University of Alberta

Improving energy efficiency in broadcasting and multicasting applications

by

Zohreh Abdeyazdan

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

in

Communications

Department of Electrical and Computer Engineering

Fall 2012
Edmonton, Alberta

Abstract

In this thesis, the problem of transmitting data in the form of multicast and broadcast traffic has been studied. The traffic can be from a real-time application which has a short block length or it can be a general broadcast traffic in an ad-hoc network. The goal is to reduce the total number of transmissions which results in reducing the delay and improving the energy efficiency of the system. Toward this goal, network coding and fountain-codes solutions are studied in this thesis. For the case of broadcasting, a new fountain-code method is proposed. Our results show that this method increases the energy efficiency compared to other methods that have the same order of complexity. For the case of multicasting, we provide a finite block length analysis on the effect of erasure on overhead. We use this analysis to investigate a method of packet construction along with using fountain codes.

Table of Contents

List of Tables

List of Figures

List of Symbols

r :receiving rate
β : probability of loss on the channel (erasure rate)
N : block length
o_t : output packet at time slot t
G : coefficient matrix of random fountain codes
G_{nt} : element nt of coefficient matrix of random fountain codes
Q : number of received packets
E : number of overhead packets
$\mu(d)$: robust soliton distribution
d : degree of encoded packet for fountain codes
d_n : degree at time n
δ : probability of failure
P : length of packet
c^i : data packet i
R : number of nodes/users
$n(i)$: number of neighbors of node i
ρ : density
ϕ : distribution of switched codes
L : length of buffer
φ : binary exponential degree distribution
α_i : weight of degree i in the proposed degree distribution
e_i : erasure rate of class i M : number of classes
s_j : M-tuple symbol
u_i : user in class i
K : total number of time slots

Chapter 1

Introduction

In wireless and wired networks, there are various applications that need broadcasting or multicasting of data. Consider wireless vehicular networks [1] where some data need to be transmitted to a number of cars, which is an example of multicasting/broadcasting. Even for an application that uses unicast traffic, broadcasting and multicasting may be required for the network setup at the beginning of the routing phase. Another example of using broadcasting is when the traffic is unicast, but the topology of the ad-hoc network is changing over time and as a result the broadcasting is used as a mechanism to update the connection between nodes [2]. In other words, information about all the nodes such as their neighbors and the cost of transmission on the links in the network should be distributed, so that different nodes can send their data to a specific destination later.

Broadcasting or multicasting can be from an access point to a number of users in the network or it can be from one of the nodes in the network to some or all other nodes. In all the cases, it is important to reduce the number of required transmissions to distribute the necessary data.

In wireless networks, each transmission usually needs a certain amount of energy. Therefore, reducing the number of required transmissions decreases the required energy [3]. This goal is especially important in networks where nodes do not have access to an unlimited source of energy such as sensor or ad-hoc networks. In these cases, consuming less energy will increase the lifetime of the network [4]. Hence, finding a method that reduces the total number of

transmissions is one of the goals when designing the multicasting/broadcasting algorithms.

In wired networks, reducing the number of transmissions can be interpreted as reducing the required energy and also time for distributing data. In many applications such as realtime applications reducing the delay is one of the primary goals. For example, for video or audio broadcasting/ multicasting if the delay exceeds more than a certain threshold, the data is not useful anymore [5].

Another issue in multicasting and broadcasting applications is the number of feedback packets that should be transmitted over the network. If all the nodes that receive data packets in a multicasting application acknowledge the reception of data per packet to the transmitter, too many feedback packets will be transmitted over the network and the efficiency will be reduced. However, some methods such as rateless codes [6]–[8] and network coding [9] do not need a feedback per packet [6]. Receivers may only send a feedback after receiving a whole block of data. The number of feedback packets can be more important and even critical when sending feedback is very costly or even impossible.

Both rateless codes (also known as fountain codes) and network coding increase the reliability and reduce the number of required transmissions compared to the conventional routing algorithms [3], [10].

In this thesis, we study multicasting and broadcasting applications that use network coding and fountain codes. Our focus is on improving these codes for these specific applications. Specifically our goal is to reduce the total number of transmissions which will result in increasing the efficiency.

In Chapter 2, the required background on fountain codes and network coding along with some of their benefits and applications are discussed. Chapter 3 is a study on using network coding and fountain codes for broadcasting in an ad-hoc network [11], where we focus on broadcasting from all-to-all nodes. Different methods are discussed and the trade-off between complexity and energy efficiency for them is explained. Moreover, we suggest a degree distribution based on fountain codes that reduces the delay and improves the energy efficiency of the network for broadcasting.

Chapter 4 studies fountain codes for multicasting [12], focusing on realtime applications such as multimedia. Here, we analyze a method to send packets of different quality of service data classes and we study its effect on the number of overhead packets. Note that this system model also uses fountain codes. Finally, we conclude this thesis in Chapter 5 where possible future directions are suggested.

Chapter 2

Background

In this chapter, we explain the theoretical background required for Chapters 3 and 4. In the first part, fountain codes are described. Some important fountain codes such as LT codes are discussed and the applications of fountain codes are briefly introduced in Sections 2.1 through 2.5. Network coding and its benefits along with its applications are discussed in Sections 2.6, 2.7, and 2.8.

2.1 Idea of Fountain Codes

Fountain codes also known as rateless codes have been introduced for broadcasting data from one source to many destinations over erasure channels with different erasure rates [6]–[8]. Let us first introduce the erasure channel [13] since this channel model is widely used throughout this thesis.

Definition 1: An erasure channel is a communication channel where a unit of data is either lost with probability of β or is received correctly by probability of $r = 1 - \beta$.

Fig. 2.1 shows a binary erasure channel, where the unit of data is a single bit. This unit of data can be a packet in real applications and each packet can be either received correctly or lost.

One strategy to transmit packets to the receiver over an erasure channel is that the receiver sends acknowledgements for the received packets and transmitter retransmits lost packets. The receiver can send either an acknowledgement per received packet or a request for any missing packet. In response,

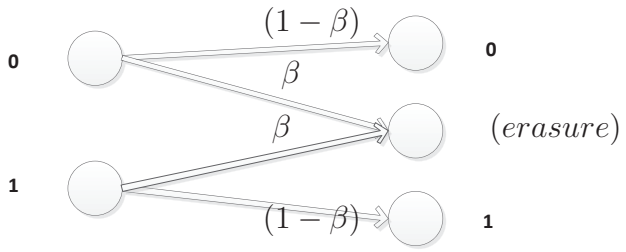


Figure 2.1. An example of a binary erasure channel.

transmitter retransmits the erased packets until they have been delivered to the destination. This method can be costly especially when β is large and many acknowledgement and retransmission packets will be sent over the network.

With more than one receiver, the situation is even worse. We know that the transmission can be in the form of multicasting or broadcasting where each packet should be received by a large group of users. In this case, if each user wants to send a feedback per packet, there will be a huge number of acknowledgement packets. Moreover, even if just one of the receivers does not receive a certain packet, the transmitter has to retransmit that packet over the broadcast channel and all the other users will receive it again. With random erasures on the erasure channel of different receivers, the number of transmissions will increase significantly. Therefore, a large number of redundant packets may be delivered.

For broadcasting or multicasting applications, these redundant packets can be significantly reduced by a solution that does not require a retransmission per lost packet. This leads to the idea of fountain codes where instead of sending each packet separately and retransmitting it in the case of an erasure, the transmitter sends a linear combination of packets each time. Any user who receives enough number of those packets can decode the whole block of data. Hence, it is not important if a user does not receive one certain packet, any encoded received packet is acceptable for users and by receiving enough of them, the original data block can be retrieved by the end of decoding. Each user may need to send a feedback after receiving the whole block of data.

A variety of fountain codes have been introduced and used in different

applications. Random fountain codes [6], LT codes [7] and Raptor codes [8] are explained in the following sections.

2.2 Random Fountain Codes

In general, the transmitter has a block of N packets of data $c^1, c^2, \ldots, c^N$. A time slot can be defined as the required time for transmission of one unit of data. Here, a unit of data which is transmitted over the channel is denoted as a packet. At each time slot, t, the transmitter generates an output packet by combining a random number of packets from its buffer, this output packet is denoted by o_t and is sent over the channel. This is done by generating N random bits G_{nt} $n = 1, \ldots, N$. The transmitted packet, o_t, is constructed from those packets whose G_{nt} is 1 as it is shown in (2.1):

$$o_t = \sum_{n=1}^{N} c^n G_{nt}. \tag{2.1}$$

From (2.1), it can be understood that an output packet can equivalently be seen as a linear combination of the input packets.

Along with a packet, usually there is a header that carries some information. For our purposes, the header indicates which packets are randomly chosen to let the receivers know the variables of the equation that is sent over the channel. This can be done by sending the ID of the data packets or just sending the number of packets and the key of the random number generator in the transmitter. In this way, the receiver is able to reproduce the same random numbers.

At each time slot on the receiver side, there may be an erasure on the channel. So, the receiver will only get some of the equations (output packets). When the receiver gets enough equations, it is able to start decoding. Assume that the receiver gets Q random equations from which it wants to retrieve a block of N packets. Obviously if $Q < N$ the receiver does not have enough equations to decode a block of N packets. Therefore $Q \geq N$ is required. Now

let us define matrix G as

$$G = \{G_{nq}\}_{q=1}^{Q}. \tag{2.2}$$

For the case of $Q = N$ the possibility of decoding without error depends on matrix G. If G is invertible, the receiver can decode all the packets and each packet c^n can be calculated as

$$c^n = \sum_{q=1}^{Q} o_q G_{qn}^{-1}. \tag{2.3}$$

The invertibility of matrix G depends on the linear independency of the Q received equations. For large N, the probability of receiving N independent equations out of Q received ones in almost equal to 0.28 when $Q = N$ [6]. However, it is shown [6] that this probability can be close enough to one if receiver gets $Q = N+E$ equations, where E is a small overhead. Although this overhead causes more transmissions which consume time and energy, instead it reduces the probability of the failure of decoding process. With E overhead packets, the probability of decoding failure is around $1/2^E$. For large N, the overhead percentage can be very small, while decoding success is guaranteed with very high probability.

The main draw back of random fountain codes is their decoding complexity which is cubic with N. Notice that for a low overhead percentage, large N is needed, where complexity is not feasible.

2.3 LT Codes

LT codes define a class of the fountain codes that not only have the desirable characteristics of fountain codes mentioned earlier, but also have practical complexity, that is, the decoding process of LT codes is less complex compared to random linear fountain codes.

In LT codes, instead of using random bits (that are used in random fountain codes) and generating matrix G, a degree distribution is applied for the encoding process which also affects the decoding process as well. In the following the encoding and decoding process and the degree distribution of LT codes will be discussed in detail.

2.3.1 LT Codes: Encoding

The output packet of the LT encoder, denoted by o_t, is generated from a block of N data packets. Here, instead of generating the random bits, G_{nt}, a distribution is used to determine the number of data packets that should be chosen. By definition, the degree of output packet, d, is the number of data packets that are combined to construct the equation. This degree, d, is randomly generated from a degree distribution, $\mu(d)$, which will be introduced later. To generate an output packet, a new d will be generated from the distribution, $\mu(d)$, then d random packets are uniformly chosen and modulo-2 sum (XOR) of them is transmitted over the channel. The characteristics of the desired degree distribution affect the decoding process directly, so the decoding is explained first and the degree distribution will be discussed afterward.

2.3.2 LT Codes: Decoding

The decoding strategy of random codes, i.e., inverting matrix G, is applicable here too, but this high complexity solution is not desired. Assume that the degree distribution generates degrees of equations such that at least one degree-one equation exists at the beginning of the decoding process.

Since the value of a variable c^n in a degree-one equation is known, this value can be inserted to any other equation that has this variable. Therefore, the degree of those equations are reduced by one.

At the second round, a new degree one packet should be found and the same process is repeated. This process is repeated until the value of all the original data packets c^n, $n = 1, \ldots, N$ are determined.

If somewhere in the decoding process, no degree-one packet is available to continue this procedure, the decoding fails. Therefore, enough small degrees should be produced to prevent failure of decoding. To reduce the probability of failure, degree distribution of d should be designed carefully to provide enough small degrees while it is desirable to reduce redundant packets as well.

2.3.3 Degree Distribution

Degree distribution plays an important role in reducing the overhead and in the decoding process as well [6]–[8]. The desired degree distribution should generate enough low degrees especially degree ones to let the decoding start and continue. If in any stage of decoding, the decoder runs out of degree-one packets, decoding will fail. Therefore, the design of degree distribution should prevent this situation. On the other hand, all the original packets should be covered in the received equations. If a packet is not covered by any equation, it cannot be recovered. Thus, a proper coverage is required. Finally these two goals are preferred to be satisfied in a minimum number of overhead packets.

The complexity of encoding and decoding depends on the number of non-zero entries in G, hence the average degree of encoded packets is an important factor. From the discussions in [6] and [7] in order to have a successful decoding, if the number of received packets is close to the optimal number N, the average degree of each packet should be around $\log N$. Therefore, the complexity is now reduced to $O(N(\log N))$ [6]. Complexity can be very important in many applications such as mobile ad-hoc networks where nodes do not have access to unlimited power and processing abilities.

