A Mobile Application Prototype using Network Coding

Pedersen, Morten Videbæk; Heide, Janus; Fitzek, Frank Hanns Paul; Larsen, Torben

Published in:
European Transactions on Telecommunications

DOI (link to publication from Publisher):
10.1002/ett.1448

Publication date:
2010

Document Version
Early version, also known as pre-print

EUROPEAN TRANSACTIONS ON TELECOMMUNICATIONS
Eur. Trans. Telecomms. 2010; **21**:738–749
Published online 11 October 2010 in Wiley Online Library
(wileyonlinelibrary.com) DOI: 10.1002/ett.1448

A mobile application prototype using network coding†

Morten V. Pedersen*, Janus Heide, Frank H. P. Fitzek and Torben Larsen

Department of Electronic Systems, Aalborg University, Denmark

SUMMARY

This paper looks into implementation details of network coding for a mobile application running on commercial mobile phones. We describe the necessary coding operations and algorithms that implements them. The coding algorithms forms the basis for a implementation in C++ and Symbian C++. We report on practical measurement results of coding throughput and energy consumption for a single-source multiple-sinks network, with and without recoding at the sinks. These results confirm that network coding is practical even on computationally weak platforms, and that network coding potentially can be used to reduce energy consumption. Copyright © 2010 John Wiley & Sons, Ltd.

1. INTRODUCTION

Network Coding (NC) has received a lot of attention since the term was coined by Ahlswede *et al*. [1]. Several research works have investigated [2, 3] and implemented [4, 5] NC to prove the feasibility of this novel technique. NC can be applied in many communication scenarios such as multicast or meshed networking, where NC delivers promising results for throughput and reliability. While most codes are end-to-end, with NC packets can be recoded at each node in the network, which can be of special interest in multi-hop networks.

The concept of NC has been proven to work in theory, some current questions are how to design NC algorithms and whether these algorithms are too complex for a given platform. In References [6, 7], it has been shown that NC can be applied to sensor networks and meshed networks formed by mobile phones. One finding was that NC techniques must be designed with care if they are to be applied to the mobile or embedded domain. These platforms have limited resources such as energy, memory and computational power in addition to the general problems in mobile networking such as limited wireless capacity.

This paper introduces a mobile demo application using NC that is running on the Symbian/S60 platform used on most Nokia smartphones and by other manufactures such as Motorola, Samsung and Sony Ericsson. The main idea is that users wish to share content over short range wireless technologies such as WiFi. Instead of uploading the content to social networks such as MySpace or Facebook, the content can be conveyed directly to nearby mobile phones, which would allow a user to easily share photos with his/her friends ad hoc.

The use of NC is motivated by the fact that the transmission from one source to many sinks must be done in a reliable and efficient manner. NC enables this as it allows for efficient spectrum usage and a low complexity error control system. NC can be applied at different protocol layers, ranging from the physical layer over the network layer to the application layer. In this work we focus on the application layer. Furthermore, the paper provides some implementation guidance on how to keep the complexity of NC low.

* Correspondence to: Morten V. Pedersen, Department of Electronic Systems, Aalborg University, Denmark. E-mail: mvp@es.aau.dk
† A previous version of this paper was presented in the 15th European Wireless Conference (EW 2009), Aalborg, Denmark.

Received 30 November 2009
Revised 12 July 2010
Accepted 14 July 2010

The remainder of this work is organised in the following sections. Section 2 introduces different transmission approaches. Section 3 introduces NC operations and algorithms. In Section 4 the functionality and interface of the application prototype is introduced. Section 5 presents the obtained results. Section 6 provides a discussion on important considerations when implementing NC. The conclusion is presented in Section 7.

2. TRANSMISSION APPROACHES

Different approaches for transmitting the data are possible, here we present some possibilities. We assume that a single source s broadcast data to N sinks $t_1 \ldots t_N$ and that the source has a direct wireless link to the sinks, as shown in Figure 1. The data can be divided into a number of packets, g. Transmitting packets over the wireless link may lead to packet loss due to the characteristics of the wireless channel thus an error control system is needed.

2.1. Unicast

The simplest solution is for the source to send the data in a round robin fashion using a reliable unicast protocol e.g. Transmission Control Protocol (TCP). Such an approach is fully reliable as each sink is served individually. Each sink acknowledges received packets and therefore the source device can determine when all sinks have received all packets. This solution is simple and the computational complexity is low. However, if N is high the amount of redundant information sent from the source becomes significant.

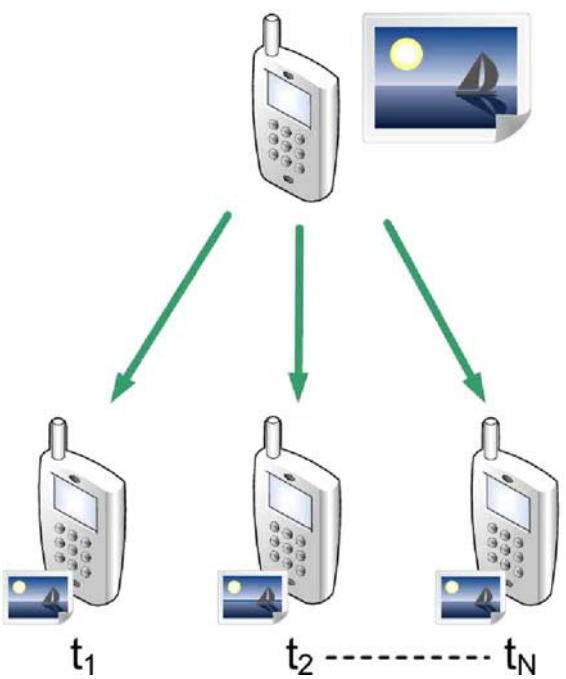

Figure 1. The basic PictureViewer setup.

2.2. Broadcast

Instead of sending to each device individually the source could broadcast the data to the sinks. This approach is highly efficient as long as no errors occur on the wireless link. However, when packet losses occur some form of error correction is needed. To achieve reliability the source needs to know which packets have been lost by one or more sinks and those must be retransmitted, this introduces the need for feedback information which consumes spectrum and time. The amount of feedback information depends on N and the Packet Error Probability (PEP). The feedback messages can be fairly small and as such do not require a lot of spectrum. However, they potentially introduces collisions in the network as both the source and sinks will attempt to transmit packets simultaneously. Thus the performance of such a broadcast approach depends on the effectiveness of the Medium Access Control (MAC).

