

The Implementation of Avalanche Decoding in Random Network Coding Networks

Suné von Solms, Albert S. J. Helberg

TeleNet Research Group

School for Electric, Electronic and Computer Engineering

North West University, Potchefstroom Campus

Tel: (018) 299 1961, Fax: (018) 299 1977

E-mail: sune.vonsolms@nwu.ac.za, albert.helberg@nwu.ac.za

Abstract— We introduce a decoding method based on the low complexity decoding scheme of LT codes, called avalanche decoding. Other than LT codes, where the intermediate nodes must encode packets according to a specific degree distribution, avalanche decoding enables a network to implement random network coding where intermediate nodes may encode packets randomly and independently.

When compared to traditional decoding methods that implement Gaussian elimination, a decrease can be seen in decoding time as well as a significant increase in decoding success.

Index Terms— Decoding, Fountain codes, LT codes, Network Coding, Random Network Coding.

I. INTRODUCTION

In practical networks, the topology of a network is not always known or frequently changing, and maintenance of network coordination can be expensive or impractical [3].

Random Network Coding, introduced in [2], is a decentralized approach to Network Coding, introduced in [1], which can be implemented in networks without the need for network control and planning and knowledge of network topology. Random Network Coding is optimal, since only k channel packets are needed by the receiver to obtain the original k source packets with a very high probability [2].

When traditional erasure codes are implemented in a Random Network Coding network, $(n - k)$ redundant symbols are generated at the source node from the k information packets. A total of n encoded packets are then transmitted over the Random Network Coding network (which acts as an erasure channel), where the receiver node can recover the original message from a received subset of the n encoded packets. The subset needed to recover the original message is $k' \subseteq n$ [2, 7, 9]. The receiver node of such a network would wait until it receives a set of k' channel packets in order to decode the original message of k packets using Gaussian elimination techniques [10].

With this method optimal coding can be achieved, without the need for any coordination between neighbouring nodes [10]. The intermediate nodes in the network only need the information sent to it by its neighbour, which it encodes uniformly at random. The receiver and source nodes do not

need to carry any knowledge of the topology of the network, or how the channel packets are encoded.

Due to the non-deterministic nature of Random Network Coding, the global encoding vectors sent in the packet header provide the information of each packet's content in order for the receiver to decode the data. The lack of knowledge concerning the topology of the network and therefore the content/combination of the channel packets that will be received makes it difficult for the receiver node to know how long it should wait to obtain sufficient channel packets for decoding.

Another problem associated with traditional erasure codes is that the size of the linear block code must be chosen before the encoding process begins and remains fixed throughout the transmission [9]. Also, these codes become impractical as the values of k and n become too large, due to the increase in decoding complexity of Gaussian elimination [4, 7]. In energy constraint networks, such as wireless sensor networks, high computational complexity leads to greater energy consumption, which is not ideal.

In this paper, we aim to address these shortcomings of Random Network Coding. We aim to eliminate the need for a fixed size of code before transmission, as well as the high decoding complexity at the receiver nodes. In Section II, we look at related work that helps us in the development of an effective method. In Section III we introduce the avalanche decoding method and evaluate the method in Section IV. This paper is concluded in Section V.

II. RELATED WORK

A. Fountain Codes

The random linear fountain code, described in [7, 9] is a rateless erasure code which means that from a set of source symbols, potentially endless encoded symbols can be generated as needed by the decoder. This means that more encoded packets can be generated for the decoder if needed to decode the source data.

Random linear fountain codes work as follow [9]: The source sends messages, data containing k symbols, s_1, s_2, \dots, s_k , from the source alphabet, Z in a finite field \mathbb{F}_q . Every clock cycle, n , the encoder generates k random bits, $G_{kn} \in \{0,1\}$ and an encoded packets is generated, consisting of the source symbols for which $G_{kn} = 1$.

$$t_n = \sum_{\kappa=1}^k s_{\kappa} G_{\kappa n} \quad (1)$$

We assume that the receiver knows the global encoding vector of each packet i.e. the original symbols, s_1, s_2, \dots, s_k , that each packet contains. For each new encoded packet the receiver obtains, another column, G_n , is added to the generator matrix, \mathbf{G} .