In order to implement a desired degree distribution with this average degree and having ideally exactly one degree-one packet at each stage of decoding process, ideal Soliton degree distribution is an initial option [7]:

$$\sigma(d) = \begin{cases} \frac{1}{N} & d = 1 \\ \\ \frac{1}{d(d-1)} & d = 2, 3, \ldots, N \end{cases} . \tag{2.4}$$

The average degree is about $\overline{d} = \log N$. In spite of the desired characteristic of soliton distribution, this degree distribution is not practical. In many middle stages of decoding process, the probability of having no degree-one packet to continue the decoding process is still high and decoding can fail. Moreover, some of the source packets may not be covered at all.

In order to overcome the above mentioned problems, LT codes uses robust soliton distribution which is a modified version of the ideal soliton distribution. Two new parameters in this distribution are δ and S. S is the expected number

of degree-one packets which is designed to be greater than one in all stages of decoding process.

$$S \equiv c \log_e \frac{N}{\delta} \sqrt{N}, \tag{2.5}$$

where c is a constant of order 1. δ is an upper bound probability of the failure of decoding process after receiving Q number of encoded packets.

In order to generate robust soliton degree distribution another function $\tau(d)$ is defined as

$$\tau(d) = \begin{cases} \frac{S}{Nd} & d = 1, 2, \ldots, (\frac{N}{S}) - 1 \\ \frac{S}{N} \log \frac{S}{\delta} & d = \frac{N}{S} \\ 0 & d > \frac{N}{S} \end{cases} . \tag{2.6}$$

The normalized addition of $\sigma(d)$ and $\tau(d)$ is the robust soliton distribution,

$$\mu(d) = \frac{\sigma(d) + \tau(d)}{Z}, \tag{2.7}$$

where Z is

$$Z = \sum_d \sigma(d) + \tau(d). \tag{2.8}$$

Fig. 2.2 shows $\mu(d)$ which is the robust soliton distribution for $\delta = 0.05$ and compare it to the ideal soliton distribution that suffers from coverage problem.

Using this distribution, it is guaranteed [6] that the decoding process succeeds with probability of $(1 - \delta)$, when the number of received packets is $Q = ZN$. The two required characteristics that were explained earlier is provided by $\tau(d)$. In other words, high probability on small d's provide enough small degrees to let the decoding process start and go on. Moreover, the fairly high weight on $\frac{N}{S}$ will help the coverage problem that the ideal soliton distribution suffers from. Examples in [6] show that the overhead for a file size of $N = 10000$ can be as small as 5%.

Usually in LT codes, just a small portion of packets can be decoded before receiving Q packets where Q is slightly larger than the original file size N. But after that point, the decoding is very likely to become complete.

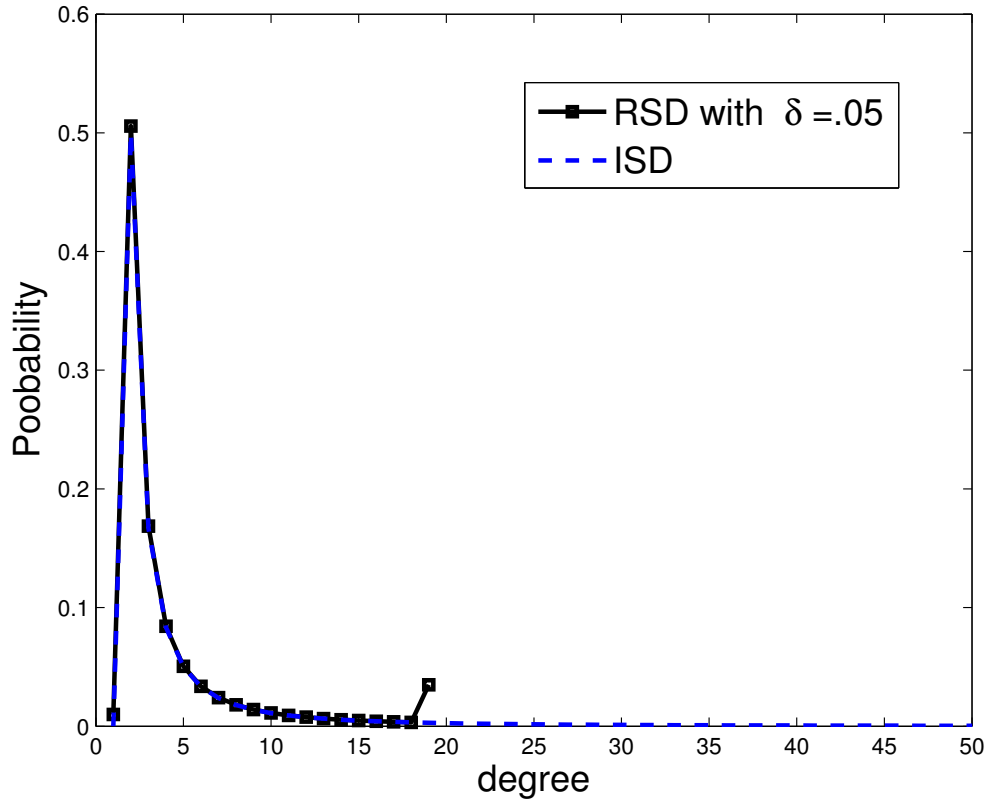


Figure 2.2. Comparison of Ideal Soliton Distribution(ISD) and Robust Soliton Distribution (RSD) for $\delta = 0.05$.

2.4 Raptor Codes

LT codes have a reasonable encoding and decoding complexity as we explained in Section 2.3.3, but Raptor codes improve this complexity even more and achieve linear time encoding and decoding [8]. Based on the explanation for LT codes, the average degree is $\overline{d} = \log N$ which affects the complexity directly. Raptor codes [8] reduce this average to $\overline{d} = 3$. But, having a low average degree may cause some problems, a fraction of source packets may not be covered in the equations. For $\overline{d} = 3$, this fraction is about $\widetilde{f} = e^{\overline{-d}} = 5\%$.

In order to reduce the complexity and solve the coverage problem at the same time, Raptor codes first use an outer error-correction code to pre-code the block of N data packets to $\widetilde{N} = N/(1 - \widetilde{f})$ where $\widetilde{f}$ is the fraction of packets that are not covered in the equations and can be calculated by having

the distribution [6] as we mentioned above. After that, an LT code with a low average degree is used which has a lower complexity compared to original LT codes. Therefore, the erasure packets from fountain codes, i.e., the fraction of packets that were not covered, can now be recovered by means of the outer code.

In [6] there is an example that shows original LT codes can recover 10000 packets out of 11000 received packets, where the low average LT codes can recover 8000 packets out of 9250 received packets. This simple example shows how using pre-coding can reduce the need to recover the the whole data from fountain code. For the outer code, in [8] an irregular low density parity check code has been used.

2.5 Application of Fountain Codes

Fountain codes can be used in different applications. Two important applications are briefly explained in the following:(1) storage of data and (2) broadcasting/multicasting to a number of users. The second application is further studied in Chapters 3 and 4.

2.5.1 Storage

Storing a file on a hard drive, disc or even in a distributed network traditionally has been done by storing the original data packets. By using fountain codes, instead of storing data packets, the encoded packets can be stored in any place on the device. To read this file from the device, all we need to do is to read Q encoded packets which gives us Q equations and consequently Q original packets can be extracted from them. This way if some of the packets are lost, recovering is much easier and faster since any Q packets are enough to recover the data and there is no need for any specific packet. So even if a few packets are lost, the whole data can still be recovered as long as Q good packets are still available. Note that the storage can be in one place or in a distributed manner in a network [14].

2.5.2 Broadcast/Multicast

One of the main applications of fountain codes is broadcasting/multicasting data to a number of users [10], [15]. An example could be a stream of multimedia which is being transmitted to a large number of users. If the transmitter broadcasts the original packets without fountain coding, with the erasure rate of β for a user, that user on average will receive $(1-\beta)N$ of the packets and this β may vary for different users. Therefore, the source node has to retransmit those βN packets again while other users with better channel condition are likely to already have received them. This traditional method makes the number of transmissions and feedbacks incredibly large even for small values of β.

As we explained before, by using a fountain code, each user only needs to receive any Q of transmitted packets. Therefore, the total number of transmissions will be almost $(1+\beta)N$ and no retransmission is required anymore.

One important aspect of using these codes is to reduce the need to send and receive feedback. For some cases where sending feedback costs too much or is even impossible, these codes are ideal, because users do not need to report lost packets and ask for retransmission [6]. The user only needs to receive any Q packets.

A similar scenario can be applied for multicasting, where different classes of data are available and different users wish to receive different data classes. This case will be discussed in more detail in Chapter 4. Another example is broadcasting from all-to-all instead of one-to-all in a multi-hop basis. In that case, all the users are the source of a portion of data while this data should be received by all users. This case will be discussed in Chapter 3.

2.6 Network Coding

Usually in traditional methods of transmitting information, each stream is a separate flow that can not be combined with other flows of information until it is delivered to its destination. However, network coding [16], [17], [18], allows different sessions or flows to be combined in the intermediate nodes of

a network. This idea can potentially improve the overall throughput of the network and increase the robustness as well. For example, in a case of packet loss or when an error happens, the network is much more resilient compared to the time when separate flows have been transmitted over the network [9]. In a simple form of network coding, an intermediate node can send the summation of all the input streams as the output stream.

In a very simple example, the idea of network coding can be explained. Fig. 2.3 shows how combining two different flows of information in a wireless network can improve the throughput. In this figure the goal is to deliver the data of two nodes A and B to each other. As it is shown in Fig. 2.3 network coding reduce the overall required time of transmission.

In linear network coding, any linear combination over some finite field can be sent and these fields must be large enough to provide linear independency between transmitted equations [17]. Note that this linear combining requires each middle node of the network to have some computational capabilities.

Since all the calculations and combinations in the network coding are done in Galois fields, before discussing network coding, the basics of Galois fields are briefly explained.

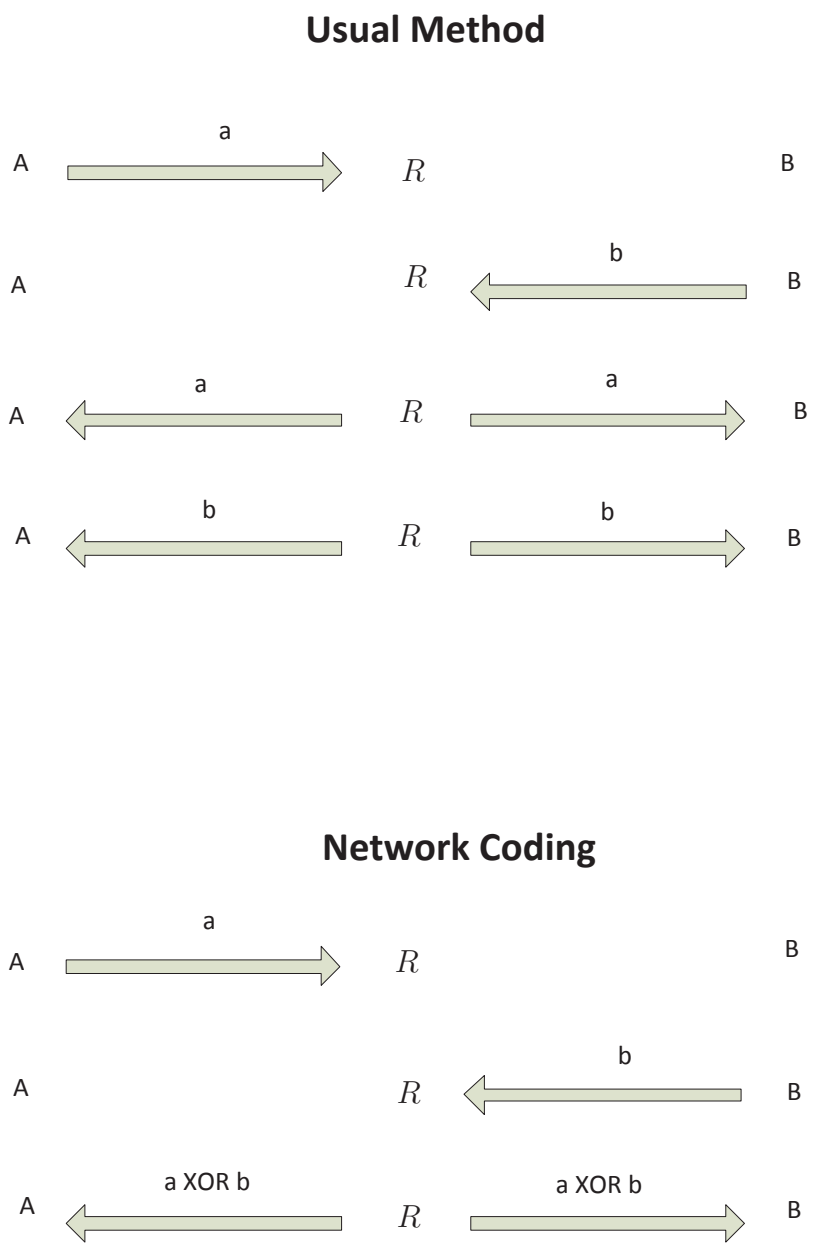


Figure 2.3. Using network coding for wireless networks, where R is a relay node.