Furthermore, the retransmissions by themselves is suboptimal as not all sinks will lose the same packets, thus each retransmitted packet will only be useful for a subset of the sinks. For example, if mobile devices 1, 2 and 3 have lost packet 17, 21 and 16, respectively, three broadcast packets must be transmitted, and each retransmitted packet is only useful for a single sink. Generally broadcast can be faster than unicast if $N > 1$ and its performance is less sensitive towards the number of sinks.

2.3. Pure network coding

One NC approach that lends itself to this scenario, is Random Linear Network Coding (RLNC) [2]. With this approach coding is used to simplify the problem of correcting lost packets at the sinks and furthermore reduces the requirement for feedback. In NC, nodes can combine the information in the network to create new packets [8]. Hence, the source codes $g + r$ packets from the g original packets and broadcasts these packets. r is the number of redundant packets and should be chosen according to the PEP of the link. Each sink only has to receive any g linear independent packets, which can then be decoded to recreate the original packets.

DOI: 10.1002/ett

Table I. Estimates of the achievable capacity, C, decoding delay, D, computational complexity, O, and energy consumption, E, when $N >> 1$.

	C	D	O	E
Unicast	Low	Low	Low	High
Broadcast	Med.	Low	Low	Med.
Pure NC	High	Med.	High	Med.
Systematic NC	High	Med.	Med.	Low

The advantage of NC can be illustrated by the previous example. In this case the source could code packets 16, 17 and 21 together into a new packet of the same length as the original packets. This packet is broadcasted to the three sinks, which each remove from the coded packet the packets they already got and thus decode the packet into the packet they lost. Thus, the retransmission that needed three transmissions using broadcast can be done by a single transmission using NC.

As the coding and decoding operations introduces complexity the computational requirement is increased. These operations will increase the Central Processing Unit (CPU) load and thus the energy consumption. However, the number of redundant packets transmitted from the source and feedback messages sent from the sinks can be decreased which help to decrease energy consumption.

2.4. Systematic network coding

To decrease the complexity systematic NC can be used [9]. Systematic NC combines the broadcast and NC approaches. As there is no obvious gain in coding the first g packets, the source broadcasts these packets and code the remaining r packets. Each uncoded packet is useful for all N sinks as they are linear independent. The following r packets are coded and have a high probability of being independent of the n uncoded packets. This approach decreases the computational complexity at the source and the sinks as only r packets has to be coded and decoded.

The different approaches are compared in Table I.

3. NETWORK CODING

This section introduces the coding operations necessary in NC and the algorithms used in the demo application, for details and analysis see Reference [10]. We base our solution on performing RLNC over a Galois field. When Galois fields are implemented on computer systems the Galois elements are generally of the form 2^i, where $i \in \mathbb{Z}^*$, and typically $i \in \{8, 16, 32\}$. We choose the smallest possible Galois Field, GF(2), to decrease the computational complexity of coding operations. This is done to overcome the challenges posed by the limited computational resources available on the test platform.

3.1. Coding operations

In NC 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 refereed 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 $\boldsymbol{M} = [\boldsymbol{m}_1 \boldsymbol{m}_2 \dots \boldsymbol{m}_g]$, where $\boldsymbol{m}_i$ is a column vector.

3.1.1. Encoding. To encode a packet $\boldsymbol{x}$ at the source, $\boldsymbol{M}$ is multiplied with a randomly generated vector $\boldsymbol{g}$ of length g, $\boldsymbol{x} = \boldsymbol{M} \times \boldsymbol{g}$. In this way we can construct $\boldsymbol{X} = [\boldsymbol{x}_1 \boldsymbol{x}_2 \dots \boldsymbol{x}_{g+r}]$ that consists of $g + r$ coded data packets and $\boldsymbol{G} = [\boldsymbol{g}_1 \boldsymbol{g}_2 \dots \boldsymbol{g}_{g+r}]$ that contains $g + r$ randomly generated encoding vectors, where r is the number of redundant packets.

Note that if an encoding vector consists of all zeros except a single scalar that is one, the coded packet is equal to an original packets and we say that it is trivially encoded.

3.1.2. Recoding. Any relay or sink node that have received $g - i > 1$ linear independent packets, can recode and thus create new coded packets. All received coded packets are placed in the matrix $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}_1 \hat{\boldsymbol{x}}_2 \dots \hat{\boldsymbol{x}}_{g-i}]$ and all encoding vectors are placed in the matrix $\hat{\boldsymbol{G}} = [\hat{\boldsymbol{g}}_1 \hat{\boldsymbol{g}}_2 \dots \hat{\boldsymbol{g}}_{g-i}]$, we denote this the decoding matrix. The number of received linear independent packets $g - i$ is equal to the rank of $\hat{\boldsymbol{G}}$. $\hat{\boldsymbol{G}}$ and $\hat{\boldsymbol{X}}$ is multiplied with a randomly generated vector $\boldsymbol{h}$ of length $g - i$, $\tilde{\boldsymbol{g}} = \hat{\boldsymbol{G}} \times \boldsymbol{h}$, $\tilde{\boldsymbol{x}} = \hat{\boldsymbol{X}} \times \boldsymbol{h}$. In this way we can construct $\tilde{\boldsymbol{G}} = [\tilde{\boldsymbol{g}}_1 \tilde{\boldsymbol{g}}_2 \dots \tilde{\boldsymbol{g}}_{g-i}]$ that contains $g - i$ randomly generated recoding vectors and $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1 \tilde{\boldsymbol{x}}_2 \dots \tilde{\boldsymbol{x}}_{g-i}]$ that consists of $g - i$ recoded data packets.

Note that $\boldsymbol{h}$ is only used locally and that there is no need to distinguish between coded and recoded packets when further recoding or decoding is performed.