If the receiver collects less than k encoded packets, it will be unable to decode the source message, due to insufficient received information. Random linear fountain codes have a very low probability of being an optimal code for the erasure channel, because when the receiver collects exactly k encoded packets, successful decoding is possible, but unlikely due to the probability of the occurrence of the same encoded packet more than once. However, if the receiver collects k' encoded packets, the successful decoding of these codes become more likely as k' becomes larger. When k' becomes larger, the probability for the generator matrix, \mathbf{G} , to consist of at least k linearly independent columns increases and successful decoding can take place [9]. When generator matrix \mathbf{G} consists of k linearly independent columns, the matrix can be inverted by Gaussian elimination and the original symbols can be recovered.

$$s_k = \sum_{\eta=1}^n t_{\eta} G_{k\eta}^{-1} \quad (2)$$

A representation of random linear fountain codes can be seen in Fig. 1. The encoder encodes the source packets and transmits them over the erasure channel, where some packets are not received (grey). The collected packets are used by the receiver to construct a generator matrix, \mathbf{G} , for decoding.

Note that the encoding process is only implemented on native source packets, where only the source node is able to generate more encoded packets. This means that once a packet is encoded, it is sent to the receiver node via intermediate nodes that simply forward these packets over the erasure channel [10].

In a Random Network Coding environment, all the intermediate nodes perform Random Network Coding on the packets they receive. This means that each node is able to generate new encoded packets from original or already encoded packets. Due to this network characteristic, random linear fountain codes can be implemented very easily in a Random Network Coding environment. The intermediate nodes of the Random Network Coding network act as the fountain encoder and generate a supply of encoded packets for the receiver node until it has collected sufficient packets which it can use to decode the original message.

A representation of the implementation of random linear fountain codes can be seen in Fig. 2. Encoded packets are generated by intermediate nodes and transmitted over the network to the receiver node. The white packets are successfully transmitted and collected by the receiver node, while the transmission of the grey packets is unsuccessful. The

receiver constructs the generator matrix, \mathbf{G} , from the received packets for decoding.

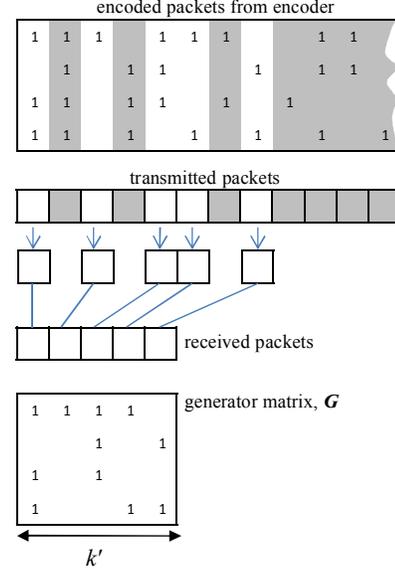


Figure 1: Random linear fountain codes (adapted from [9])

Another advantage of the implementation of this code in a Random Network Coding environment is that there is no fixed, predetermined size for fountain codes, as with traditional erasure codes. In a network where a block code is implemented, the values of k and n of the block code must be determined before transmission, because the source node must encode the k information packets into n encoded packets.

For fountain codes this is not necessary, because the encoded packets can be generated as needed for the receiver node to decode the original data [7, 8]. As with the implementation of traditional erasure codes in such an environment, however, the decoding complexity becomes significantly large as the value of k increases.

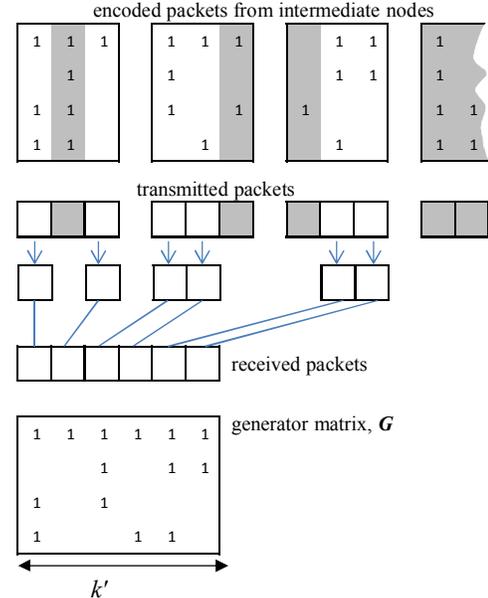


Figure 2: Implementation of random linear fountain codes in a Random Network Coding scenario.