2.6.1 Galois Fields

A field is a set of elements where we can perform addition, subtraction, multiplication and division without leaving the set. These mathematical calculations should have some characteristics. Formal definition of a field from [19] is

Definition 2: "Let F be a set of elements on which two binary operations called addition "+" and multiplication "$\cdot$" are defined. The set F together with two binary operations "+" and "$\cdot$" is a field if the following conditions are satisfied.

- F is a commutative group under addition "+". The identity element with respect to addition is called the zero element.

- The set of nonzero elements in F is a commutative group under multiplication "$\cdot$". The identity element with respect to multiplication is called the unit element of F.

- Multiplication is distributive over addition. That is for any three elements a, b, and c in F

$$a \cdot (b + c) = a \cdot b + a \cdot c." \tag{2.9}$$

Consider a prime number p. Modulo-p addition can be defined over the set of integers $0, \ldots, p-1$ as the remainder of the real addition between any two elements of this set divided by p.

Modulo-p multiplication can be defined in the same way. It is shown in [19] that the set of integers $0, 1, \ldots, p-1$ is a commutative group under modulo-p addition. Moreover, the set of $1, \ldots, p-1$ is commutative under modulo-p multiplication. Following the fact that real multiplication is distributive over addition, it can be checked, [19], that modulo-p multiplication is distributive over modulo-p addition. Therefore, the set of $0, 1, \ldots, p-1$ is a field of order

p and is denoted by $GF(p)$. A famous prime field is GF(2) which is a simple binary field.

Galois fields can be constructed from either a prime number p or any $q = p^m$. A widely used field in coding and data storage is $GF(2^m)$ and is based on the binary field. A lookup table for addition and multiplication can be used for any field. The one for binary is following the boolean logic rules of bitwise "XOR" for addition and "AND" for multiplication. For each $GF(2^m)$ a polynomial can be defined and with that polynomial all the elements can be constructed. More detail about these polynomials and a list of them for various powers of 2 can be found in [19].

2.6.2 Linear Network Coding

Using the concept of fields, we can explain the idea of network coding. Consider a relay node in a wireless network. When it receives incoming packets, instead of repeating them as the output packets, the node can combine the received packets, so several output packets from the received packets can be created.

Packets can be generated from one or several sources and they can be destined to only one node or in a general case, they can be delivered to many nodes depending on the requirements of the application. The number of sources or destinations does not change the function of intermediate nodes and encoding/decoding process.

The details of encoding which results in creating output packets and decoding the encoded packets at the destination are discussed in the following.

Encoding and Decoding

Let us assume that P is the length of a packet which means a packets consists of P bits. Combining packets should not change the length of a packet but if the received packets do not have the same length, enough 0s are added to smaller packets to provide the same length for all packets. After linearly combining received packets of length P in $GF(2^m)$, the resulting output packet with the same length is ready to be transmitted [9]. Assume that $c^1, \ldots, c^N$ represent original data packets that were generated from one or several source

nodes and each of them is constructed by symbols taken from a certain Galois field. For example if the field is GF(2) the symbols are bits. An output packet can be constructed as

$$W = \sum_{i=1}^{N} g_i c^i, \tag{2.10}$$

where g_i is a random coefficient in $GF(2^m)$. When $m > 1$, each symbol of the packet W which is denoted by W_l can be defined as

$$W_l = \sum_{i=1}^{N} g_i c_l^i. \tag{2.11}$$

where c_l^i is the lth symbol of c^i. If a relay or an intermediate node receives packet W that is already linearly encoded as in (2.10), it will apply its own coefficients, h, on packet W with coefficient vector g. Therefore, the coefficients of the resulting packet, W', which is denoted by g', can be calculated as

$$g'_l = \sum_{j=1}^{x} h_j g_l^j, \tag{2.12}$$

where x is the number of coefficients of the second node. It is assumed that each node sends the coefficients as part of header along with the packet, so that the calculation of (2.12) is possible. Moreover, to decode all the received equations in the destination, all the coefficients are needed as well.

To get the original packets at the destination, receiver needs to have $Q \geq N$ equations to retrieve $c^1, \ldots, c^N$ packets. Hence, the receiver should solve a system of linear equations as 2.10.

However, to guarantee that even with $Q \geq N$ equations the system can be solved, the linear independency of equations should be provided. The linear independency of equations depends on the selected coefficients at the intermediate nodes. There are two ways to provide this required independency. The first one is to use centralized deterministic coefficients at the intermediate nodes that are designed to provide the independency [20].

Another method is to use decentralized random coefficients in each node which will result in certain probability of dependency depending on the size of the field [17]. To get a desirable result with decentralized method, the field

should be large enough and coefficients are randomly chosen over that field. It has been shown in [21] that even for fairly small field of $GF(2^8)$ the probability of linear dependency is almost negligible.

In order to solve the received equations as a system in the destination, all the coefficients are stored in a matrix that is called the decoding matrix. The unknowns is the vector of c^i's, $i = 1, \ldots, N$, and the vector W is known. Hence, each encoded packet is stored in a row of the decoding matrix, and it can be checked if it is innovative or non-innovative, the non-innovative packets can be removed. A packet is innovative when it increases the rank of the matrix which means it is linearly independent from previously stored packets. As soon as there are N rows in the decoding matrix, it can be solved through Gaussian elimination method.

There are some issues that should be considered for linear network coding. Decoding is done by Gaussian elimination which has a high complexity. Hence, the size of the decoding matrix is a practical limitation. One practical solution to reduce this complexity is to reduce the size of the matrix by using generations [22]. It this way, each group of packets are marked as one generation and will be combined only in their own generation in the intermediate nodes. Although there are some methods that combine some generations to reduce the delay [23], the idea of generations is to keep them separate to reduce the complexity. Hence, the size of the matrix can be reduced to the size of generation. The trade-off between the required memory and computational complexity along with other considerations such as delay for real-time applications usually determine the size of generations.

2.7 Benefits of Network Coding

Throughput: One of the main advantages of network coding is to improve the throughput of the network, especially in the case of multicasting. In other words, if there is more than one receiver and all the receivers require to receive the data of all the sources, it cannot be done efficiently without network coding [16], [24]. Fig. 2.4 shows an example where S_1 and S_2 are two sources of data

and both R_1 and R_2 want to receive data packets x_1 and x_2. If no network coding is used in the intermediate nodes, the achievable rate is less compared to the case where simple network coding is used.

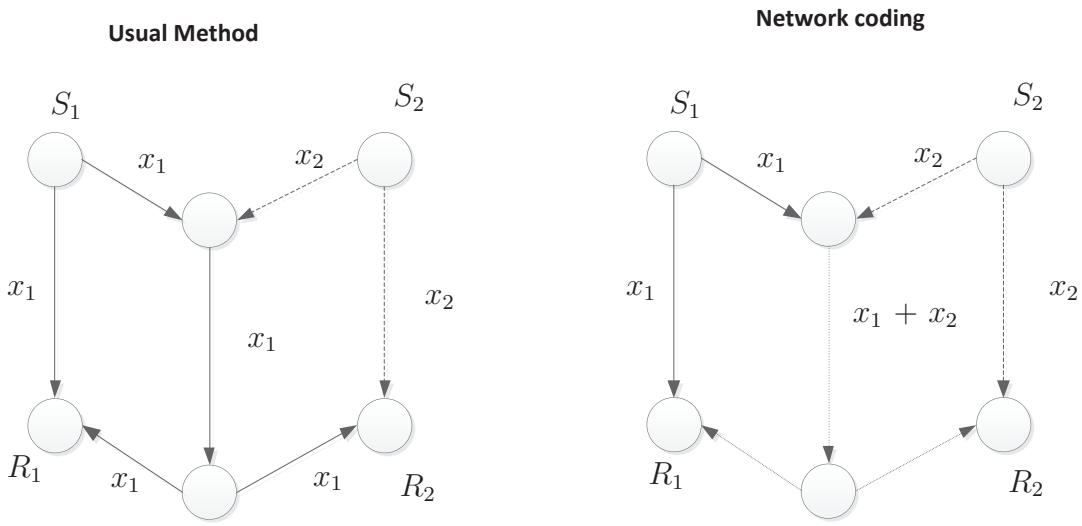


Figure 2.4. Using network coding to improve throughput.

Throughput advantage is not limited to multicasting applications. Even for broadcasting and unicasting, it can be seen that throughput can be improved via network coding. For example, in Fig. 2.4 even when S_1 wants to send data to R_2 and R_1 is the destination of S_2, network coding is still useful and can reduce the overall transmission time. Achieving the optimal throughput for multicasting applications with routing is an NP-complete problem [9]. However, by using network coding, the optimal throughput for multicasting applications can be reached via polynomial time algorithms.

Robustness: One of the most important features of network coding is robustness. Network coding takes different data packets and produces a stream of encoded packets similar to other coding methods. On the receiver side, if enough number of encoded packets are received, the node is able to recover the whole data and there is no need to receive a certain data packet (similar to fountain codes).

Using network coding makes the network more adaptable. For example in Fig. 2.3, if relay sends the linear combination instead of the original data packets, any time that either A or B are in idle mode without notifying relay, there are still some information for the other awake node. However, without network coding, transmission of only A or B data packets might be a waste of time without knowing which one is in idle mode. In order to explain robustness and adaptivity of network coding, two examples are considered in the following.

One of the applications of network coding is collected coupon problem [25]. In one form of this problem, there is a network with R nodes and the number of messages that should be passed is of $O(R)$ and all the nodes must receive all messages. It has been shown that a centralized gossip-based algorithm, where each node forwards messages based on a forwarding probability, needs the same number of rounds that a decentralized network coding method does [26]. However a decentralized routing-based algorithm can not achieve that bound without network coding in place. More details on this problem for both network coding and fountain codes will be discussed in the next chapter.

There are also applications that are delay sensitive or have high data rates where the channels follow the erasure channel model and the packets can be

dropped. One strategy is automatic repeat request (ARQ) where each time a packet is dropped, requests for missing packets will force a large delay. Another strategy is to use forward error correcting (FEC) methods which are not rate efficient but their delay is less than ARQ such as fountain codes [6]. In order to reduce the delay and have the optimal rate at the same time, network coding can be used as a solution. Therefore, by letting the intermediate nodes to combine the packets, it is possible to deal with the problem of erasure in an efficient way.

2.8 Applications of Network Coding

Peer-to-Peer Networks: One of the important applications of network coding is related to the distribution of files in peer-to-peer networks [27], [28]. In these applications usually users download information from the server which will be transmitted in the form of data blocks. Not only users receive blocks of data from the server, but also the data is distributed among users too. Therefore, each user as a node gets some of the data blocks from the server and some from other nodes while it sends its own blocks to other nodes which are its peers.

These peers can be randomly selected among the set of available peers to a node. Having linear network coding available in the server as well as in peers, the server sends the linear combinations of data packets instead of the original data. Each peer is transmitting a linear combination of data packets to other peers until they do not receive innovative packets from that node anymore. At that point, the node stops transmitting and waits to receive more innovative packets itself to create innovative packets for its peers again.

Using network coding reduces the download time and eliminates the need for a complex scheduling algorithm [27]. All the nodes send linear combinations of their received packets instead of checking what other peers have and check who should send which specific data packets. Moreover, the robustness of network coding is much better than the traditional method, if any of the nodes or the server becomes idle, because of the diversity of coded blocks, this

system is still robust [9].

Wireless Networks: Network coding can be used in a variety of applications in wireless networks. One example is shown on Fig. 2.3.

Another application is broadcasting from all-to-all nodes in a wireless network which will be discussed in the following chapter in more detail. Using network coding or fountain codes for this application reduces the number of transmissions significantly and is energy efficient.

Ad-hoc Sensor Networks: One of the practical difficulties of sensors in wireless sensor networks is related to their oscillators [29]. The oscillator should be tuned and accurate enough to make the communication possible. For that purpose, quartz oscillators which are complex and costly have been used. However, if analog oscillators, that are cheaper and easier to built, are used, the problem of untuned frequency should be considered, because the probability of one successful direct communication between two nodes will be reduced due to their untuned oscillators [29].

If many nodes have this problem, the probability of finding a path and communicating through it, will be reduced even more. However, for a dense network with many paths and network coding in place, there is no need to find a single path to send a message. Network coding is more reliable compared to traditional routing algorithms and messages are distributed all over the network. Therefore, the use of cheaper oscillators is possible by using network coding [29].

Network Tomography: Network coding can also be used to infer information about network such as packet loss rate of the links [30], [31]. Usually to probe the network, probing packets are multicast over the network. The receivers can estimate the loss rate of the multicasting tree, by the number of probes they receive, assuming that the number of transmitted probes are known. However, using network coding can even provide more information about network to the receivers. Packets not only can be multicast in the middle points of the multicasting tree, but also they can be merged.