3.1.3. Decoding. In order for a sink to successfully decode the original data packets, it must receive g linear independent coded packets and encoding vectors. All received coded packets are placed in the matrix $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}_1 \hat{\boldsymbol{x}}_2 \dots \hat{\boldsymbol{x}}_g]$ and all encoding vectors are placed in the matrix $\hat{\boldsymbol{G}} = [\hat{\boldsymbol{g}}_1 \hat{\boldsymbol{g}}_2 \dots \hat{\boldsymbol{g}}_g]$. The original data $\boldsymbol{M}$ can then be decoded as $\hat{\boldsymbol{M}} = \hat{\boldsymbol{X}} \times \hat{\boldsymbol{G}}^{-1}$.

DOI: 10.1002/ett

Note that the set of g linear independent packets can contain any mix of uncoded, coded and recoded packets.

3.2. Coding algorithms

In this section we present pseudo code for the coding operations in RLNC based on GF(2).

3.2.1. Encoding. A packet in GF(2) can be encoded in two simple steps. First the encoding vector, $\boldsymbol{g}$, of length g, 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 $\boldsymbol{M}$ is the data buffer containing all original packets, $\boldsymbol{g}$ is an encoding vector and $\boldsymbol{x}$ is the resulting encoded packet.

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

3.2.2. Recoding of the received $g - i$ packets in $\hat{\boldsymbol{M}}$ is performed similar to encoding. However, instead of combining all g original data packets, the received $g - i$ received packets are combined. First the recoding vector, $\boldsymbol{h}$, of length $g - i$, is generated as a random bit vector, where the indices in the vector corresponds to received packets i.e. index one corresponds to packet one. The second step is performed by iterating over the recoding vector and adding packets where the corresponding index in the encoding vector is 1. Simultaneously the packets corresponding encoding vectors are added in order to create a new encoding vector.

The following listing shows the recoding algorithm in pseudo code, where $\hat{\boldsymbol{M}}$ is the data buffer containing all received packets both partially and fully decoded, $\hat{\boldsymbol{G}}$ is the corresponding encoding vectors, $\boldsymbol{h}$ is the recoding vector, $\tilde{\boldsymbol{x}}$ is the resulting recoded packet, and $\tilde{\boldsymbol{g}}$ is the resulting encoding vector.

```
1:  procedure RECODEPACKET(M̂,Ĝ,h,x̃,g̃)
2:      x̃ = 0̄, g̃ = 0̄
3:      for each bit b in h do
4:          if b equal 1 then
5:              i = position of b in h
6:              x̃ = XOR(x̃, M̂[i])
7:              g̃ = XOR(g̃, M̂[i])
8:          end if
9:      end for
10: end procedure
```

3.2.3. Decoding is performed on the run in two steps with a slightly modified Gauss-Jordan algorithm. 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 and is discarded.

In the second step we perform backward substitution with the newly identified pivot packet. This is done 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 $\hat{\boldsymbol{M}}$ is the packet decode buffer of packets received and decoded so far and $\hat{\boldsymbol{G}}$ is the corresponding encoding vector buffer, $\hat{\boldsymbol{x}}$ is a newly received encoded packet and $\hat{\boldsymbol{g}}$ is the newly received encoding vector.

```
1:  procedure DECODEPACKET(M̂,Ĝ,x̂,ĝ)
2:      pivotposition = 0
3:      pivotfound = false
4:      for each bit b in ĝ do          ( Forward substitution
5:          if b equal 1 then
6:              i = position of b in ĝ
7:              if i'th packet is in M̂ then
8:                  ĝ = XOR(ĝ,Ĝ[i])
9:                  x̂ = XOR(x̂,M̂[i])
10:             elseif pivotfound equal false
11:                 pivotfound = true
12:                 pivotposition = i
13:             end if
14:         end if
```