B. LT codes

In 1998 [5], Luby proposed a variation of traditional fountain codes, called the Luby Transform (LT codes). These codes can be seen as sparse linear fountain codes [8] that reduce the decoding complexity associated with random linear fountain codes by eliminating the use of Gaussian elimination at the receiver.

For LT codes, encoded symbols are not generated randomly, but according to a specific degree distribution, $\rho(d)$. The *degree* of an encoded packet is an indication of the number of source symbols present in the packet. For instance: the degree of encoded packet $t_n = \{s_1, s_2, s_4\}$ is $d_{t_n} = 3$.

This probability distribution ensures that certain encoded packets have a low degree to ensure that the decoding process can start, keep going, and that the number of iterations is kept to a minimum. Packets with a higher degree must also be generated to ensure that all packets contain a different collection of source symbols and that all source symbols are included in the received set of encoded packets.[9].

This specific encoding algorithm guarantees efficient LT decoding times, through the use of a low complexity decoding algorithm. This algorithm uses a dedicated data structure instead of constructing the traditional generator matrix [7, 8].

The decoding process of LT codes can be described in the following steps [8].

1. Find an encoded packet, t_n , that is only connected to a single source symbol, s_k .
2. Set $s_k = t_n$.
3. Add the value of s_k to all the encoded packets that is connected to s_k :

$$t_{n'} = t_n + s_k, \forall n \text{ where } G_{kn} = 1. \quad (3)$$
4. Remove source node s_k and disconnect.
5. Repeat from (1) until all $s_i, i \in \{1, k\}$ is determined.

An example of this decoding method can be seen in Fig. 3, where 3 source symbols (bits in finite field, \mathbb{F}_2), are transmitted and 4 encoded packets, $\{t_1, t_2, t_3, t_4\} = \{1\ 0\ 1\ 1\}$, are received, [9].

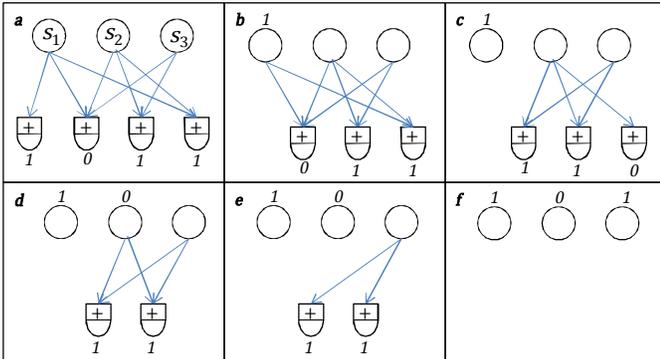


Figure 3: LT code example [9]

In the first iteration the only encoded packet that contains only one source symbol is t_1 , and will be used as the first check node (panel a). It is clear to see that the value of $s_1 = 1$ and the check node is deleted (panel b). The value of s_1 is added to all the encoded packets containing symbol s_1 and the node is disconnected. In the second iteration packet t_4 is the

next check node (panel c). The process repeats where s_2 is set to $t_4 = 0$, the check node deleted (panel d) and the value of s_2 added to the remaining encoded packets and disconnected (panel e). Finally, there exist two check nodes that show the value of $s_3 = 1$.

This encoding distribution differentiates LT codes from random linear fountain codes. Due to the specific encoding distribution of an LT encoder, this decoding algorithm, called avalanche decoding [8], has a very low complexity compared to that of generator matrix construction.

This however is not so easily implemented in a Random Network Coding environment, because encoding must take place according to LT distribution, and not randomly as with random linear fountain codes. Another problem associated with the implementation of LT codes in a decentralized network is that not only source packets get encoded by the nodes. As already encoded packets are sent to another intermediate node, that packet will get recoded with other encoded packets.

A novel approach to reduce the computational complexity of the decoding process for erasure codes in a Network Coding network is introduced by Champel et. al. in 2009 [10]. The authors propose a method, called LT Network Coding, where the intermediate network nodes encode the packets according to the specific distribution of LT codes [5]. Each intermediate node encodes each packet so that its degree distribution matches that of LT codes so that low complexity decoding (belief propagation decoding) can be implemented at the receiver node. This method leads to a significant reduction in decoding complexity, but adds to the overall communication overhead [7, 10].

III. AVALANCHE DECODING IN RANDOM NETWORK CODING

When looking at the implementation of LT Network Coding in [10], the advantages can clearly be seen. In practical networks, however, where the topology is unknown or constantly changing, LT Network Coding can become difficult to implement.