Merging packets with the use of network coding on their way to the receiver, is the same as when the receiver initiates a multicasting itself as a source.

Therefore, extra information about the topology of the network is provided through network coding. In order to find out which link causes the loss, some known coefficients can be used for network coding. Hence, the link that causes the loss can be determined from the received packets [30].

Network Security: There are different issues that can be addressed in network security, but the one that network coding can help with [32], is eavesdropping. Eavesdropping pertains to the attempts to recover a part or the whole data by unauthorized users. Applying network coding makes it harder for eavesdroppers to recover the original data, because the data is distributed and coded over the network. Moreover, coefficients can be designed to provide more secure communication between source and receivers [32], in a way that the mutual information between the packets that the eavesdropper receives and the original packets is zero. By using network coding, eavesdropper might not be able to receive all those packets because a user needs to get sufficient number of linearly coded packets to be able to decode them.

Chapter 3

Energy Efficient Broadcasting for Multi-hop Wireless Networks

In this chapter, the problem of all-to-all nodes broadcasting in wireless networks is studied. We start by an introduction in Section 3.1, then system model and an overview of some previous solutions are discussed in Section 3.2. The proposed method is presented in Section 3.3. Section 3.4 presents the numerical results and Section 3.5 concludes this chapter.

3.1 Introduction

Wireless ad-hoc networks have been widely used for various applications, unicast, multicast, and broadcast traffic can be transmitted over these networks depending on the requirements of the application. Although an application may only need unicast or multicast traffic, it still uses broadcasting for the discovery phase of the routing. Therefore, broadcasting plays an important role in these networks. Our definition of broadcasting means that all nodes have data that should be received by all other nodes. Broadcasting can be done in different ways such as flooding [33].

Since each transmission consumes a certain amount of energy, the number of transmissions is directly proportional to the energy consumption in the network. Therefore, the number of transmissions can be used as the indicator of the energy usage. Total delay is another performance metric for comparing

different broadcasting methods. Total delay is defined as the total number of time slots required until all nodes receive the data of all other nodes.

The problem with flooding is the huge number of required transmissions resulting in a poor energy efficiency. Consequently, the total delay will be quite large too. A general idea to reduce the number of transmissions is to apply coding. One solution which takes advantage of network coding is proposed in [3], [34]. Network coding lets intermediate nodes to mix the received data packets, hence it is suitable for this problem configuration where all nodes want to receive data from different nodes in the network. Although the method of [3] and [34] is energy efficient, it suffers from high complexity. In each node, decoding is done by Gaussian elimination, so the complexity of decoding is $O(R^3)$ where R is the number of nodes in the network.

Another solution is to use fountain codes for this problem. Fountain codes have been used to broadcast data from a transmitter to a number of users [6]–[8]. For example [10] applies fountain codes to broadcast data in a wireless sensor network from a single source to all other nodes.

Using fountain codes for broadcasting from all-to-all nodes has been considered in [35]. In this method, intermediate nodes decode all the received encoded packets and re-encode the data packets again and transmit new encoded packets on the broadcast channel. Therefore, fast decoding in the intermediate nodes is one of the main considerations for choosing a proper code. At the same time, reducing the number of transmissions and consequently improving the energy efficiency and delay of the network is a goal.

Using fountain codes reduces the complexity of decoding from $O(R^3)$ to $O(R)$, but simulation results show that the number of transmissions and delay suffer in return. In this chapter, we propose a solution for broadcasting from all-to-all nodes based on fountain codes. In comparison to [35], our solution has the same decoding complexity, a lower encoding complexity and a reduced number of required transmissions. It is also more energy efficient and improves the delay as well.

3.2 System Model and Previous Solutions

Consider a network with R nodes where all the nodes have some data to send, so there are R sources of data in the network. The data of each node should be broadcasted to all other nodes. The data is transmitted in packets in the network.

Any node in the transmission range of a generic node is called a neighbor of that node. When node i broadcasts a packet, for non-neighboring destinations, the packet goes on a multi hop route. It is assumed that the transmission of a general node is not received nor heard by the nodes that are out of the transmission range of that node. The number of neighbors of node i is denoted by $n(i)$. Based on the transmission range and the area of the network, the number of neighbors of node i can be varying from 0 to $R-1$.

As we mentioned earlier, there are two methods to broadcast data packets in the whole network using coding. The first one applies network coding [3], while the second one, [35], uses fountain codes. These two methods will be explained below. Please note that for simplicity, in the following discussion we assume each node has one data packet to send, but this solution can be applied to a general case as well.

3.2.1 Probabilistic Network Coding

The first method that we review is probabilistic network coding, [3], [34], that use a probabilistic forwarding strategy and network coding together for an energy efficient broadcasting. Suppose that the data is represented in $GF(2^q)$. In the scheduling phase which is determined by the MAC layer, a node is chosen as a transmitter. The transmitter chooses random coefficients in $GF(2^q)$ to form a linear combination of the previously received packets that are stored in its buffer. This linear combination is sent over the broadcast channel. It also sends the coefficients with the encoded packet, so that the packets can be decoded at the destination.

The probabilistic forwarding approach is based on a forwarding factor $d(i)$ for node i. Based on the forwarding factor the number of transmissions in each

node is determined. In comparison to methods that always perform forwarding upon a new arrival, this probabilistic forwarding method reduces the number of redundant packets. The probability of forwarding upon a new arrival for a generic node i is determined by its forwarding factor $d(i)$ which itself depends on the density of the network and the number of its neighbors.

By the end of transmissions, every node should be able to solve its matrix of received equations (encoded packets) to decode all transmitted packets from different nodes. More details on how to determine $d(i)$ and other aspects of this method can be found in [3], [34].

Although probabilistic network coding is an energy efficient way to broadcast data for this problem, this method has two important shortcomings. First, the Galois field should be large enough to provide proper random coefficients that avoid linear dependency of equations. Another important problem is the high complexity of decoding at the receiver. In other words, since Gaussian elimination must be used for decoding, the complexity of decoding N packets at a given node is $O(N^3)$. If back substitution decoding could be used instead of Gaussian elimination, the complexity would be $O(N)$. This is the main reason we seek a solution based on fountain codes to provide reliable broadcasting.

3.2.2 Switched Codes

In this subsection, we review a special fountain code (referred to as switched code) which is designed in [35].

Switched codes are developed in [35] based on using fountain codes for the problem of broadcasting from all-to-all nodes. In this method, each node starts transmitting packets based on a fountain code with a certain degree distribution. The transmitting node chooses a degree d from a degree distribution $\phi(d)$ and forms the XOR of d randomly selected packets from its buffer of decoded packets and its own packets to be transmitted on the broadcast channel.

On the receiver side, when a node receives an encoded packet, it starts the decoding process. Based on the previous decoded packets and its own data

packets, if there is only one unknown data packet in the new received encoded packet, it can be immediately decoded. Otherwise, the packet will be stored in a separate buffer until its decoding will be possible. Once a new data packet is decoded in a node or a degree one packet is received, it will be used to decode all the stored packet in the buffer or to reduce their degrees if possible.

The authors in [35] and [36] use a combination of two different degree distributions for this application. At the beginning, they use binary exponential distribution (BED):

$$\varphi(d) = \begin{cases} \frac{1}{2^d} & d = 1, \ldots, L(i) - 1 \\ \frac{1}{2^{L(i)-1}} & d = L(i) \end{cases} . \tag{3.1}$$

where $L(i)$ is the number of data packets available at node i or equivalently it can be assumed as the buffer size of node i. When the decoding starts at the receiver, the high probability of choosing small degrees in BED helps fast decoding. Hence, in the receiver, decoded packets can be used as the data packets for the encoding. In other words, the next time that this node acts as the transmitter, it can use these packet to transmit a new encoded packet. But BED causes redundancy too, covering all the data packets will take a long time because of choosing small degrees with high probability. That is the reason [35] suggests switching the degree distribution to that of LT codes after a while. Based on two counters suggested in [35], the nodes switch the distribution from BED to LT (i.e. from $\varphi(d)$ to $\mu(d)$). The decision of changing the distribution is based on the number of packets that has been sent and the number of data packets available in the buffer of that node (its own packets and received packets).

Although this method reduces the complexity, it suffers from a large number of transmissions. The main difference between using fountain codes and network coding for this problem definition is that network coding combines encoded packets in the intermediate nodes and send the linear combination on the channel while the methods that are based on fountain codes decode the received packets and then re-encode data packets. The reason for decoding and re-encoding data packets is to maintain a certain degree distribution. This degree distribution causes the lower complexity of decoding in fountain

codes compared to network coding that has $O(N^3)$ complexity for N nodes.

In the next section we introduce our solution to this problem. Our solution is based on fountain codes to ensure a low decoding complexity. Moreover, simulation results in Section 3.4 show our method improves delay and energy efficiency compared to that of [35].

3.3 Proposed Method

Our proposed solution applies fountain codes to decrease the decoding complexity compared to the solutions based on network coding. There are optimal degree distributions for fountain codes that are used on one-to-many broadcast setups, but those solutions are not immediately applicable to our setup where many-to-many broadcasting is desired. In other words, it is not trivial to maintain a desired degree distribution at receivers while the source and the encoding nodes are distributed in the entire network.

In this work, we propose a degree distribution for the transmitters that at the receiver side mimics a distribution similar to the optimal LT degree distribution. As a result, this degree distribution improves various performance measures when compared with the previous method. Our proposed method has linear complexity of decoding and it also reduces the total number of transmissions compared to switched code that has the same complexity. It is important to note that using a code such as LT code in the transmitter will cause large delays and an increased number of transmissions (because of the nature of the distributed sources) and [35] shows that switched codes already outperform LT code for this problem setup.

Studying the received degree distribution at a decoding node reveals that switched codes themselves suffer from sending an unnecessarily large number of degree-one packets. This is because at the beginning each node has only its own data and the degree is forced to be one. Notice that, choosing a degree distribution with a large weight on degree one results in inefficiency. The main goal of this work is to find a more efficient degree distribution for the many-to-many data sharing setup.

Before introducing our improved degree distribution, we first discuss a simple modification that we suggest. This modification, referred to as forced first transmission strategy (FFTS), improves the efficiency of switched codes or any other solution based on fountain codes. Thus, we use this modification for our own degree distribution too.

3.3.1 Forced First Transmission Strategy

In FFTS, all the nodes have to use their own data in the encoding process of their first transmission. In a general node, if the first transmission happens before any receiving, the node automatically sends its own data. In fact, the degree d is forced to be one and the transmitted packet is the node's own data packet since there is no other data packet available in its buffer. But for a node who has received some data packets before its first transmission, without FFTS the chance of sending its own data is reduced as more packets are arrived. Thus, nodes that are scheduled late for their first transmission may not send their own data for a long time, resulting in major delays in the whole network awaiting those packets.

Applying FFTS at the first transmission, instead of choosing d random packets from the buffer, the transmitter chooses $d-1$ packets from received data and forms the XOR of these $d-1$ packets with its own data packet to be sent over the channel. Thus, the FFTS forces the propagation of each packet to start as soon as possible.

3.3.2 Proposed Degree Distribution

As was mentioned earlier, the optimal solution of LT codes for one-to-all broadcasting is not optimal for the stated problem where the source is distributed. Having a distributed source makes the number of available packets at each node (the buffer size) time dependant. Therefore implementing a code such as LT is not an option here. But, keeping the ideas of LT code in mind, we need to design a code for the transmitters such that at any receiving node the received degree distribution benefits from the design of LT codes.

Here, due to the distributed and probabilistic nature of the source, it is impossible to obtain the exact same optimal distribution at each receiver. However, we can ensure that the main characteristics of this distribution is preserved. In other words, we seek enough low-degree equations for the purpose of fast decoding and some high-degree equations for the purpose of coverage. To allow efficient optimization, we consider a simple case where the degree distribution has weight only on degrees 1, 2, 3 and $L(i)$ (the buffer size). Therefore, our suggested degree distribution is as follows.

$$\psi(d) = \begin{cases} \alpha_1 & d = 1 \\ \alpha_2 & d = 2 \\ \alpha_3 & d = 3 \\ \alpha_{L(i)} & d = L(i) \end{cases}, \tag{3.2}$$

where $\sum_i \alpha_i = 1$. It is worth noting that if $L(i) < 3$, some of the weights are forced to be zero. For example, if $L(i) = 1$, it is determined that $\alpha_1 = 1$ and all other $\alpha_i = 0$.