DOI: 10.1002/ett

```
    end for
    if pivotfound equal false then
        exit procedure                    ( Discard packet
    end if
    for each packet j in M̂ do ( Backward substitution
        k = Ĝ[j]
        if bit at pivotposition in k equal 1 then
            Ĝ[j] = XOR(Ĝ[j],ĝ)
            M̂[j] = XOR(M̂[j],x̂)
        end if
    end for
    Ĝ[pivotposition] = ĝ                  ( Insert packet
    M̂[pivotposition] = x̂
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.

4. DEMO APPLICATION

A demo application, PictureViewer, has been developed to illustrate what is happening when NC is applied. Therefore, the PictureViewer application allows users to broadcast images located on their phones to a number of receiving devices. To illustrate the difference between different NC approaches the application allows users to monitor the decoding process directly. The decoding process is displayed by drawing the actual content of the decoding matrix onto the display of the receiving phones. As the application is primarily meant for demonstration purposes it may not be very useful in the real world. However, if the pictures where substituted with some other data, e.g. video or audio it might be useful for streaming in a local network or similar.

In Figure 2 the first column of screenshots shows the decoding process when pure NC is used. Here only coded packets are transmitted, and initially as shown in Figure 2(a) the content of the decoding matrix appears random. As the decoder receives more linear combinations, the decoding process solves the decoding matrix, and the original picture start to appear, see Figure 2(c). In Figure 2(e) the picture has been decoded and the transmission is complete. The second column of screenshots shows systematic NC, where all data is first transmitted uncoded. Figure 2(b) shows how uncoded packets are being inserted into the decoding matrix. In Figure 2(d) the application has entered the coding phase, where erasures which occurred during the uncoded phase are repaired by transmitting encoded packets. In this test the PEP was approximately 30% and therefore 70% of the data was received uncoded without need for additional decoding. This illustrates the advantage of the systematic approach as the number of packets that had to be decoded was reduced by 70%.

5. RESULTS

In this section we present the results of three measurements evaluating the performance of the used algorithms. The first tests focuses on the performance of the algorithms i.e. the amount of MB/s which can be encoded and decoded using the presented algorithms and the additional energy consumed. In the second test the code is used as an end-to-end code in an ad hoc Wireless Local Area Network (WLAN) to measure the impact of encoding and decoding. In the third test recoding is added and the sinks form a small cooperative cluster, hence the test provides information about the impact of recoding and simple cooperation. These test are intended to provide basic information about how the use NC impacts throughput and energy consumption, in a small ad hoc broadcast network comprising mobile devices with low computational capabilities.

5.1. Coding throughput

To determine the synthetic performance of the encoding and decoding algorithms we have implemented a coding library designed to deliver high throughput by optimising it through assembly and Single Instruction, Multiple Data (SIMD) instructions. This implementation was then ported to the Symbian platform and used in the PictureViewer application which allowed testing the algorithms on commercially available mobile phones. In the following tests the Nokia N95-8GB mobile phone with the following specifications was used; ARM 11 332 MHz CPU, 128 MB RAM, Symbian OS 9.2. In the throughput test a single phone was used to perform both the encoding and decoding operations by first encoding packets, saving the encoded data to memory, and subsequently decoding them. Packets were coded using the generation sizes $q = \{16, 32, 64, 128, 256\}$ and a packet size of 1200 bytes. This test was performed both for pure NC and systematic NC. For pure NC g coded packets were generated and subsequently decoded. To get an indication of the impact of using systematic NC a test was also be conducted where the first $0.7 \cdot g$ of the packets were uncoded and the last $0.3 \cdot g$ packets were coded. Thus, the coding performance corresponds to what would be expected if the packets were transmitted over a channel with PEP = 0.3.

DOI: 10.1002/ett

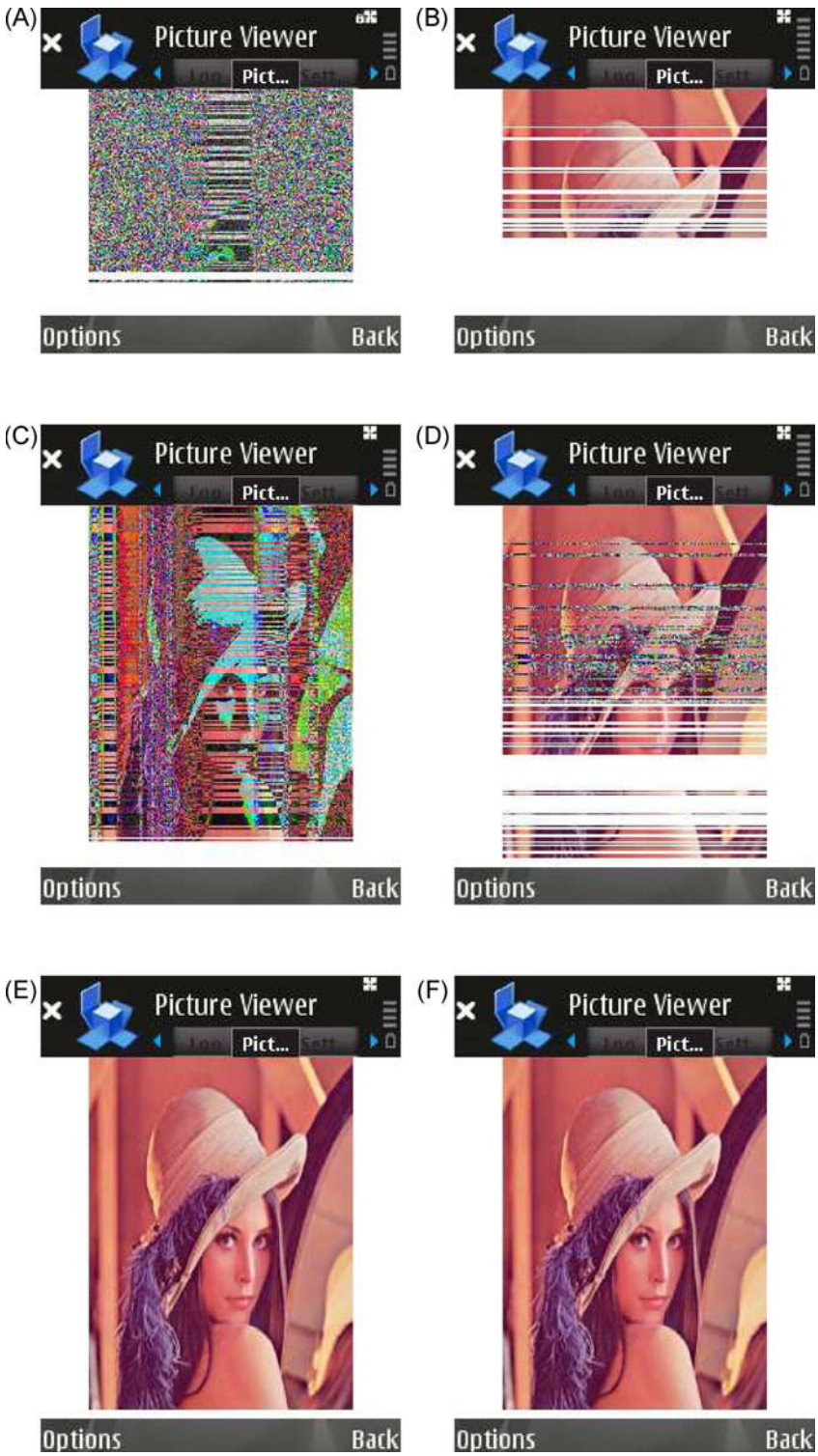

Figure 2. Pure NC: (a) partially decoded data, (c) image starting to appear as the decoders rank increases, (e) the final decoded image. Systematic NC: (b) received uncoded data, (d) erasures corrected by coded packets, (f) the final decoded image.

DOI: 10.1002/ett

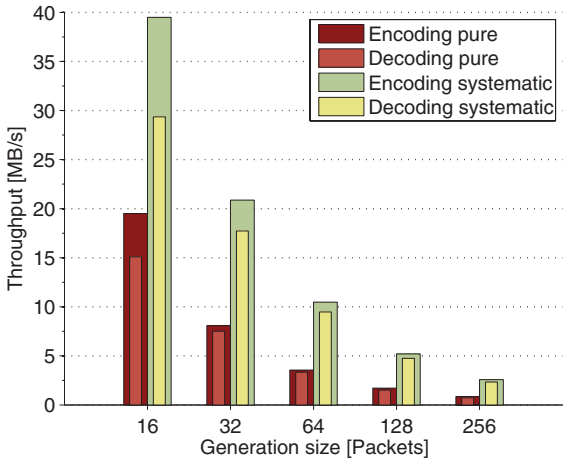

Figure 3. Synthetic throughput for encoding and decoding.

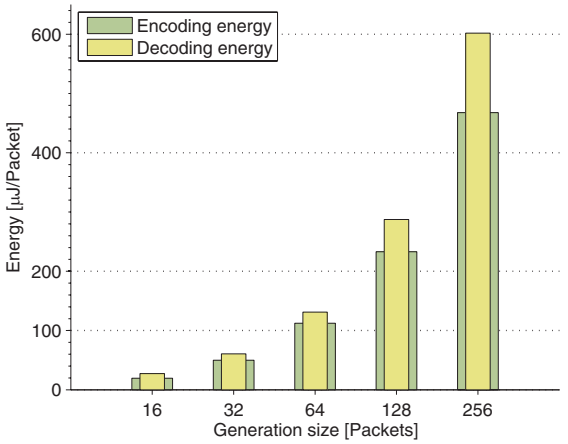

Figure 4. Energy spent per packet during encoding and decoding.

As seen in Figure 3 the encoding and decoding speed decreases as the generation size increases. Additionally the decoding throughput is somewhat lower than the encoding throughput. This is expected due to the higher computational complexity of the decoding algorithm. The test also shows that the systematic approach achieves approximately twice the throughput compare to the pure NC approach for a generation size of 16. For generation size of 64 and above the throughput is approximately tripled. Encoding and decoding of trivially coded packets require no computation, which results in a large speedup for the systematic approach. We note that the coding performance in a real network will depend on the ratio between uncoded and coded packets. In the extreme case where all packets are received coded, the two approaches perform identically and thus have the same throughput. The measured coding throughput indicate