Application Layer Coding for Delay and Feedback-Constrainted Scenarios

Thesis submitted to the

Indian Institute of Technology Bhubaneswar

For End Semester Evaluation

of

Bachelor of Technology Project

by

Vatsalya Chaubey

Under the guidance of

Dr Siddhartha S Borkotoky

SCHOOL OF ELECTRICAL SCIENCES

INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

May 2021

CERTIFICATE

This is to certify that the thesis entitled **Application Layer Coding for Delay and Feedback-Constrainted Scenarios**, submitted by **Vatsalya Chaubey (17EC01044)** to Indian Institute of Technology Bhubaneswar, is a record of bonafide research work under my supervision and the report is submitted for end semester evaluation of the B.Tech project.

Date :

Dr Siddhartha S Borkotoky
Assistant Professor
School of Electrical Sciences
Indian Institute of Technology Bhubaneswar
Bhubaneswar, India

DECLARATION

I certify that

a. the work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.

b. the work has not been submitted to any other institute for any degree or diploma.

c. I have followed the guidelines provided by the institute in writing the thesis.

d. I have conformed to the norms and guidelines given in the ethical code of conduct of the institute.

e. whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f. whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Vatsalya Chaubey

Acknowledgments

This work was done as part of my final year B.Tech project. I would like to thank my supervisor Dr Siddhartha S Borkotoky for his invaluable guidance and technical help. I would also like to thank him for the level of freedom provided by him to help me do various experiments which often very not fruitful. This work was during the pandemic where everyone was stuck at home and without any direct contact. I would like to thank him for his availability over video calls and his patience in guiding me through this tough times. I would also like to thank my parents for their support during this tough period.

Vatsalya Chaubey

Abstract

Application layer in a communication network is an abstraction layer that provides a set of shared protocols and interfaces between various hosts for information transfer. It is the topmost layer in various communication models like TCP/IP and OSI and masks the underlying mechanisms and allow for communication between various applications in different hosts. Application-layer coding has been shown to be an effective means to increase network reliability via transmission of redundancy to compensate for random packet losses in the network. In this work, we develop a low-complexity application-layer coding scheme suitable for delay-constrained communications with intermittently available receiver feedback. Such a scheme could be widely used for control operations in wireless sensor networks where a number of sensors transmit data to a common gateway for analysis and decision making purposes.

Contents

List of Figures

List of Algorithms

List of Symbols

$\binom{k}{l}$	Binomial coefficient
s_i	Information symbol number i
$\mathbf{p}_i$	Packet number i
b	Maximum number of symbols a packet can carry
$\mathbf{q}_i[k]$	k-th symbol in payload $\mathbf{p}_i$, $1 \leq k \leq b$
u	Sequence number of the oldest undelivered packet
β	Number of unexpired packets undelivered
$c_d(\mathbf{v})$	A coded symbol produced coding XORing d random symbols from the set $\mathbf{v}$
$c_D(\mathbf{v})$	A coded symbol produced coding XORing D random symbols from the set $\mathbf{v}$, where D is a uniform random variable
$\mathbf{v}[k]$	k-th entry of vector $\mathbf{v}$
$\mathbf{v}\backslash\{x\}$	Set of all entries of $\mathbf{v}$ except x
s_{relay}	Oldest information symbol present in the relay memory

Chapter 1

Introduction

This chapter provides a brief summary of the Application layer which forms an integral part of the TCP/IP and OSI networks models. This chapter also discusses the need of application layer coding and basic description of some schemes already in use. In the latter sections, we also discuss our use case along with a description of a previous works already done in this area.

1.1 Application Layer

The Transmission Control Protocol (TCP) and the Internet Protocol (IP) form the backbone of today's internet and other data communication methods. They are collection of protocols that define how the data is to be packetized, framed, addressed, transmitted, routed and received. All of these various functions are abstracted through various layers each having a specific function. The TCP/IP protocol suite has five layers and application layer is one of them.