We showed in Section II *a* that the implementation of fountain codes in a Random Network Coding scenario eliminates the need for the length of the code to be fixed as with traditional erasure codes. Intermediate network nodes are able to generate encoded packets on the fly.

We aim to add to the advantage of implementing random linear fountain codes by developing a decoding method based on the low complexity decoding scheme of LT codes to eliminate the need for Gaussian elimination at the receiver nodes.

A. Model

We adopt the notation used in [2, 5, 6] for an acyclic network model. The network is represented by graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} is the set of nodes in the network and \mathcal{E} the set of edges in \mathcal{G} which represents the communication channels. An edge from node v_1 to v_2 is indicated by $(v_1, v_2) \in \mathcal{E}$.

B. Preliminaries

When we look at the construction of the degree distribution designed by Luby [7] we can see that the specific degree distribution is created to have certain important characteristics. These characteristics can also be concluded from the example in Fig. 3. They are the following:

1. The decoding complexity is a function of the connectivity of the graph. This means that in order for decoding to be successful, all the source symbols must be included in the received encoded packets at least once.
2. Reduce redundancy as far as possible. This means that the packets must be encoded in such a way that for each decoding iteration, a single encoded packet has a degree of $d = 1$, and after it is processed a new single-degree packet appears.
3. A single received encoded packet must only contain one source symbol to start the decoding process.

By considering these characteristics, we are going to construct a more general avalanche decoding version and implement it in a Random Network Coding network.

C. Proposed method

The source node, S , arranges the source data of k information symbols, s_1, s_2, \dots, s_k , into a data carousel. A data carousel is a transmission mechanism that arranges the source data in a circular fashion, with each data symbol in a set position in the circle, and the transmission point as a fixed point. As the circle rotates, the data symbols passing the transmission point is transmitted into the network. The transmission of the contents of the carousel is cyclically repeated. This source data is transmitted over a Random Network Coding network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the intermediate nodes form random linear combinations of their inputs, creating encoded packets.

The encoded packets are generated by using randomly generated coefficients g_{ij} from a finite field \mathbb{F}_{2^q} so that

$$t'_i = \sum_{j=1}^k g_{ij} x_j, i = 1, 2, \dots, n. \quad (4)$$

The set of coefficients $g_{i1}, g_{i2}, \dots, g_{ik}$ can be referred to as the *encoding vector* for t_i and this vector is sent as a packet header in order to record the linear combination of the messages present in the received packet [2, 12]. These encoded packets are transmitted to the other intermediate nodes where they are recoded and transmitted once again until eventually collected by the receiver node, R . Network properties, such as connectivity, *min-cut* and the presence of cycles influence the contents and the time of arrival of the encoded packets. As the encoded packets are collected by the receiver node, their encoding vectors are evaluated to determine their degree. As soon the receiver obtains a packet of degree $d = 1$, the decoding process can start.

The avalanche decoding process operates in the following steps for every cycle:

1. Collect new encoded packets from network and store in buffer.

2. Evaluate their encoding vectors to determine their degree.
3. Find an encoded packet, $t_j, j \in \{1, k'\}$, of degree $d = 1$.
4. Avalanche decoding:
 - a. Evaluate encoding vector to determine which source symbol $s_i, i \in \{1, k\}$ is present in the packet.
 - b. Set the source symbol $s_i = t_j$.
 - c. Add (x-or) the value of s_i to all the encoded packets that contains the source symbol s_i (can be seen in encoding vector) to obtain their new value: $t_{n'} = t_n + s_i, \forall n$ where $g_{ni} = 1$ and $t_n \neq t_j$.
 - d. Reduce the degree value of these packets by one: $d' = d - 1$.
 - e. Replace old t_n packets with the updated $t_{n'}$ packets in buffer.
 - f. Evaluate updated packets in buffer for packet, $t_{j'}, j' \in \{1, k'\}$, of degree $d = 1$.
5. Repeat step (4) as long as a packet with degree $d = 1$ is present.
6. Repeat from (1) until all $s_i, i \in \{1, k\}$ is determined.

This process eliminates the need for the receiver to wait until it receives sufficient encoded packets in order to construct an invertible generator matrix. This avalanche decoding method reduces the decoding complexity as well as the speed of decoding.