that the coding algorithms are fast enough to saturate the WLAN interface for all tested generation sizes, when compared to the achievable WLAN data rates of the Nokia N95 [11]. This result is important as the computational complexity introduced by coding should have minimal impact on the network and device performance, when compared to strategies without NC. A test was also performed to estimate the cost of coding in terms of energy. To measure the energy used for the coding operations, the Nokia Energy Profiler (NEP) was used during the tests. The results from the energy measurement are shown in Figure 4. To calculate the approximate energy consumption per coded packet, the test application first measured the idle power of the device using NEP. This was subtracted from the measured values during encoding and decoding, which gave the approximate power consumption caused by the coding operations.

As seen in Figure 4 the energy consumption per packet increases as the generation size grows. The power consumption was approximately constant during all tests, but as the throughput decreased the energy per packet increased. The energy used to decode a packet is slightly higher than for encoding, this can be attributed to the higher complexity of decoding and the following lower coding throughput.

5.2. Coding in a network

A simple approach to distribute the data from the source to all sinks in Figure 1 is to use systematic coding at the source and overshoot with a comfortable margin. Thus the source transmit so many packets that with a very high probability all sinks are able to decode, and as the source know nothing about the PEP for the sinks, r needs to be high. This is not very useful in a real network, but it allow us to observe the impact that the code has on the channel throughput and energy consumption, and thus choose parameters for the code that are appropriate for our test devices.

We measured from the first packet was received until the packet which completes the decoding of the generation was received. This allows us to obtain the performance as if we had a perfect feedback-channel and feedback scheme. The sink records the following parameters; time per generation, PEP, total packets, uncoded packets, coded packets and linear dependent packets.

The test was conducted using two Nokia N95s, one source and one sink. Packets were coded using the generation sizes $q = \{16, 32, 64, 128, 256\}$ and a packet size of 1200 bytes. Approximately 100.000 test runs were completed in total. All measurements were binned according to their PEP, Table II shows the number of measurements in each bin. We

DOI: 10.1002/ett

Table II. The number of generations counted for different values of PEP.

	PEP [%]				
	0–10	10–20	20–30	30–40	40–50
$g = 16$	40707	10789	509	124	56
$g = 32$	18820	3407	794	662	676
$g = 64$	10907	3373	695	343	150
$g = 128$	9393	1372	287	183	158
$g = 256$	4263	890	215	142	138

Table III. Average number packets per generation for $g = 16$.

	PEP [%]				
$g = 16$	0–10	10–20	20–30	30–40	40–50
n_{sent}	16.63	19.09	22.09	26.92	30.87
n_{recieved}	16.20	16.55	16.91	17.36	17.23
n_{uncoded}	15.52	13.78	12.52	10.64	9.18
n_{coded}	0.48	2.22	3.48	5.36	6.82
$n_{\text{dependent}}$	0.20	0.55	0.91	1.36	1.23

note that the uncertainty in the measurements are higher for high PEP values as fewer results were observed in those bins.

In the following Tables III–VII we have grouped the results according to the different generation sizes. For each generation n_{sent} denotes the average number of packets sent from the source before completing the generation. n_{sent} was calculated using the first and last sequence number in the generation. n_{received} denotes the average number of packets received to complete the generation i.e. including n_{uncoded}, n_{coded} and $n_{\text{dependent}}$ which denote, respectively, the coded packets, uncoded packets and linear dependent packets received.

Several trends in the tables are similar for all generation sizes. As the measured PEP increases the ratio between uncoded packets and coded packets change. This is to be expected as the number of uncoded packets sent in the initial phase is fixed, and as the PEP increases, more and more erasures must be fixed in the second phase of the systematic code. Additionally the amount of linear dependent received packets increases. This makes sense as each coded packet has a non-zero probability of being linear dependent. Inspecting n_{received} we see that the generations are typically completed using between zero and two additional packets depending on the PEP. This is in agreement with the analytical results of the coding performance presented in Reference [10].

In Figure 5, the development in throughput versus PEP is shown. The throughput approximately drops affine with the PEP, and it can be seen how the higher computational complexity of the larger generations sizes affect the performance as the PEP increases and more packets must be coded. The average maximal throughput 0.395 MB/s measured lies approximately 33% below the maximal WLAN throughput without coding presented in Reference [11] on

Table IV. Average number packets per generation for $g = 32$.

	PEP [%]				
$g = 32$	0–10	10–20	20–30	30–40	40–50
n_{sent}	32.88	38.04	43.93	51.86	61.33
n_{recieved}	32.27	32.76	33.25	33.56	33.58
n_{uncoded}	31.32	27.88	24.61	21.30	18.28
n_{coded}	0.68	4.12	7.39	10.70	13.72
$n_{\text{dependent}}$	0.27	0.76	1.25	1.56	1.58

Table V. Average number packets per generation for $g = 64$.

	PEP [%]				
$g = 64$	0–10	10–20	20–30	30–40	40–50
n_{sent}	66.17	75.82	86.53	99.94	118.59
n_{received}	64.56	65.28	65.36	65.37	65.54
n_{uncoded}	62.28	54.52	49.29	44.12	35.91
n_{coded}	1.72	9.48	14.71	19.88	28.09
$n_{\text{dependent}}$	0.56	1.28	1.36	1.37	1.54

Table VI. Average number packets per generation for $g = 128$.

	PEP [%]				
$g = 128$	0–10	10–20	20–30	30–40	40–50
n_{sent}	130.66	150.42	171.63	199.21	237.15