Application Layer is present at the top of the TCP/IP and OSI models. It is through this layer that a user interacts with the system. It also provides services and interfaces which ensures that an application can communicate with any other application on the

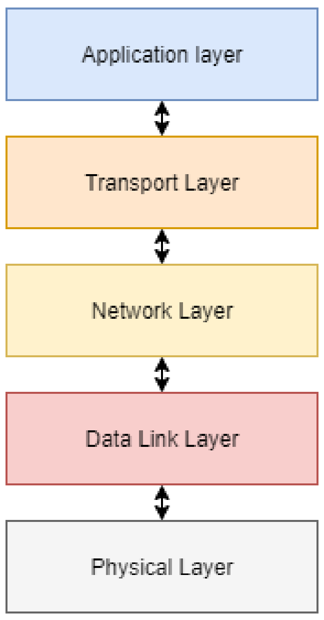


Figure 1.1: Various layers of the TCP/IP protocol. The Application Layer sits at the top

same network. Though the name states “Application Layer”, it should not be thought of a software application rather it is a component of the application which allows it to communicate to other applications on the network through the various lower layers of TCP/IP protocol. It serves to abstract the messy inner steps involved with the lower layers during data communication or transfer. Figure 1.1 shows the various layers involved in the TCP/IP protocol. The application layer protocols provides various functions:

- **Identification of Communicating Nodes:** The application layer decides the availability of a node for an application with data to transmit.
- **Resource availability:** It also decides whether the amount of network resources required for any communication are available.
- **Synchronization:** It make sures that applications communicating with each other are in sync.

1.2 Problem Formulation and Motivation

Many wireless networks operate in certain in frequency band where they are subjected to various duty cycle constraints. Moreover, the current trend of building connected devices or systems has made it necessary for using many sensor nodes transmitting data to a receiver for further processing and decision making. So these duty cycle constraints limit the ability of the receiver to transmit feedback about the reception of packets back to the sensors. In a similar fashion, the sensor nodes themselves can only make a limited number of retransmissions for erasure correction. Now, the task of designing an efficient application layer coding scheme which can used in the above scenario becomes challenging, but, if the use cases are delay sensitive then it becomes further complicated. In such a scenario, one data packet generated at the source is of value to receiver only for a certain time frame and after passage of that threshold it becomes useless. Examples of such networks can be found in sensor networks in industries which monitor and control various processes in a plant and where most decisions are made in real-time.

In this thesis, the focus is on a duty cycle and feedback constrained delay sensitive multipoint to point communication. The work is based on creating modifications to an existing coding technique [1]. In our use case, the data generated is valid for a fixed time interval, after which its importance cease to exist. All transmission made by the sensor node can be a uncoded packet (pure information packet or message) or a coded message (a data packet formed by linear combination of two or more messages). Since most sensor are low-powered devices we aim to develop a coding scheme which is computationally cheap and avoids coding packets unless absolutely necessary. The feedback structure of the system is also designed in a way such that maximum information about the receiver state can be transmitted back to the receiver. The design also makes sure that even if feedback is missing coding is done effectively to ensure better packet reception. We limit our design to GF(2) where coding is done by simply XORing two symbols to avoid complex coding schemes in higher order fields.

1.3 Literature Survey

LT codes [2] and Raptor Codes [3] have been used widely due to their strong erasure correction. They operate over a large set of symbols which translates to higher decoding delays. This delay makes them useless for delay-constraints applications. [4] attempts a coding scheme with smaller blocks of information symbols. They investigate the effectiveness of their approach for networks where a number of senders transmits to a single receiver over an erasure channel but with perfect feedback. They account for the delay constraints but fail to acknowledge the duty cycle constraint in our case. Later, [5] introduced a new network adaptive coding scheme for low latency communication. They aimed at reducing both random and burst erasures by coding in a higher order field, namely GF(256). Since, their work was focused on streaming applications they could afford coding in such high order fields, but in our resource constrained use case coding in such higher order fields would drastically increase the computational complexity, which would make the sensors power hungry and energy inefficient. Some of the above mentioned problems were addressed in [6], where they design a coding scheme for LoRaWAN which provides better packet reception and reliability against burst and random errors. The system is designed in complete absence of feedback, which proves to be a vital resource if used efficiently.

The authors of [1] introduce two coding schemes—*Windowed Coding and Selective Coding* for our use cases. Both of the schemes make use of available intermittent feedback along with a unique packet structure and coding scheme. They intelligently decide when to code and how many symbols are to be coded. They also provide steps to perform when feedback is unavailable. This current work is heavily based on the *Windowed Coding* scheme making modifications to it to remove the implicit shortcomings of that approach. The *Windowed Coding* scheme is described in detail in Chapter 2.

1.4 Channel Models

A number of mathematical models are currently available which emulate how a communication channel will perform under certain conditions. In this particular section we describe the various channel models that have been used throughout this study namely the Bernouli and Gilbert-Elliot channel models.