An example of this method can be viewed in Fig. 4. The following steps are taken by the receiver node to decode source packets $\{s_1, s_2, s_3, s_4\}$.

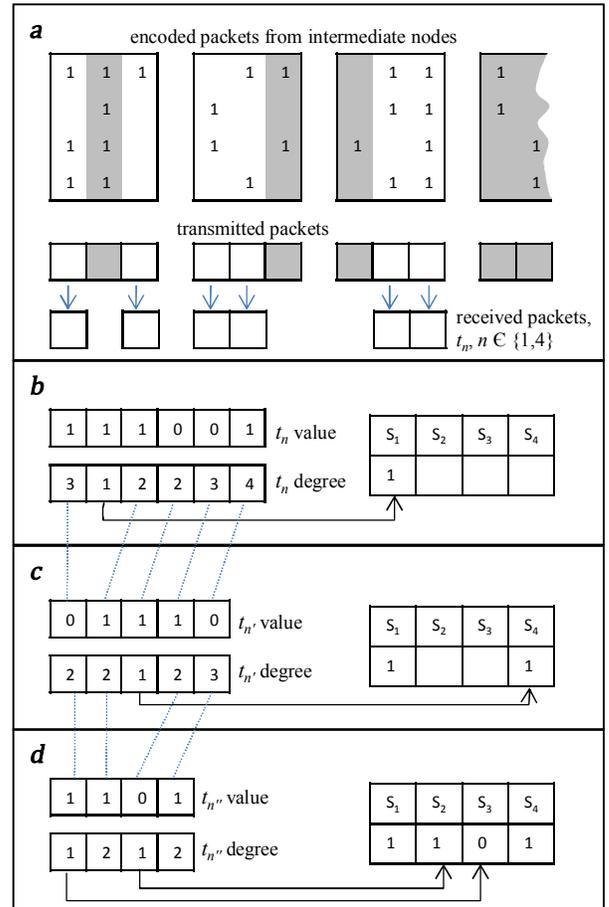


Figure 4: Example of decoding method

Panel a: The packets encoded by the intermediate nodes are transmitted over the erasure channel and collected by the receiver (step 1). *Panel b:* The receiver evaluates the degree of the received packets (step 2), and determines that packet t_2 has a degree of one (step 3). The receiver determines the value of $s_1 = t_1 = 1$ (step 4 a-b). *Panel c:* The receiver adds the value of s_1 to packets that contain the source symbol and reduce their degree by one. The updated packets are now evaluated to determine if a new packet is present with degree $d = 1$ (step 4 c-f). Step 4 a-b is repeated to determine that $s_4 = t_3 = 1$. *Panel d:* Step 4 is completed and iterated again to determine the values of $s_3 = 0$ and $s_2 = 1$. \square

Note: In this example, all the source symbols are transmitted in a single transmission and successful decoding does not require more than a single iteration of the method. In a data carousel, not all source symbols are transmitted simultaneously, but in a fixed periodic sequence. This means that all the source symbols transmitted in the data carousel will not reach the receiver in the first cycle, which will require more than a single iteration.

IV. EVALUATION

In this section we evaluate the decoding performance of the proposed decoding method. In order to do so, we compare it against the traditional decoding method of matrix construction and Gaussian elimination.

A. Experimental setup

We consider a randomly generated network of N nodes and a single source node, S , that transmits the source symbols on a data carousel into the network in a fixed periodic sequence. The intermediate network nodes randomly encode the packets they receive and transmit them to other nodes in the network, including the receiver node.

We randomly generated a set of 1000 Random Network Coding networks with the following properties:

1. Source data from finite field \mathbb{F}_q of length, k .
2. Network size: N
3. Each intermediate node randomly and independently generates a linear combination of its input packets.

In this simulation, we assume that there is no processing delay at the intermediate nodes and that a single unit time is allowed for each packet transmission. This time delay approach is used in [11]. Also note that this network is randomly generated, so the presence of network cycles is not eliminated. An example of one of the generated random networks, with $N = 10$, and $S = 4$, can be seen in Fig. 5. The circles indicate the broadcast area of the nodes and the line if a connection is established.

At the receiver node, two different decoding methods will be implemented, namely traditional decoding and avalanche decoding. The traditional decoding method will instruct the receiver to wait until it receives enough linearly independent encoded packets to create an invertible generator matrix, G . Through Gaussian elimination it will then determine the values of the original source symbols.