n_{recieved}	128.67	129.39	129.56	129.68	129.69
n_{uncoded}	125.99	109.99	95.12	81.68	72.52
n_{coded}	2.01	18.01	32.89	46.32	55.48
$n_{\text{dependent}}$	0.67	1.39	1.56	1.68	1.69

Table VII. Average number packets per generation for $g = 256$.

	PEP [%]				
$g = 256$	0–10	10–20	20–30	30–40	40–50
n_{sent}	260.69	301.28	340.35	398.73	465.49
n_{received}	256.79	257.59	257.50	257.47	257.61
n_{uncoded}	252.34	219.57	186.49	153.87	120.94
n_{coded}	3.66	36.43	69.51	102.13	135.06
$n_{\text{dependent}}$	0.79	1.59	1.50	1.47	1.61

DOI: 10.1002/ett

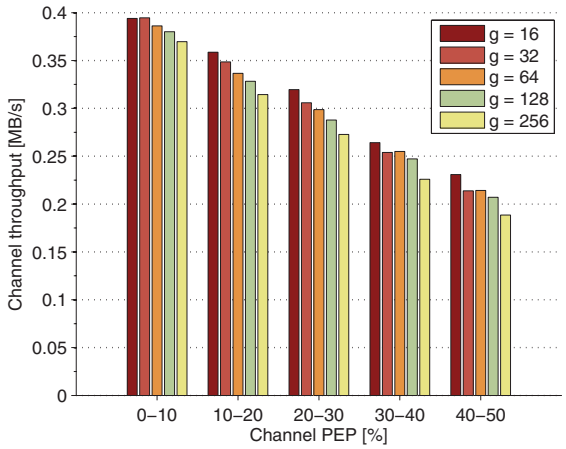

Figure 5. Throughput for different strategies with a single sink as a function of the PEP.

the same type of device. This stresses that the coding operations are not 'free' and further work should be done to optimise their implementation.

Using the measured ratios given in Tables III–VII we are able to calculate the energy consumption of the different schemes. To compute this we use the energy consumed due to the sending and receiving and the energy consumed due the coding operations. We use the values given in Reference [7] for energy receiving and sending data over WLAN and the energy measurement given in Section 5.1. Figure 6 shows the development in energy per byte spent as the PEP increases. The lower generation sizes perform worse, in terms of energy consumption, compared to the larger generation sizes, especially for higher PEPs. Thus in this case

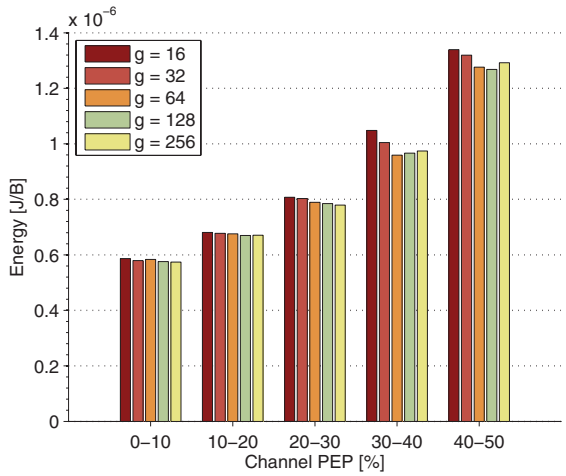

Figure 6. Energy for different strategies with a single sink as a function of the PEP.

we see that the higher overhead in terms of linear dependent packets for small generation sizes outweighs the higher energy consumption of the encoding/decoding of the larger generation sizes.

Based on these results we see an interesting trade-off between energy and speed. Where small generation sizes deliver high throughput, but higher generation sizes deliver a better energy per byte ratio. We also stress that the prototype did not use any form of feedback from the sink to the source. If feedback was introduced e.g. on a per generation basis the lower generation sizes may loose its advantage in speed as it would need a higher amount of signalling.

5.3. Coding and cooperation in a network

To observe the effect of recoding we need to test a setup where the sinks are cooperating by forwarding recoded packets to each other. The simplest approach is to let each sink recode and forward whenever it receives a packet from the source, with some fixed probability p_R. This probability should be chosen in accordance with the PEP of the sinks in the cluster, and will also depend on what parameters we wish to optimise, here we have chosen 5% and 10%. This simple protocol allows us to observe the effect of recoding. Enabling recoding should offload the source, in terms of both energy and computations, by moving some of the coding to the cluster. The effect should be biggest when the PEP is high, and especially visible in cases where the channel between the source and a sink is weak, but the channels between cooperating sinks are strong. In this case a fixed p_R was used; however, for a real protocol implementation it will be important to consider when a sink has enough information to be a useful 'recoder'.

In the test we measured from the first packet was received until the packet which completes the decoding of the generation was received. This allows us to obtain the performance as if we had a perfect feedback-channel and feedback scheme. The sink records the following parameters; time per generation, PEP, total packets, uncoded packets, coded packets, linear dependent packets, relayed coded packets, relayed linear dependent packets.

As in the previous setup the test was conducted using three Nokia N95s, one source and two sinks. Packets were coded using the generation size $q = 64$ and a packet size of 1200 bytes. Approximately 9.000 test runs were completed in total for each p_R. All measurements were binned according to their PEP, Table VIII shows the number of measurements in each bin. We note that the uncertainty in the measurements are higher for high PEP values as fewer results were observed in those bins.

DOI: 10.1002/ett

Table VIII. The number of generations counted for different values of PEP.

	PEP [%]				
	0–10	10–20	20–30	30–40	40–50
$g = 64$ $p_R = 5\%$	8091	891	176	53	63
$g = 64$ $p_R = 10\%$	8114	1355	266	140	109

As in the previous tables rows starting with n denotes data originating from the server, in addition the rows starting with c i.e. c_{coded} and $c_{\text{dependent}}$ denotes, respectively, the useful coded packets and linear dependent packets received from the relay. As shown in the Tables IX and X we can observe the same tendencies as in the non-cooperation case for the source to sink communication. However, in the relay communication we can observe, as expected, that the relay becomes an increasingly better source of information as the PEP increases. The main reason for this, is that the relay will only be able to repair the uncorrelated losses, that is losses which occurred only at the other sink. In addition since RLNC is used the relay will randomly pick which packets to recode, further minimising the probability of selecting an useful packet. However, as the PEP increases so does the probability that one relay has useful packets to offer the second relay. This tendency can be seen for $p_R = 5\%$ where the ratio of useful packets changes from 19% to 73% and for the $p_R = 10\%$ case where the ratio changes from 21% to 58%.

These results indicate that a successful relay protocol should be able to adapt to the current channel conditions in order to avoid sending unnecessary redundant data e.g. when the PEP is low. We do however also see that any packets coming from the relay will aid the source, in the way that the source needs to transmit a relative lower overhead to overcome the channel PEP as part of the redundancy is now coming from the relays.