Based on the above discussion, we perform a numerical optimization to minimize the total number of transmissions for a given density. In other words, for different sets of weights, the number of transmissions are obtained and the set of weights which minimize the total number of transmissions is of interest to us. So, Eq. (3.3) shows the optimization problem where cost function, N_T, is the number of transmissions. For example, given the density of 0.3 for the network, the results show that $\alpha_1 = 0.35$, $\alpha_2 = 0.05$, $\alpha_3 = 0.5$ and $\alpha_L = 0.1$ generate the best results which are shown in Fig. 3.2(a). This figure shows that based on these weights, the proposed method reduces the total number of transmissions about 18% compared to switched code. Table 3.1 shows the optimal weights for the proposed method that have been calculated for different densities.

$$\begin{aligned} &\min\{N_T(\alpha_1, \alpha_2, \alpha_3, \alpha_L)\} \\ &0 \leq \alpha_i \leq 1 \forall i \\ &\textstyle\sum_i \alpha_i = 1 \end{aligned} \tag{3.3}$$

Table 3.1. Table of optimal weights based on density

Density	α_1	α_2	α_3	α_L
0.2	0.35	0.3	.25	0.1
0.3	0.35	0.05	0.5	0.1
0.4	0.4	0.05	0.45	0.1
0.5	0.5	0.05	0.4	0.05
0.6	0.55	0	0.35	0.1
0.7	0.7	0.05	0.15	0.1
0.8	0.75	0	0.1	0.15
0.9	0.85	0	0.15	0
1	1	0	0	0

Weights $\alpha_1, \ldots \alpha_L$ of the proposed degree distribution for various densities of the network have been calculated through a numerical optimization.

The weights α_1 and α_2 can also be optimized separately for the case that $L(i) = 2$. Our results show that this case is not frequent and happens around 1% of the time, so the choice of these two weights when L(i)=2 does not have a significant effect on the total number of transmissions. Based on our observations, $\alpha_1 = 0.6$ and $\alpha_2 = 0.4$ are appropriate weights for all the densities.

Fig. 3.1 compares the received degree distribution of our codes with that of switched codes for a network with 80 nodes and no erasure when density is 0.2. The purpose of this comparison is to show that compared to switched codes, our optimized degree distribution (at the transmitter side) results in a degree distribution (at the receiver side) which is closer to that of the optimal LT codes.

As we discussed in section 3.2.2, switched codes use LT codes in the encoding process while the length of the data buffer, L is varying. This means that in $\mu(d)$ and consequently in $\sigma(d)$ and $\tau(d)$ in Eq. (2.4) and (2.6), L is not a constant. Therefore, each time a transmission happens, the transmitter has to calculate these distributions again and this causes an extra encoding complexity. On the other hand, if L was constant, this complexity of calculating the degree distribution would be removed because the degree distribution could be stored in a buffer and used whenever needed.

Our proposed degree distribution does not have this varying length issue in the calculation of $\psi(d)$. In other words, varying L does not affect our degree distribution. Thus, it can be saved in a buffer and used anytime a transmission happens. Therefore, in addition to reducing the total number of transmissions, our solution reduces the encoding complexity as well.

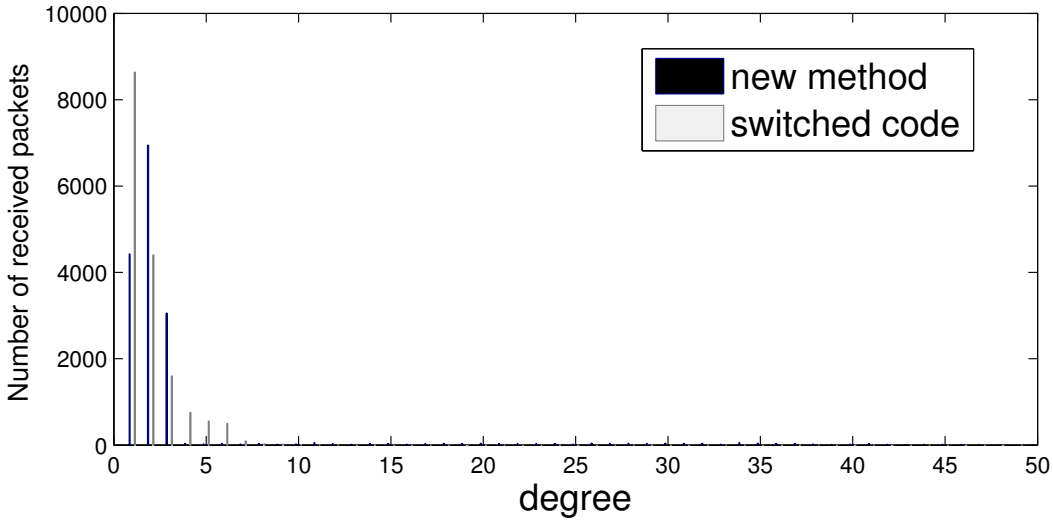


Figure 3.1. Overall degree distribution of two methods.

This figure compares the received degree distribution of the proposed code with that of switched code. These are the packets that are received in the whole network.

3.3.3 Transmitter and Receiver Algorithms

In the following the detailed algorithms are presented, the first part explains the transmitter algorithm while the second part discusses the receiver side.

Transmitter side: Based on a scheduling algorithm in the MAC layer, at each time slot, one or more nodes are determined as the transmitter(s). In the transmitting node, by knowing an estimation of density, first a random number d is chosen based on the $\psi(d)$ distribution from Eq. 3.2 and parameters of Table 3.1. Then, d random packets are chosen from data packets stored in the buffer of the transmitter. The XOR of them is sent as an encoded packet over the channel. When all the neighbors of a node received a certain data packet, that node will not broadcast that packet anymore. Note that for the first transmission of every node FFTS, that was explained earlier, will be applied. Algorithm 1 shows this procedure as well.

Algorithm 1 The Transmitter Side Algorithm

Input: Data packets
Output: One encoded packet
1. Generate a random number, d, based on the $\psi(d)$ distribution from Eq. 3.2 and Table 3.1.
2.
if this is the first transmission of this node **then**
  apply FFTS and choose $d-1$ random data packets from the buffer of the transmitter.
else
  choose d random packets uniformly from data packets stored in the buffer of the transmitter.
end if
3. XOR of selected packets is sent as an encoded packet over the channel.

Receiver side: When an encoded packet arrives, the decoding process starts. Based on the stored data packets in the buffer of the receiver, if the arrived packet cannot be decoded, it will be stored as an encoded packet. Otherwise, the extracted data packets will be stored in the buffer. In the next step, these new arrived data packets will be used to decrease the degree of the stored encoded packets at the receiver if possible. In other words, any new received or extracted data packet that was used in the construction of previous

Algorithm 2 The Receiver Side Algorithm

Input: Encoded received packet
Output: The extracted data packet or equation
if there is more than one unknown data packet in the received equation **then**
the received packet will be stored as an encoded packet in the buffer.
else
the received packet can be decoded and a new data packet is generated.
while there is a new data packet **do**
1. use new data packet to decode previously stored encoded packets in the buffer or decrease their degree if possible
2. extract new data packets if possible and store new data packets in the buffer.
end while
end if

received encoded packets is a known variable now. Therefore, the degree of stored equations can be decreased. This process goes on until the degree of the equations cannot be decreased anymore. All new extracted data packets can be used to construct an encoded packet for the next time that this node acts as a transmitter. Algorithm 2 shows this process of decoding too.

3.4 Numerical Results

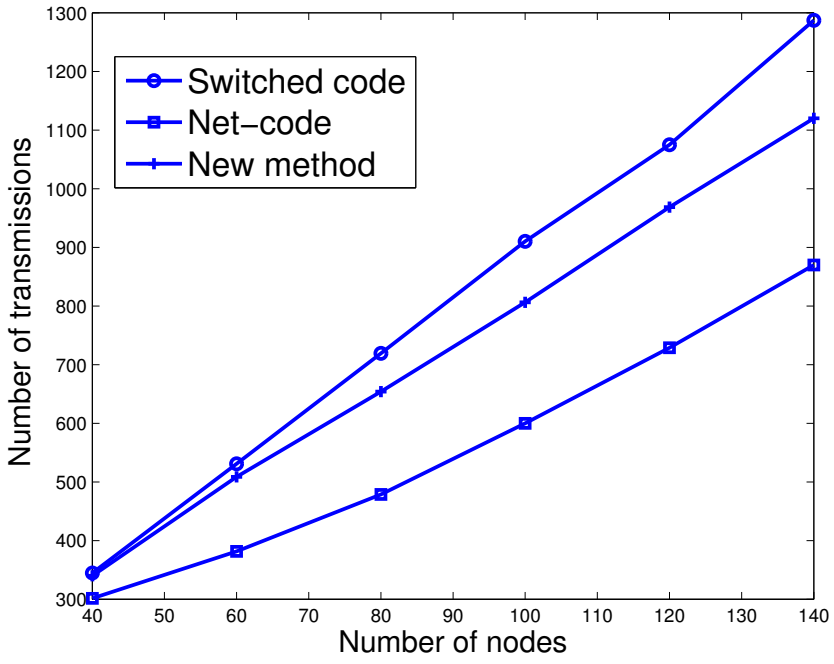
In this section, our proposed method and the existing ones are compared in similar situations. To study these methods, nodes are randomly distributed over the surface of a torus to avoid edge effect. The transmission range of each node is calculated as

$$T = \sqrt{\frac{\rho A}{\pi}}, \tag{3.4}$$

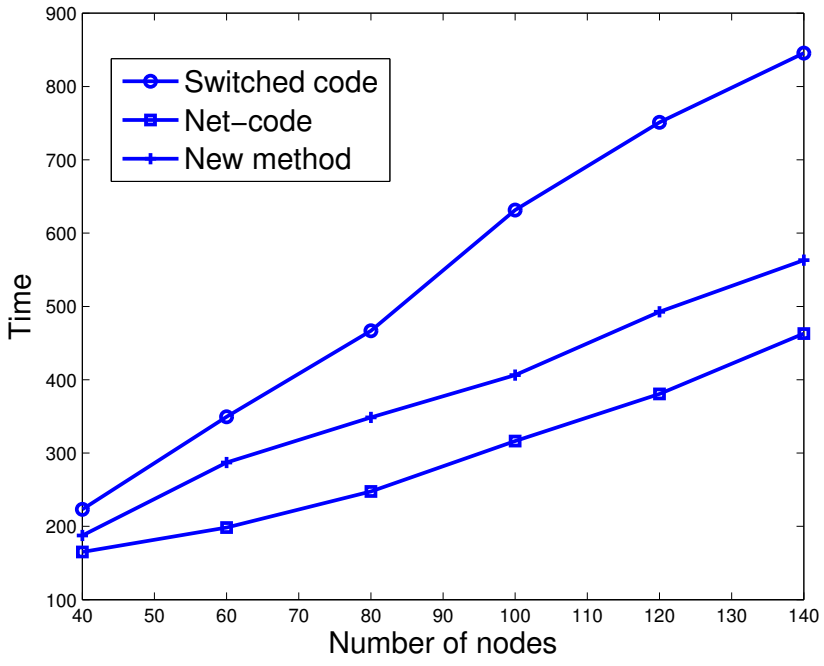
where T is the transmission range of each node, $\rho, 0 \leq \rho \leq 1$ is the density, and A is the area of the network. We assume that the nodes are uniformly distributed over the area, A, so the ratio of $\pi T^2/A$ is the same as density of the network. So, for a fixed area, A, by changing the transmission range of nodes, T, we can change the density of the network. The scheduling in the MAC layer is done by distributed randomized TDMA [37]. At each time slot, one or more number of nodes will be chosen as the transmitter by sending

some handshake messages between a candidate node and its neighbors. It is not necessary to set up the schedule for the whole network at the beginning. The scheduling in the MAC layer can be done in one time slot just for that time slot, or it can be done each time for few number of time slots. In all simulation scenarios the total number of required transmissions to propagate all the packets in the entire network has been measured.

The first simulation compares switched codes, probabilistic network coding (net-code), and the proposed method. As mentioned before, to avoid linear dependency in network coding, high Galois fields are used. Here we use $GF(2^8)$ and we also implement algorithm 6B in [34] which refers to dynamic forwarding factor. Note that the complexity of network coding is higher than the two other methods that apply fountain codes. The results of network coding is presented in Fig. 3.2 as a benchmark. Here, the erasure rate, β is zero, the density is 0.3, and the number of nodes in the network is varying from 40 to 140. Fig. 3.2(a) shows the required total number of transmissions for each method. As it can be seen, the proposed method has less transmissions compared to the switched code and still more than network coding while the complexity is less than the other two methods. Fig. 3.2(b) shows the total delay for each method. It is clear that network coding has the smallest delay, but among the methods with the same complexity of decoding, the proposed method reduces the delay. The gap between switched code and the proposed method becomes more pronounced as the number of nodes increases.



(a) Total number of required transmissions vs. number of nodes



(b) Total delay vs. number of nodes

Figure 3.2. Comparison of network coding, switched code, and the proposed method.

Second simulation shows the effect of density on the system. The number of nodes is 80, with no erasure and the density is changing from 0.2 to 1. As we can see in Fig. 3.3, when the density is increased, the required number of transmissions decreased. However, the number of transmissions will not reach to the exact number of nodes because of random distributed TDMA scheduling. In all cases the proposed method outperforms switched codes.

