Fitzek, T. V. Kozlova, and T. Larsen, "Know your neighbour: Packet loss correlation in IEEE 802.11b/g multicast," in *The 4th International Mobile Multimedia Communications Conference (MobiMedia '08)*, (Oulu, Finland), July 7-9 2008.