Table IX. Average number packets per generation for $g = 64$. Using 5% p_R.

	PEP [%]				
$g = 64$	0–10	10–20	20–30	30–40	40–50
n_{sent}	67.12	73.44	80.30	87.75	93.46
n_{received}	63.45	64.65	64.21	62.69	62.72
n_{uncoded}	61.40	57.01	52.57	48.15	44.49
n_{coded}	2.04	5.92	9.98	13.64	16.92
$n_{\text{dependent}}$	0.01	1.72	1.66	0.90	1.31
c_{coded}	0.56	1.07	1.45	2.21	2.59
$c_{\text{dependent}}$	2.46	1.87	1.32	0.92	0.92

Table X. Average number packets per generation for $g = 64$. Using 10% p_R.

	PEP [%]				
$g = 64$	0–10	10–20	20–30	30–40	40–50
n_{sent}	67.48	73.37	80.36	87.31	93.11
n_{received}	62.77	63.75	62.99	63.29	62.66
n_{uncoded}	60.35	56.06	52.14	50.09	47.46
n_{coded}	2.41	6.17	9.19	11.24	13.47
$n_{\text{dependent}}$	0.01	1.52	1.66	1.96	1.73
c_{coded}	1.24	1.77	2.67	2.67	3.07
$c_{\text{dependent}}$	4.65	3.98	3.18	2.89	2.18

6. DISCUSSION

When NC is used several parameters must be defined, these parameters influences the performance in terms of coding throughput, network throughput, decoding delay, etc. A good choice will depend on the type of application, the target platform, the network characteristics, etc. In the following we will discuss these parameters and how they can be selected.

6.1. Parameter considerations

The field size, q, defines the size of the field over which coding operations are performed and also the size of the data symbols. From a network perspective a high q is preferable as it gives a low probability that packets are linear dependent. However, a high q can result in low coding throughputs which can be problematic in many applications and can influence the energy consumption negatively. Here we have considered only $q = 2$, as this is the only choice that have currently been shown to be practical realisable on the target platform [12, 7, 10, 13]. On other platforms this choice can be less restricted [14, 15].

The generation size, g, defines the number of packet in each generation and thus the number of packets coded together. A low g gives a high coding throughput but a higher probability of linear dependent packets, a higher g gives a lower probability of linear dependent packets but a lower coding throughput [10]. Thus the choice of g is a trade-off between network performance and coding throughput. Additionally a higher g increases the decoding delay, which is important for some applications, e.g. audio and video streaming.

DOI: 10.1002/ett

The packet size, m, defines the number of symbols per packet. A higher m increases the coding throughput [14, 7]. However, a high m can be impractical as it can result in fragmentation at lower layers. If one coded packet is fragmented into several frames, and one of these frames is lost, the rest of the frames will be useless.

A good choice of these parameters depend on the application data. For bulk data transfer the requirements to decoding delay is loose, the file(s) will not be usable until everything is decoded. However, if relatively large amounts of data is to be transferred quickly, it is important that the coding throughput is high, in order to reduce the usage of computational resources. For audio and video streaming a very important requirement is a low decoding delay, but the requirement can be loosened by increasing the playback buffer size. This is not possible for VOIP and video conferencing as it would introduce lag in the communication.

6.2. Protocol considerations

The challenge of ensuring reliable multicast transmission in arbitrary networks is an open problem with no solution within sight. To create a usable application this problem needs to be addressed at least for the scenario where the application is deployed.

The solution in the prototype is simply to overshoot, thus sending additional packets for each generation in order to compensate for packet losses. Such an approach is for example used in Multimedia Broadcast and Multicast Services (MBMS) system where the overshooting is tuned based on infrequent feedback from nodes in the network, such that a predefined fraction of the sinks can decode. Because the overshooting is fixed at some level the sinks that experience a packet loss below this level will be able to decode the data, while the remaining sinks will not. This approach is simple and works well if the sinks have relatively uniform and static channel conditions. If the feedback channel is weak or non-existing this may be the only available solution.

Another approach is to let the sinks request more data if they need it. The source sends data from a generation, alternatively it also send some overhead, and then proceeds to the next generation. If any of the sinks were not able to decode the generation they signal that they needs additional information which the source sends. This approach adapts better to changing channel conditions and as such can utilise the channel better, however, the feedback from the sinks introduces the exposure problem [16] and the crying baby problem [17]. Thus this approach works best if the sinks have relatively uniform channel conditions, and if the number of sinks is moderate.

As the links to sinks are independent they will hold different information when the source has transmitted data. Thus an interesting approach is to let the sinks cooperate and thus exploit the connection diversity. Instead of a sink requesting additionally data specifically from the source, any node that received the request could respond, thus more than one node could potentially attempt to answer the request, which would introduce the implosion problem [16]. One of the main drawbacks of this approach is the high complexity it introduces, one technique to remedy this could be the NC. Additionally if done correctly it could potentially allow for transmission in partially connected networks.