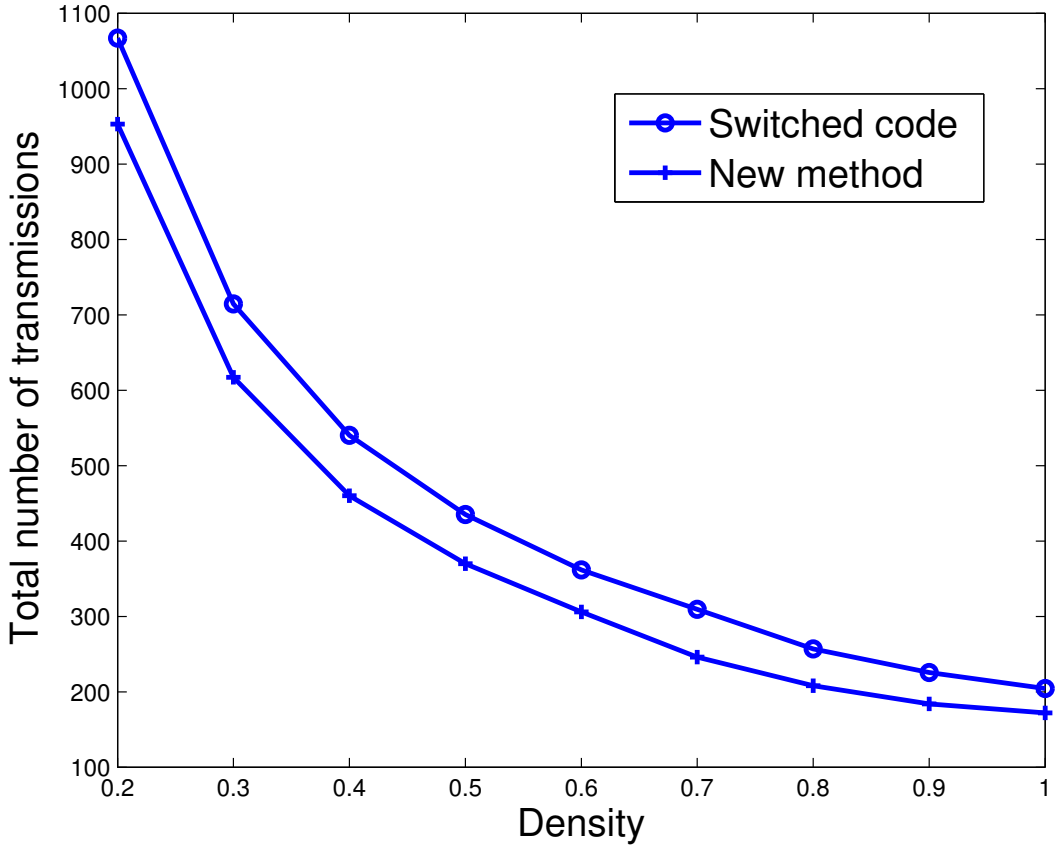


Figure 3.3. Comparison of switched code and proposed method.

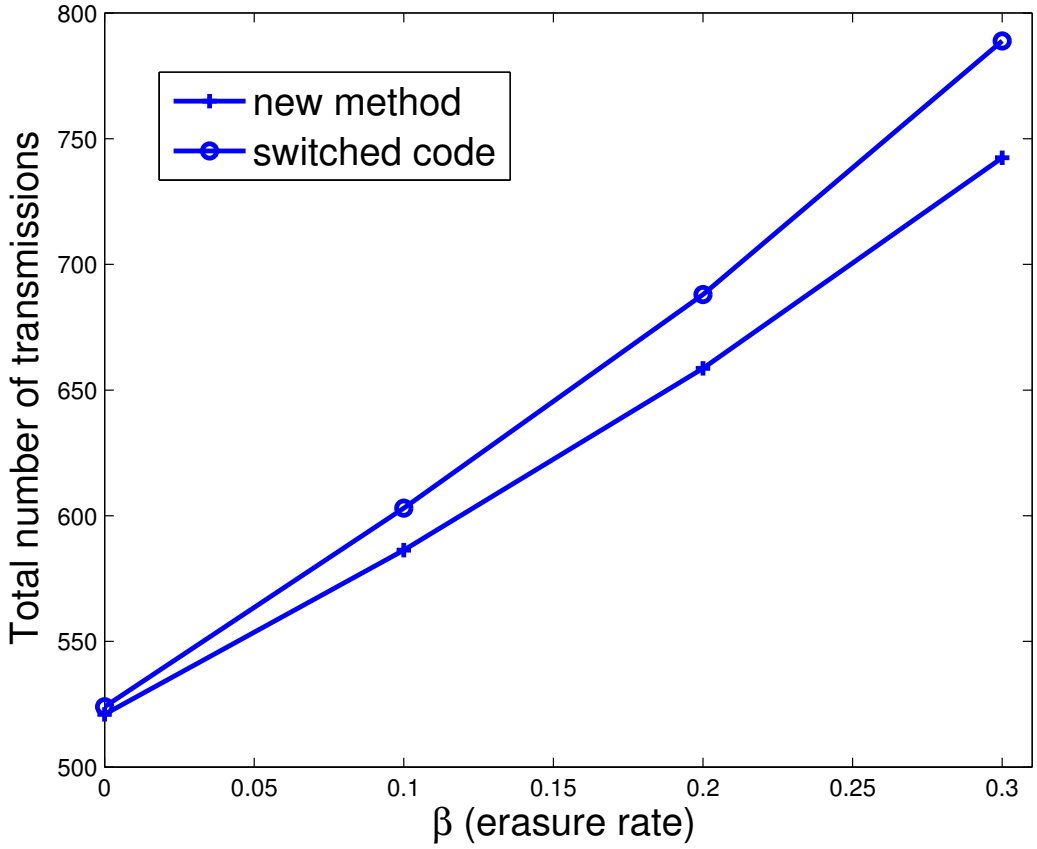


Figure 3.4. Comparison of switched code and proposed method.

In the last simulation, the number of nodes is 60 and density is 0.2. We tested switched codes and the proposed method for a variety of erasure rates and measured the total number of required transmissions to propagate the packets in the whole network. The results in Fig. 3.4 show that under different erasures, the proposed method outperforms the existing method.

3.5 Conclusion

In this chapter we studied the problem of all-to-all nodes broadcasting in wireless ad-hoc networks. In order to reduce the complexity of decoding, fountain codes have been applied. A new degree distribution was proposed and the simulation results showed that it reduces the delay and improves energy efficiency compared to the existing switched code that has the same order of complexity for decoding, while our method reduces the complexity of encoding as well.

Chapter 4

Reduced-overhead Multicasting of Different QoS Data Classes

In this chapter the problem of multicasting using fountain codes is explained. Here, two solutions are studied and their overhead based on the finite block length analysis has been calculated. In the following an introduction to this chapter is provided in Section 4.1. In Section 4.2, the system model and the problem statement are explained. Section 4.3 will discuss problem solutions, starting with a review of a method called SDI, discussing the effect of finite block length on the system, and then discussing another solution, MTS method. Section 4.4 provides the numerical results, where MTS method is compared with SDI. This chapter is concluded in Section 4.5.

4.1 Introduction

Broadcasting data to a number of users with different channel quality can efficiently be done using fountain codes [6]–[8]. The transmitter, however, must continue the transmission of a data block until all the receivers can decode it. Only at this point, the transmitter can start transmission of the next block of data. In this setup, although, the best user can receive one block very quickly, it must remain idle until the transmission of the next block has started.

In some applications, such as multimedia, different quality of service (QoS) data classes can be defined to be sent to different users. More specifically, users

can be classified according to their reception quality to receive different quality multimedia. Since transmissions are done in packets, the user quality can be defined based on the packet erasure rate.

A user with a low erasure rate can potentially receive more QoS data classes than a user with a higher erasure rate. This is in contrast with slowing down the high quality users by the user with worst channel quality. As a result, users that experience good channel quality can receive high quality multimedia during the same time that users with poor channel condition receive the same multimedia at a lower quality. Another example is in vehicular networks [1]. All the vehicles in a highway can receive the necessary information about the traffic and road conditions while those with better channel condition may receive other data classes such as data for entertaining applications or their requested data.

Here, we study the problem of multicasting different QoS data classes on erasure channels. The source node has data packets from different QoS classes intended for users that experience various erasure rates. Users with smaller erasure rates can receive more packets than users with poorer channel conditions. Different solutions have been suggested using rateless codes for multimedia multicasting with different QoS data classes. For instance, [38], [39] propose expanding window fountain codes. This method, however, alters the degree distribution of the fountain code whose increased complexity of decoding is not desirable.

References [40], [41] introduce a solution to this problem using fountain codes and a scheduling algorithm based on data interleaving. Since this method is based on scheduling and data interleaving, we will refer to it as the SDI method. In this solution, there are two data classes where users with good channel quality will receive data from both classes. Users with poor channel quality will only receive one class of data. Since SDI is based on scheduling, data from different data classes are transmitted separately. As we will see later, for applications with small data block sizes such as realtime applications, this data separation may result in a considerable overhead needed to guarantee a certain probability of successful transmission. The source of

this overhead is the fact that the behavior of the erasure channel may vary significantly from its average when used over a small block length.

Another solution to this problem that will be discussed in this chapter is to mix the data of all QoS classes together [42]. In other words, the source can transmit the encoded data of all classes together over all time slots. We will refer to this method as MTS (M-tuple symbols). This idea is used in [42] to formulate and solve the allocation problem of source bits of different data classes based on a defined cost criterion.

In this chapter, we investigate the effect of this approach on the overhead. For this purpose, we first provide a finite block length analysis. By using this analysis, we compare MTS with SDI method in different erasure rates to show that MTS enjoys a lower overhead. Thus, for applications such as multimedia which usually have a short block length, MTS should be the method of choice.

4.2 System Model and Problem Statement

Consider a source that multicasts data to R users. In multicasting, a direct transmission from the source is not a desirable solution. This is because, even if one of the users fails to receive the packet, the packet must be retransmitted. Thus, as the number of users increases, direct transmission becomes less desirable. It is well known that fountain codes can avoid this problem [6]–[8]. With fountain codes, the source continues the transmission until the data for all intended users is provided. Another benefit of fountain coding over direct transmission is that instead of needing a feedback per each received packet per user, it only needs a feedback at the end of reception of the whole block by each user. Moreover, fountain codes can handle users with various or even unknown erasure rates. Thus, here we assume that data is transmitted using fountain codes.

The R users experience different erasure rates. Therefore, they can be classified into L different classes based on their erasure rates where $L \leq R$. Users in the same class are assumed to have equal erasure rate [1].

[1] In practice, users with almost equal erasure rates are grouped together

Let $\beta = e_i$ be the erasure rate of users in class i. The receiving rate can then be defined as $r_i = 1 - e_i$. We also order user classes according to their erasure rate such that for any two user classes $i, j \in \{1, \ldots, L\}$, $i < j \Rightarrow e_i > e_j$. A typical user in class $i \in \{1, \ldots, L\}$ will be referred to as u_i. With this definition, if $i < j$, potentially u_j can receive more packets than u_i during the same period of time. Therefore, different QoS classes can be defined for users with various erasure rates.

Multimedia streaming is an application of the setup described above because different QoS levels of data can naturally be defined since users can receive multimedia with different qualities. Suppose the data stream is split into M classes, $C_1, C_2, \ldots, C_M$. Here, C_1 is the part of stream which provides the lowest quality multimedia and therefore necessary for any user who wants to receive the multimedia stream. Packets in $C_m, m \geq 2$ are intended for users that want higher quality of service. So, the more QoS classes a user receives, the better the quality of the stream it receives. The best quality is provided to the users that receive all M classes. Similar discussions are valid when these data classes are completely independent and from various applications with different priorities.

Without loss of generality we assume $L = M$. In other words, the number of classes of users is equal to the number of different QoS classes of data. With this assumption, by class i, we mean the users whose erasure rate is e_i and expect to receive data from classes $C_1, C_2, \ldots, C_i$. We also assume that all data classes are greedy, meaning that they always have data to send.

Similarly, one can view this as a system, where M data classes are broadcasted on the channel. Each data class has a predefined erasure threshold, and its data is intended for any user whose erasure rate is below the threshold. In this view, e_m represents the predefined threshold for class m. Notice that in this setup, the transmitter does not need to know the erasure rates of users. Moreover, users are naturally classified to different classes according to their erasure rates.

For the defined system, the problem is to devise a data transmission algorithm that $\forall i, 1 \leq i \leq M$ provides $C_1, \ldots, C_i$ to u_i with a failure rate

guaranteed to be less than δ for all user classes. A failure at user class i is defined as not having enough received data from data classes $C_1, \ldots, C_i$ to be able to decode the data of all i classes.

Here, we assume erasures are the only reason for packet loss, so our results are valid for memoryless erasure channels.

The next section explains two solutions to this problem. One solution is SDI method and the other is the MTS method. To motivate MTS solution, the effect of finite block length on the system is also studied in the next section.