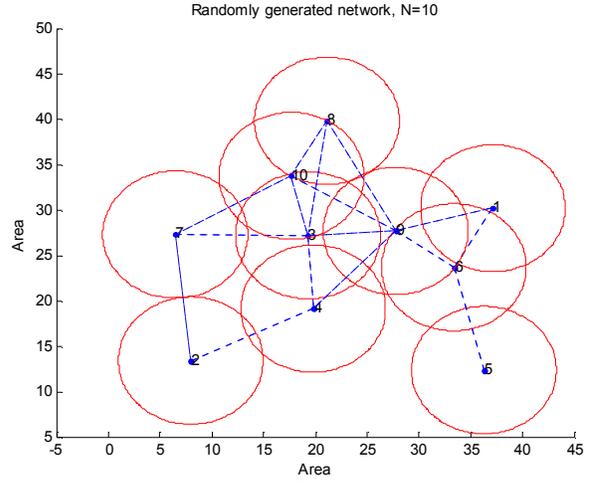


Figure 5: Random Network Coding network

For the avalanche decoding method, the receiver follows the steps discussed in Section III c. As soon as the receiver collects an encoded packet with degree $d = 1$, the avalanche decoding process is started.

Note: Due to the random generation of the networks as well as the absence of a specific encoding degree distribution, the encoded packets collected by the receiver may not adhere to the specifications for successful LT decoding, named in Section III b. This means that successful avalanche decoding is not guaranteed in the networks generated for this simulation.

In Random Network Coding networks, however, a large amount of encoded packets are generated and transmitted to the receiver by intermediate nodes. This characteristic of Random Network Coding makes the receiver's chances of collecting packets that *do* adhere to the LT specifications significantly higher.

We compare the performance of these two methods to determine the difference in their decoding time.

B. Experimental results

Fig. 6 plots the decoding times of both methods in terms of network size, N . The solid lines represent the decoding performances with the transmission of the minimum amount of source data $S = 4$, and the broken lines with the transmission of larger amount of source data, $S = 8$.

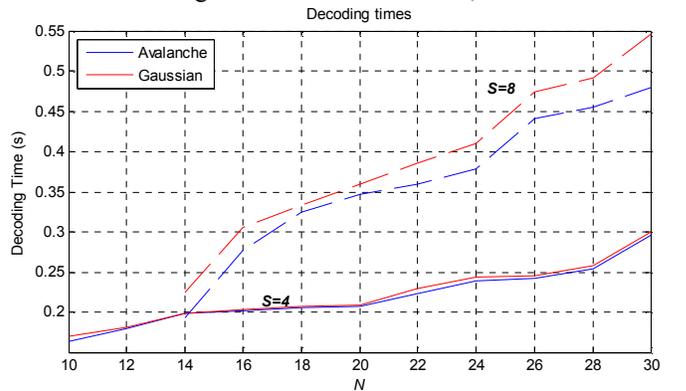


Figure 6: Decoding times in networks of size N

In networks where small amounts of source data are transmitted, we can see that the decoding time of both models are nearly the same. With the transmission of more source packets over the network, the avalanche decoding method outperforms that of traditional Gaussian elimination.

Fig. 7 plots the decoding success of both methods in terms of network size, N . The solid lines represent the decoding performances with the transmission of the minimum amount of source data $S = 4$, and the broken lines with the transmission of larger amount of source data, $S = 8$.

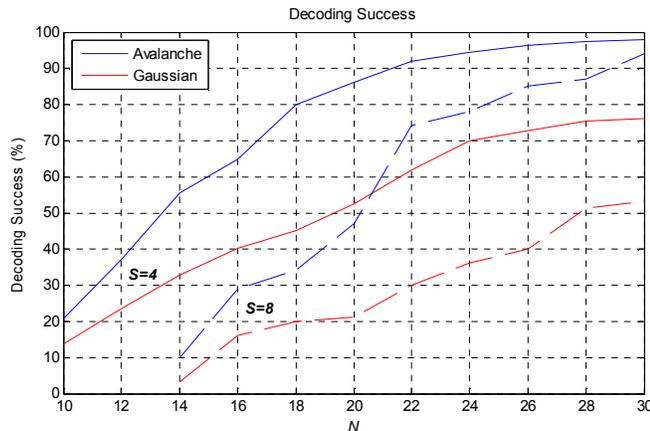


Figure 7: Decoding success in networks of size N

In smaller networks, it can be seen that the decoding success of avalanche decoding is higher, and increases dramatically as the network gets larger. As the network size and the amount of transmitted source data increases, the avalanche decoding success is higher than that of the Gaussian elimination method.