1.4.1 Bernouli Channel

A Bernouli channel is the simplest of all channel models where each packet transmission is independent of the other. This is a channel erasure model where a packet transmitted can either be successfully received or can be lost while transmission but it cannot get corrupted (It is assumed that certain channel coding protocols have been used which ensure error free delivery or error correction capabilities if errors do happen). This type of channel can be characterized by the probability of correct reception or success, which indicates that if a packet is transmitted from the sender with what probability it will be successfully received by the receiver. Transmission through the channel is usually modelled by generating a random number in $[0, 1]$ and if this number is less than the channel success probability the packet transmission is successful.

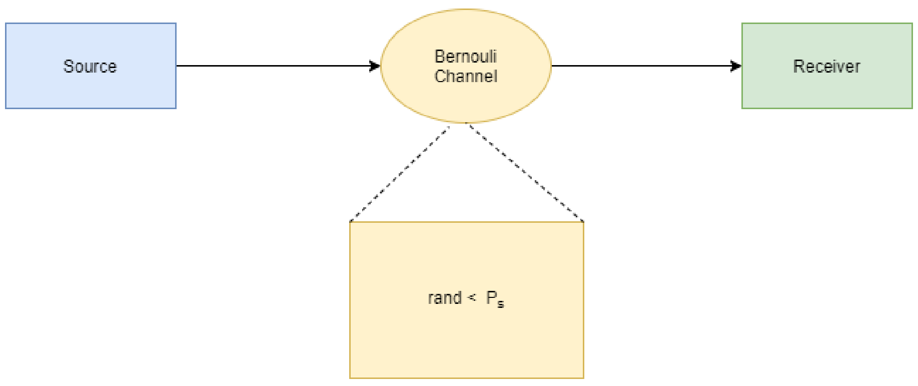


Figure 1.2: Successful transmission through a Bernouli channel happens when an uniformly random number in $[0, 1]$ is less than the packet success probability.

1.4.2 Gilbert-Elliot Channel

A Gilbert-Elliot Channel model is a two state Markov process which is used to model correlated erasures that might happen in a wireless channel. Such erasures may happen when there is a temporary obstacle between the source and receiver leading to some successive transmissions being lost. The Markov process has two states - good and bad. A packet transmitted during the good state is received correctly and is lost if transmitted during the bad state. The states may change from one to another based on the state transition probabilities - p_{bg}, probability of bad state to go into good state; and p_{gb}, probability of good state to go in bad state.

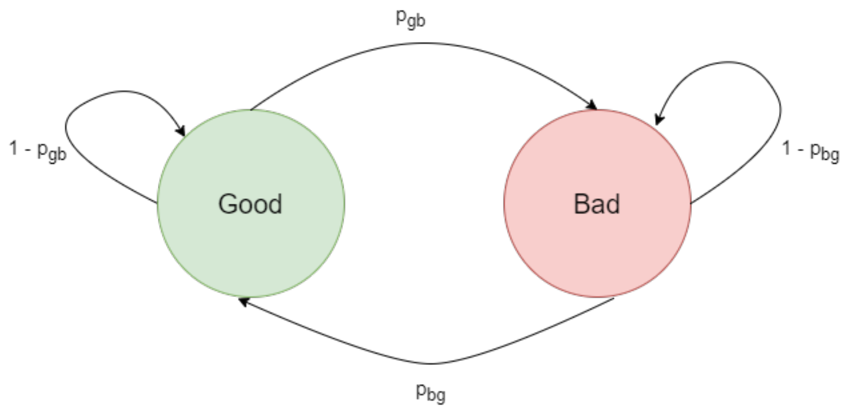


Figure 1.3: State transitions in a Gilbert-Elliot channel

$$P_{ss} = \begin{bmatrix} p_{gg} & p_{gb} \\ p_{bg} & p_{bb} \end{bmatrix} = \begin{bmatrix} 1 - p_{gb} & p_{gb} \\ p_{bg} & 1 - p_{bg} \end{bmatrix} \tag{1.1}$$

1.5 Summary

In this chapter, we discussed the basics of application layer, the previous works and basic channel models which we would be using throughout our work. In the succeeding chapter we will describe the *Windowed Coding* scheme from [1] which forms the backbone of our work.

Chapter 2

Windowed Coding

In this chapter, we discuss the *Windowed Coding* scheme introduced in [1] on which this work is based upon.

2.1 General Packet and Network Structure

The sender generates a information symbol s_i at the current time instant, i. Let us assume δ_{max} to be the maximum delay tolerance. So, all the symbols generated before the time instant $i - \delta_{max}$ are of no use to the receiver. Multiple symbols are transmitted together as a single packet, instead of transmitting each symbol as seperate packet as done in [4, 5]. This is done to utilize the packet structure of LoRa, where the packet duration remains same data size of 1 byte through 4 byte [7], allowing to maintain the energy and duty cycle restrictions.