4.3 Problem Solution

As mentioned earlier, to handle the problem of various erasure rates of different users, the methods discussed in this section use fountain codes [6]–[8]. When N data packets are fountain coded, any user who received $N' = (1 + \epsilon)N$ encoded packets can decode the N data packets. Here, ϵ is the overhead of the fountain code and is typically due to the linear dependency of some received encoded packets and the suboptimal decoding. Since all methods discussed in this section use fountain codes, we treat N' as the block size when comparing these methods. In other words, the block size is defined as the number of encoded packets needed at the user side. This way, we can compare different methods without the need to consider the fountain coding. From this point of view, by one bit in data class m we mean a fountain encoded bit of this data class. In the remainder of this work, we use N instead of N' for the ease of notations.

4.3.1 SDI Method

For the defined system with M data classes and different erasure rates, [40] proposed an interleaving based method to transmit data. In this method, each class has its own Raptor [8] encoder and therefore data of each layer (class) will first be encoded internally. At each time slot, class $m \in \{1, \ldots, M\}$ will be chosen with probability γ_m. Then an encoded packet from this class will be broadcasted over the channel. [40], [41] optimize probabilities $\gamma_1, \ldots, \gamma_M$

based on the channel erasure rates. The results show that these probabilities are proportional to the erasure rate of their corresponding classes.

Please note that in the time slots that the source transmits packets from C_i, $u_j, \forall j < i$, is in idle mode. Moreover, since u_j has a higher erasure rate than u_i, the total number of packets transmitted from C_j is more than what u_i needs. Thus, even u_i will be in idle mode for a portion of time when packets from C_j are transmitted.

Before discussing MTS [42], we provide a finite block length analysis of the system. While MTS is proposed in [42] for the first time, it is suggested for solving the allocation problem of various data classes. Here, we discuss another advantage of MTS, i.e., its lower overhead compared to SDI. For this purpose, we first need a finite length analysis of the system. Also, in order to make this advantage more clear, we will review MTS from a new point of view in Section 4.3.3.

4.3.2 Finite Block Length

On an erasure channel, erasures happen randomly and independently. For finite block length, the actual number of erasures may differ from the average expected number. Thus, if we need N received packets at the output of a channel with erasure rate e, $N/(1-e)$ transmissions may not be enough. In fact, to guarantee N received packets with high probability, the number of transmissions must be larger than $N/(1-e)$. Thus, the number of extra packets needed can be define as the transmission overhead. This overhead is especially important and fairly large in applications that have small block sizes such as realtime applications. Please note that this overhead is different from the overhead of fountain codes that we discussed earlier.

We previously defined failure for user in class i, $i \in \{1, \ldots, M\}$ as "u_i does not receive enough encoded packets to decode the whole block of data from classes $1, 2, \ldots, i$." Then, to guarantee a probability of failure smaller than δ, one can find the needed transmission overhead of each data class.

Now, consider the data class m, with data blocks of size N_m packets. For users in any class l, $l \geq m$ that experience channel erasure rate e_l, the num-

ber of received packets X_l from data class C_m after K_m transmissions is a Binomial$(K_m, 1-e_l)$ random variable. For guaranteed transmission of C_m to user class l, we wish to have $X_l \geq N_m$ with probability at least $1-\delta$ or equivalently

$$p[X_l < N_m] < \delta.$$

When N_m is larger than a few hundreds, this Binomial distribution can accurately be approximated with a Gaussian distribution. Thus, $X_l \sim \mathcal{N}(K_m(1-e_l), K_m e_l(1-e_l))$ and $p(X_l < N_m)$ can be found using the Q function. Thus, the reception condition for u_l is

$$Q\left(\frac{K_m(1-e_l)-N_m}{\sqrt{K_m e_l(1-e_l)}}\right) < \delta. \tag{4.1}$$

This means that for finite block length, the number of transmissions K_m must be larger than $N_m/(1-e_l)$. Thus, a transmission overhead representing the number of extra encoded packets (compared to the expected number) can be defined as

$$k_{l,m} = K_m - \frac{N_m}{1-e_l}.$$

Although u_l is supposed to receive data from C_1 to C_l, since its erasure rate e_l is smaller than $e_1, e_2 \ldots, e_{l-1}$, the overhead considered for those user classes would satisfy the reception condition of u_l. Thus, among the users that receive data from data class C_m, i.e., $u_m, u_{m+1}, \ldots, u_M$, the highest erasure rate belongs to u_m. As a result, u_m needs the largest overhead among all user classes that need C_m. In other words, for each data class m we only need to satisfy the reception condition for u_m. Thus, the actual needed overhead for class m is

$$k_m = K_m - \frac{N_m}{1-e_m},$$

where K_m can be found using

$$Q\left(\frac{K_m(1-e_m)-N_m}{\sqrt{K_m e_m(1-e_m)}}\right) < \delta. \tag{4.2}$$

The overall overhead is the sum of the overheads of each data class, which can be found as

$$k_{\text{total}} = \sum_{i=1}^{M} k_i \tag{4.3}$$

Clearly as the block length increases, the needed overhead compared to the block length becomes smaller. For small block length, however, the transmitting time of overhead can be a significant portion of the total transmission time. We like to emphasize that by sending k_m extra encoded packets for each data class m, the transmitter guarantees a probability of failure less than δ.

Now let us define a time slot, as the time period needed to transmit one packet over the channel. Here, the total of K time slots are available where $K = K_1 + \ldots + K_M$. In SDI method, these time slots are first divided among M data classes, meaning that the transmission time allocated to data class m is $\gamma_m K$. Depending on the number of data classes, $\gamma_m K$ can be significantly smaller than K resulting in a significant needed overhead to combat the finite block length effect.

Our main insight in this work is to reduce the overhead by sending the data of all classes over all time slots, so that the needed overhead will be calculated for time K instead of $\gamma_m K$. As a result, the needed overhead is for a much larger data block and therefore is smaller. The details of this idea is provided in the next section.

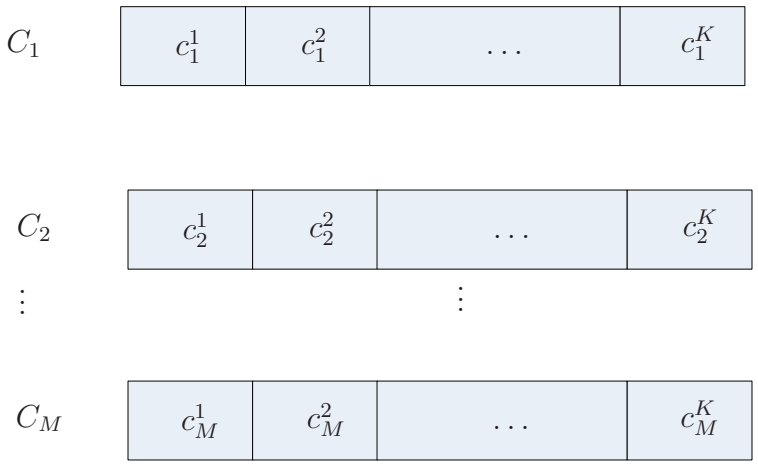
4.3.3 The MTS Method

Considering M QoS data classes, $C_1, C_2, \ldots, C_M$, let c_i^j represent the jth encoded bit of data class C_i. We define symbol s_j as an ordered M-tuple constructed from fountain encoded bits of all data classes i.e. $s_j = (c_1^j, c_2^j, \ldots, c_M^j)$. A symbol in this method is the smallest unit of data from which a packet is formed. A transmitted packet, therefore, contains $\lfloor P/M \rfloor$ symbols where P is the size of a packet. Fig. 4.1(a) shows different data classes and Fig. 4.1(b) shows a symbol s_j and a transmitted packet p in this method. We call this method MTS, because it works with M-tuple symbols.

In MTS, similar to SDI, each data class has its own fountain encoder. In other words, the data of each QoS data class is fountain encoded separately and then the encoded bits are used to generate symbols. Thus, as long as each user receives enough number of encoded packets from a data block of a certain data class, it will be able to decode the whole block. This means that

our method does not effect the coding part. Thus, comparisons can be made without considering the effect of fountain codes.

Although in MTS the same number of bits from all classes are put into each packet, it does not mean that users of all classes are receiving the same amount of data. This is because, each class is working independently and the symbols are created by encoded bits (not raw bits). To clarify this point, let us consider data class C_m. The number of transmitted encoded bits is determined based on three factors. The first one is the data block length V_m (similar to N_m in SDI method) which is determined based on the requirements of the application. The second factor is the erasure rate e_m of the worst case user that can receive this data. So, on average, $\frac{V_m}{1-e_m}$ encoded bits are required to be transmitted, but as we discussed in Section 4.3, for guaranteed QoS, the actual number of needed transmissions is $K'_m = \frac{V_m}{1-e_m} + k'_m$, where k'_m is the overhead which depends on the acceptable probability of failure δ and the data block length. Here, K'_m which is the total transmission time for class m, is equal to K for $m = 1, \ldots, M$, because encoded bits of any class are present in all K transmitted packets.



(a) Blocks of data from different classes, $C_1, C_2, \ldots, C_M$ which contain encoded bits.

(b) A generic symbol, s_j, and a transmitted packet, p, which is formed by defined symbols.

Figure 4.1. Generating transmitted packets in MTS method

To summarize, $\lfloor P/M \rfloor K$ fountain encoded bits from C_m are used in the construction of symbols during K time slots which include overhead bits as well. As soon as these symbols are constructed, the next data block from C_m will be used for construction of new symbols. Any other data class is performing a similar procedure in parallel and independently from class m. As we mentioned, it is assumed that all data classes are greedy and always have data to send. To clarify, Fig. 4.2 shows an example of different data classes, where each class (C_i) has a block of data (V_i) plus the needed overhead (k'_i). As we can see, depending on the various erasure rates, the length of the overhead of different data classes are not the same. This figure also shows how symbols and consequently packets are constructed in MTS method.

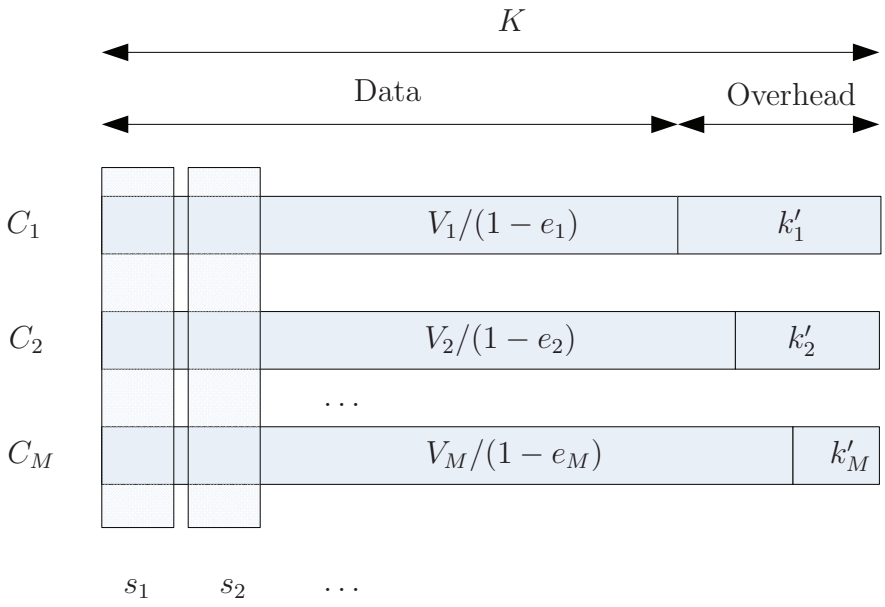


Figure 4.2. The frame shows overheads and data of different classes for a fixed period of time K. It also shows the construction of symbols in MTS method.

The benefit of MTS, as discussed earlier, is that data class m, instead of being sent over only $\gamma_m K$, is sent over K, i.e., all time slots. This way, the data experiences an erasure channel whose behavior is closer to its average. Thus, the needed overhead for guaranteed reception is reduced. By using Eq. (4.2) for MTS, we have

$$Q\left(\frac{K(1-e_m)-V_m}{\sqrt{Ke_m(1-e_m)}}\right) < \delta. \tag{4.4}$$

and V_m can be calculated for each class m. It is important to note that in Eq. (4.4) and in Fig. 4.2 , k'_m and V_m are number of bits not packets. This should be considered when comparing the overhead of two methods. Having V_m and K, the overhead of class m can be found as

$$k'_m = K - \frac{V_m}{1-e_m},$$

where k'_m is the number of overhead bits from data class C_m. To compare the overhead of MTS with that of SDI, the total number of overhead symbols for MTS is calculated as the average of k'_i's which is

$$k'_{\text{total}} = \frac{\sum_{i=1}^{M} k'_i}{M} \tag{4.5}$$

The average arises since different data classes have various size overheads. The numerical results in the next section verify that MTS reduces the overhead of SDI. Moreover, MTS does not need any optimization and the implementation is fairly simple.

4.4 Numerical Results

In this section, we numerically compare the overhead of MTS and SDI methods where the number of data classes are $M = 2$ and $M = 3$ and the total transmission time, i.e., data plus overhead is equal for both methods. We consider total transmission time as $K = 1000$ and $K = 2000$ time slots. For a certain δ and a fixed K with various erasure rates, the overhead is found. Here we added the constraint that after K transmissions all classes should be

done by transmission of one block. Thus, the block size of different classes vary. This has no effect on the overhead comparisons of these two methods, and is merely done for the ease of comparison.