C. Experimental conclusions

From the results obtained, the following conclusions can be made.

Decoding time: For Gaussian elimination, the receiver node can construct a generator matrix only after sufficient encoded packets are collected. As more source data is transmitted the receiver node must wait longer to obtain sufficient encoded packets for decoding and the generator matrix, G , becomes larger and more complex. These factors cause the complexity of decoding to increase exponentially and therefore the decoding time also increases.

In the case of avalanche decoding, the receiver can start decoding as soon as an encoded packet of degree $d = 1$ is collected, with a minimal effect on decoding complexity and time. When the network becomes larger, the connectivity of the network increases, which means that more nodes are connected to one another by overlapping broadcast areas (see Fig. 5). This is advantageous for this decoding scheme, because a large range of encoded packets reaches the receiver node in a shorter time.

Decoding success: As the amount of transmitted source data increases, the construction of an invertible generator matrix becomes more difficult. This is due to the fact that the receiver does not always collect sufficient encoded packets with

linearly independent encoding vectors. This leads to a G matrix that cannot be inverted, and data that cannot be decoded.

In the case with avalanche decoding, linearly independent packets are not necessary for successful decoding, which improves the decoding chances.

V. CONCLUSION

In this paper we implemented avalanche decoding, a decoding method based on the low complexity decoding scheme of LT codes, in a Random Network Coding network. This decoding method is based on that of LT codes, without the complication of nodes encoding according to a specific degree distribution. This proposed method can be implemented in a Random Network Coding network where intermediate nodes encode packets randomly and independently.

Through the implementation of random linear fountain codes and a data carousel, we were able to implement avalanche decoding without the need for a specific degree distribution, to obtain favourable results.

When compared to traditional decoding methods implementing Gaussian elimination, a decrease can be seen in decoding time as well as a significant increase in decoding success.

VI. REFERENCES

- [1] R. Ahlswede, N. Cai, R. Li, and R. W. Yeung, "Network Information flow," *IEEE trans. on Information Theory*, vol. 46, pp. 1204-1216, 2000.
- [2] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *IEEE Int. Symp. Information Theory*, Yokohama, July 2003, p. 442.
- [3] T. Ho, "Networking from a Network Coding perspective," PhD Thesis, Massachusetts Institute of Technology, Dept of EECS, May 2004.
- [4] S. Lin and D. J. Costello, *Error control coding: Fundamentals and applications*. Englewood Cliffs, N.J: Prentice-Hall, 1983.
- [5] D. Silva, F. R. Kschischang, and R. Koetter, "A Rank-Metric approach to error control in random network coding," in *Proc. Information Theory for Wireless Networks, 2007 IEEE Information Theory Workshop*, Solstrand, Norway, July 2007, p. 1-5
- [6] A. H. Dekker and B. D. Colbert, "Network Robustness and Graph Theory," in *27th Australasian Computer Science Conference*, in *Conferences in Research and Practice in Information Technology*, Vol. 26, V. Estivill-Castro, Ed. 2004
- [7] M. Luby, "LT codes", *Foundations of Computer Science*, 2002. *Proceedings. The 43rd Annual IEEE Symposium on*, 2002, 271-280
- [8] D. MacKay, "Fountain codes", *Communications, IEE Proceedings-*, *Communications, IEE Proceedings-*, 2005, 152, 1062-1068
- [9] D. J.C. MacKay, "Information Theory, Inference, and Learning Algorithms", Cambridge University Press, 2003.
- [10] M. Champel, K. Huguenin, A. Kermarrec, N. Le Scouarnec, "LT Network Codes", *Institut National de Recherche en Informatique et en Automatique (INRIA)*, Novembre 2009
- [11] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Trans. on Information Theory*, vol 49, p. 371, Feb 2003.
- [12] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Allerton Conference on Communication, Control and Computing*, Monticello, IL, October 2003.

Suné von Solms is born in Port Elizabeth on 26 January 1985. She completed her Bachelor of Engineering degree in Computer and Electronic Engineering in 2007 and her Master's degree in Computer Engineering in 2009 at the North West University, Potchefstroom campus. She is currently a Ph.D student in Computer Engineering for the Telkom COE at the North West University.