Thus in addition to the overall system operation there is several problems reliable transmission in a broadcast network that must be addressed, namely the implosion, exposure, and crying baby problem. Furthermore, a range of protocol functionality is necessary or beneficial, such as service discovery, cluster forming, multi-hop routing, connection loss and reconnection, TCP friendliness, and security, especially when partial connected mesh networks and cooperation is considered.

7. CONCLUSION

In this paper we have introduced a demo application for mobile phones, PictureViewer, that via network coding enables a user to share content with several other users. The application itself is simple but it demonstrates that network coding does not necessarily result in high complexity or overwhelming energy consumption. The implemented algorithms are designed to allow for high coding throughput, therefore a binary Galois Field and a systematic random code was used.

The achieved encoding, recoding and decoding throughput are relatively high when compared with the throughput of the WLAN. As the generation increases the computational complexity increases, as a result the coding throughput decreases and the energy consumption increases. Not surprisingly the systematic approach is considerably faster, especially when the PEP is low.

When the source is encoding and the sinks are decoding the rate at which the source transmits is significantly reduced when the generation size is increased. The energy consumption depends mostly on the PEP, but is also influenced by the generation size. In the test setup a generation size of 64 appears to achieve a good trade-off between coding throughput and linear dependence, if we observe the energy consumption. On platforms with higher computational

DOI: 10.1002/ett

capabilities and/or lower network throughput it is likely that a higher generation size would be a good choice.

The use of recoding was beneficial when the observed PEP was increased. For low values of PEP a large ratio of sent packets where linear dependent. However, as the PEP increased this ratio changed. This indicates that protocols using recoding in this type of networks, should be aware of the channel conditions using recoding only when detecting a certain level of PEP. The use of recoding at the relays was however in all cases beneficial for offloading the source when compared to the non-recoding case.

ACKNOWLEDGEMENTS

This work was financed by the CONE-FTP project grant No. 09-066549, by the Danish Ministry of Science, Technology and Innovation. The authors would like to thank Nokia for providing technical support as well as mobile phones. Special thanks to Mika Kuulusa, Gerard Bosch, Harri Pennanen and Nina Tammelin from Nokia.

REFERENCES

1. Ahlswede R, Cai N, Li SYR, Yeung RW. Network information flow. *IEEE Transactions on Information Theory* 2000; **46**(4):1204–1216.
2. Ho T, Koetter R, Medard M, Karger D, ros M. The benefits of coding over routing in a randomized setting. *Proceedings of the IEEE International Symposium on Information Theory, ISIT '03*, 2003. URLciteseer.ist.psu.edu/ho03benefits.html.
3. Médard M, Koetter R. Beyond routing: an algebraic approach to network coding. *INFOCOM*, 2002.
4. Katti S, Rahul H, Hu W, Katabi D, Medard M, Crowcroft J. Xors in the air: practical wireless network coding. *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06)*, ACM Press, 2006; 243–254.
5. Park JS, Gerla M, Lun DS, Yi Y, Medard M. Codecast: a network-coding-based ad hoc multicast protocol. *Wireless Communications, IEEE [see also IEEE Personal Communications]* 2006; **13**(5):76–81, doi:10.1109/WC-M.2006.250362.
6. Jacobsen R, Jakobsen K, Ingtoft P, Madsen T, Fitzek F. Practical evaluation of partial network coding in wireless sensor networks. *4th International Mobile Multimedia Communications Conference (MobiMedia 2008)*, ICTS/ACM: Oulu, Finland, 2008.
7. Heide J, Pedersen MV, Fitzek FHP, Larsen T. Cautious view on network coding - from theory to practice. *Journal of Communications and Networks (JCN)* 2008; **10**(4):403-411.
8. Fragouli C, Boudec J, Widmer J. Network coding: an instant primer. *SIGCOMM Computer Communication Review* 2006; **36**(1):63–68.
9. Xiao M, Aulin T, Médard M. Systematic binary deterministic rateless codes. *Proceedings IEEE International Symposium on Information Theory*, 2008.
10. Heide J, Pedersen MV, Fitzek FH, Larsen T. Network coding for mobile devices - systematic binary random rateless codes. *The IEEE International Conference on Communications (ICC)*, Dresden, Germany, 2009.
11. Pedersen M, Perrucci G, Fitzek F. Energy and link measurements for mobile phones using IEEE802.11b/g. *The 4th International Workshop on Wireless Network Measurements (WiNMEE 2008) - in Conjunction with WiOpt 2008*, Berlin, Germany, 2008.
12. Pedersen MV, Fitzek FH, Larsen T. Implementation and performance evaluation of network coding for cooperative mobile devices. *IEEE Cognitive and Cooperative Wireless Networks Workshop*, IEEE, 2008.
13. Shojania H, Li B. Random network coding on the iphone: fact or fiction? *NOSSDAV '09: Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ACM: New York, NY, USA, 2009; 37–42, doi:http://doi.acm.org/10.1145/1542245.1542255.
14. Shojania H, Li B. Parallelized progressive network coding with hardware acceleration. *Quality of Service, 2007 Fifteenth IEEE International Workshop on*, 2007; 47–55, doi:10.1109/IWQOS.2007.376547.
15. Vingelmann P, Zanaty P, Fitzek FH, Charaf H. Implementation of random linear network coding on opengl-enabled graphics cards. *European Wireless 2009*, Aalborg, Denmark, 2009.
16. Radoslavov P, Papadopoulos C, Govindan R, Estrin D. A comparison of application-level and router-assisted hierarchical schemes for reliable multicast. *Networking, IEEE/ACM Transactions on* 2004; **12**(3):469–482, doi:10.1109/TNET.2004.828950.
17. Holbrook HW, Singhal SK, Cheriton DR. Log-based receiver-reliable multicast for distributed interactive simulation. *SIGCOMM Computer Communication Review* 1995; **25**(4):328–341, doi:http://doi.acm.org/10.1145/217391.217468.