In SDI, the overhead for data class m, k_m, $m = 1, \ldots, M$ is found for each class separately and the overall overhead is obtained from Eq. (4.3). For MTS method, by using Eq. (4.5) the overhead can be calculated as well.

Table 4.1 provides the results for $M = 2$ and $K = 1000$. (r_1, r_2) is the receiving rate of users in class one and two respectively. Table 4.2 represents the same results for $K = 2000$ and Table 4.3 shows the results for $M = 3$ and $K = 1000$ where in all tables r_i shows the receiving rate of class i. To compare these methods more directly, we also report the percentage of overhead reduction (OR) by MTS. If k is the overhead of SDI and k' is the overhead of MTS, then the overhead is reduced by

$$\mathrm{OR} = \frac{\mathrm{k} - \mathrm{k}'}{\mathrm{k}}$$

The results shows that the needed overhead of MTS method is smaller than SDI method in all three cases. As expected, by increasing K, the ratio overhead/K for both methods is reduced. For asymptotically large K, since the channel behavior converges to its average, the overhead compared to the data block size will be negligible for both methods.

Table 4.1. The overhead of SDI and MTS methods for various receiving rates of data class 1 and 2, (r_1, r_2), when $K = 1000$ and $M = 2$ and the percentage of Overhead Reduction

(r_1, r_2)	$(0.4, 0.45)$	$(0.4, 0.65)$	$(0.4, 0.83)$	$(0.5, 0.63)$	$(0.5, 0.75)$	$(0.5, 0.93)$	$(0.6, 0.71)$	$(0.6, 0.85)$	$(0.6, 0.93)$	$(0.7, 0.81)$	$(0.7, 0.89)$	$(0.7, 0.95)$
k_{SDI}	231	193	162	174	153	114	141	116	95	106	90	62
k'_{MTS}	158	132	110	119	105	79	97	80	66	74	62	43
OR%	32%	31%	31%	31%	31%	31%	31%	31%	31%	30%	30%	30%

Table 4.2. The overhead of SDI and MTS methods for various receiving rates of data class 1 and 2, (r_1, r_2), when $K = 2000$ and $M = 2$ and the percentage of Overhead Reduction

(r_1, r_2)	$(0.4, 0.45)$	$(0.4, 0.65)$	$(0.4, 0.83)$	$(0.5, 0.63)$	$(0.5, 0.75)$	$(0.5, 0.93)$	$(0.6, 0.71)$	$(0.6, 0.85)$	$(0.6, 0.93)$	$(0.7, 0.81)$	$(0.7, 0.89)$	$(0.7, 0.95)$
k_{SDI}	324	270	226	243	215	160	198	163	134	149	126	88
k'_{MTS}	223	187	157	169	149	111	137	113	93	104	88	61
OR%	31%	31%	31%	31%	31%	31%	30%	30%	30%	30%	30%	30%

Table 4.3. The overhead of SDI and MTS methods for various receiving rates of class 1, 2, and 3, (r_1, r_2, r_3), when $K = 1000$ and M=3 and the percentage of Overhead Reduction

(r_1, r_2, r_3)	$(0.3, 0.47, 0.6)$	$(0.3, 0.53, 0.72)$	$(0.3, 0.69, 0.94)$	$(0.4, 0.55, 0.63)$	$(0.4, 0.65, 0.73)$	$(0.4, 0.67, 0.9)$	$(0.5, 0.63, 0.76)$	$(0.5, 0.65, 0.84)$	$(0.5, 0.71, 0.88)$
k_{SDI}	300	266	178	255	241	194	239	203	194
k'_{MTS}	162	144	98	139	131	107	130	111	106
OR%	46%	46%	45%	45%	45%	45%	45%	45%	45%

These three tables are the numerical results to show the comparison between the two methods.

4.5 Conclusion

We studied the problem of transmitting different QoS data classes to users with various erasure rates. Each data class was intended for any user whose erasure rate was better than a predefined threshold. Since for some applications such as multimedia small block length is needed and in that case the number of erasures introduced by the channel can significantly be greater than its average, a large overhead may be needed for acceptable probability of success.We studied the effect of the block length on the overhead and provided an analysis to compare different solutions in terms of their overhead. Our results showed that MTS requires a much lower overhead compared to SDI.

Chapter 5

Conclusion

In this thesis, we studied multicasting and broadcasting using fountain codes and network coding. In Chapter 3 the problem of broadcasting from all-to-all nodes has been studied. One solution to reduce the number of required transmissions is network coding which is energy efficient. However, the cubic complexity of decoding in each node is a problem, especially when nodes do not have enough processing capabilities. To reduce the complexity, fountain codes as a solution has been considered.

Compared to network coding, fountain codes need more transmissions in general. However, our suggested degree distribution for fountain codes reduces the number of transmissions compared to the previous method [35], [36] that used fountain codes and fills the gap between the previous proposed fountain codes and network coding. It also has the lowest complexity between all these three methods. Therefore, using this fountain code reduces the number of transmissions and complexity and consequently, reduces the required energy consumption for distributing the data in the entire network.

The problem of multicasting data from one transmitter to a number of users has been studies in Chapter 4. Different data classes were transmitted for users where each group of users experiences different erasure rates. The goal was to reduce the overall transmission time by reducing the overall overhead. By using fountain codes at the transmitter and mixing data packets of all classes together, the total overhead is reduced and the analysis showed that in various erasure rates this method can reduce the required overhead for the guaranteed

probability of success. This method is specially useful where the block length is finite such as multimedia applications. As the block length goes to infinity, the overhead compared to the block size becomes negligible and both discussed methods have almost the same result.

In general, as we expected using both network coding and fountain codes help reducing the number of transmissions and improving the energy efficiency and delay of the system while using fountain codes has less complexity compared to network coding.

5.1 Future Work

To extend this research there are some suggestions. For the case of broadcasting form all-to-all nodes, one possible work is to analytically find the optimal degree distribution in Chapter 3. Therefore, the optimal weight for each degree can be calculated and our numerical results for the proposed degree distribution can be justified.

The degree distribution can also be considered as a general distribution where instead of having weights only on four degrees, other degrees can have non-zero weights as well. In that case, the effect of all possible weights on the total number of transmissions and the dependency of results to the density can be analyzed as well.

The case of multicasting different quality of service data classes in Chapter 4 can be extended too. Instead of constructing each symbol using one bit of each class, we can have symbols where different classes have different number of bits in one symbol based on their requirements.

The idea of weighted fairness while compared to the round-robin [43], [44] in fair queuing in the computer networks can be helpful to construct weighted symbols as well, which is the same as comparing weighted symbols to the M-tuple symbols. The effect of weighted symbols in the case of erasure should be considered and overhead should be recalculated as well.

Bibliography

[1] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE Communications Surveys Tutorials*, 2011.

[2] C. Diot, J. Scott, and E. Upton, "The haggle architechture," Intel Reseach Cambridge, Tech. Rep., 2004.

[3] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "A network coding approach to energy efficient broadcasting: From theory to practice," in *25th IEEE International Conference on Computer Communications.(INFOCOM)*, Barcelona, Spain, Apr. 2006, pp. 1–11.

[4] L. Zhaohua and G. Mingjun, "Survey on network lifetime research for wireless sensor networks," in *2nd IEEE International Conference on Broadband Network Multimedia Technology, IC-BNMT '09.*, 2009.

[5] D. Wu, Y. Hou, W. Zhu, Y. Zhang, and J. Peha, "Streaming video over the internet: Approaches and directions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282 –300, Mar 2001.

[6] D. MacKay, "Fountain codes," *Communications, IEEE Proceedings-*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[7] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS)*, Vancouver, BC, Canada, Nov. 2002, pp. 271–280.

[8] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[9] C. Fragouli, J.-Y. B. Le, and J. Widmer, "Network coding: An instant primer," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, Jan. 2006.

[10] N. Rahnavard and F. Fekri, "CRBcast: A collaborative rateless scheme for reliable and energy-efficient broadcasting in wireless sensor networks," in *The Fifth International Conference on Information Processing in Sensor Networks, IPSN*, Nashville, TN, Apr. 2006, pp. 276 –283.

[11] Z. Abdeyazdan, M. Ardakani, and C. Tellambura, "Energy efficient broadcasting in wireless ad-hoc networks," in *Submitted to IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013.

[12] ——, "Reduced-overhead multicasting of different quality of service data classes," in *Canadian Conference on Electrical and Computer Engineering (CCECE)*, Apr. 2012.

[13] A. Shokrollahi, "LDPC codes: An introduction," *Digital Fountain, Inc., Tech. Rep*, p. 2, 2003.

[14] S. Aly, Z. Kong, and E. Soljanin, "Fountain codes based distributed storage algorithms for large-scale wireless sensor networks," in *Proceedings of the 7th international conference on Information processing in sensor networks*, ser. IPSN '08, 2008, pp. 171–182.

[15] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Scalable video multicast using expanding window fountain codes," *IEEE Transactions on Multimedia*, vol. 11, no. 6, pp. 1094 –1104, oct. 2009.

[16] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204 –1216, jul 2000.

[17] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *IEEE International Symposium on Information Theory*, june-4 july 2003, p. 442.

[18] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413 –4430, oct. 2006.

[19] S. Lin and J. D. J. Costello, *Error Control Coding*. Pearson Education, Inc, 2003.

[20] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, ser. SPAA '03, 2003, pp. 286–294.

[21] Y. Wu, P. Chou, and K. Jain, "A comparison of network coding and tree packing," in *International Symposium on Information Theory, ISIT*, June-2 July 2004.

[22] P. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Allerton*, 2003.

[23] Y. Li, E. Soljanin, and P. Spasojevic, "Collecting coded coupons over overlapping generations," in *IEEE International Symposium on Network Coding (NetCod)*, Jun. 2010.

[24] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371 –381, feb. 2003.

[25] G. Blom, L. Holst, and D. Sandell, *Problems and Snapshots From the World of Probability*. New York Springer-Verlag, 1994.

[26] S. Deb and M. Medard, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," in *Allerton*, Oct. 2004.

[27] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 4, march 2005, pp. 2235 – 2245 vol. 4.

[28] X. Zhang, J. Liu, B. Li, and Y., "Coolstreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3, march 2005, pp. 2102 – 2111 vol. 3.

[29] D. Petrovic, J. K. Ramchandran, and Rabaey, "Overcoming untuned radios in wireless networks with network coding," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2649 – 2657, june 2006.

[30] C. Fragouli and A. Markopoulou, "A network coding approach to overlay network monitoring," in *In Allerton*, 2005.

[31] T. Ho, B. Leong, Y.-H. Chang, Y. Wen, and R. Koetter, "Network monitoring in multicast networks using network coding," in *International Symposium on Information Theory, ISIT*, sept. 2005, pp. 1977 –1981.

[32] N. Cai and R. Yeung, "Secure network coding," in *IEEE International Symposium on Information Theory*, 2002.

[33] H. Lim and C. Kim, "Flooding in wireless ad-hoc networks," *Computer Communications*, vol. 24, no. 34, pp. 353 – 363, 2001.

[34] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "Efficient broadcasting using network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 450 – 463, Apr. 2008.

[35] N. Kadi and K. A. Agha, "Distributed switched code (DiSC): A distributed rateless code for broadcast in ad-hoc wireless networks," in *ACM Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2010.

[36] ——, "New degree distribution to improve LT code in network coding for broadcasting in ad-hoc wireless networks," in *IEEE 21st International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, Istanbul, Turkey, Sep. 2010, pp. 1820–1825.

[37] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed randomized TDMA scheduling for wireless ad-hoc networks," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*.

[38] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk, and R. Piechocki, "Expanding window fountain codes for unequal error protection," *IEEE Transactions on Communications*, vol. 57, no. 9, pp. 2510 –2516, Sep. 2009.

[39] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Scalable video multicast using expanding window fountain codes," *IEEE Transactions on Multimedia*, vol. 11, no. 6, pp. 1094 –1104, Oct. 2009.

[40] C. Yu, S. Blostein, and C. Wai-Yip, "Optimization of rateless coding for multimedia multicasting," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, Shanghai, China, Mar. 2010, p. 1.

[41] ——, "Unequal error protection rateless coding design for multimedia multicasting," in *Int. Symp. on Inform. Theory (ISIT)*, Austin, TX, Jun. 2010, p. 2438.

[42] W. Sheng, W.-Y. Chan, S. D. Blostein, and Y. Cao, "Asynchronous and reliable multimedia multicast with heterogeneous QoS constraints," in *IEEE International Conference on Communications (ICC)*, Cape Town, South Africa, May 2010, pp. 1 –6.

[43] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case,"

IEEE/ACM Transactions on Networking, vol. 1, no. 3, pp. 344 –357, Jun. 1993.

[44] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *13th Proceedings IEEE Networking for Global Communications, INFOCOM '94*, Jun. 1994, pp. 636 –646 vol.2.