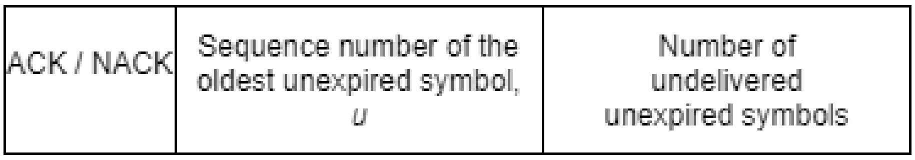
Multiple information symbols can be coded together and transmitted to increase redundancy. A *coded* message is formed by XORing d information symbols as in Eq. 2.1. d is known as the coding degree. A coded packet can be decoded at the receiver if $d - 1$ packets are already received as in Eq. 2.2. So, a coded packet can be used to decode at max one symbol, but the advantage of coding is that the retrieved symbol can be any one of the d symbols.

$$c_d = s_1 \oplus s_2 .... \oplus s_d \tag{2.1}$$

$$s_d = c_d \oplus s_1 .... \oplus s_{d-1} \tag{2.2}$$

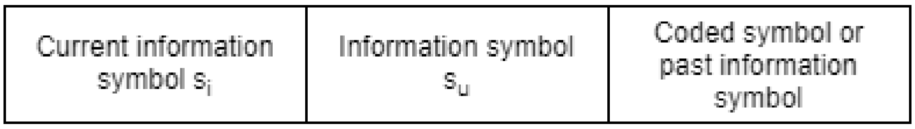
A symbol can be successfully received at the receiver if it is delivered in its raw state or if it can be decoded from a coded symbol.

The size of a packet, b is the number of symbols (coded or uncoded) that can be incorporated in a packet. If the maximum size of packet is l_p bits, decided by the duty cycle constraints, and each symbol is l_s bits then the size $b = l_p/l_s$. The first symbol in the packet is always the current symbol i.e the symbol at time i. The symbols following it may be coded or uncoded.



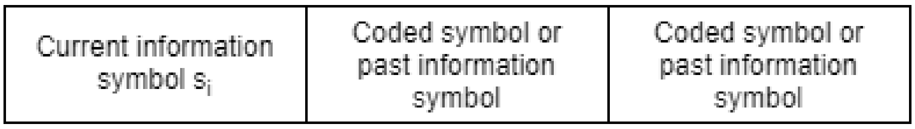
ACK / NACK	Sequence number of the oldest unexpired symbol, *u*	Number of undelivered unexpired symbols

Figure 2.1: Structure of the feedback packet



Current information symbol s_i	Information symbol s_u	Coded symbol or past information symbol

Figure 2.2: Packet structure of size 3 when feedback has been received for the previous packet



Current information symbol s_i	Coded symbol or past information symbol	Coded symbol or past information symbol

Figure 2.3: Packet structure of size 3 when feedback has not been received for the previous packet

The sender transmit a packet using one of the structures as shown in Fig. 2.2 and Fig. 2.3. The structure of the packet is decided by the coding algorithm described in the next section. The transmissions take place over a erasure channel. The paper [1] describes its results for Bernouli, Gilbert-Elliot and LoRa channels. Every sender may receive a feedback from the receiver with a probability of $p_{feedback}$. The structure of feedback is shown in Fig. 2.1 where the first bit ACK/NACK provides acknowledgement whether the current symbol, s_i was delivered or not. It is followed by two data symbols which contain the sequence number, u of the oldest unexpired symbol and the total number of undelivered unexpired symbols, β. If all the packets up till the current packet has been received, the feedback will contain $u = i + 1$ and $\beta = 0$

2.2 Coding Scheme

The sender at time instant i produces an information symbol s_i. To form the packet $\mathbf{p}_i$, the first entry would be the current packet s_i. Now it needs to choose how to prepare packet $\mathbf{p}_i$. This decision is made based on whether feedback was received for the previous packet $\mathbf{p}_{i-1}$.

1. *Feedback for the packet* $\mathbf{p}_{i-1}$ *was received*

   The feedback will inform the sender the oldest undelivered symbol, u and the number of undelivered unexpired symbols, β in the range $(i - u, i)$. Then the symbols of the interest to the receiver from a set $\mathbf{w}_i = \{s_u, s_{u+1}, ..., s_{i-1}\}$. The remaining $b - 1$ spaces are filled based on the below conditions:

   (a) $u = i$: This would mean that all the symbols upto the current symbol have been received by the receiver and nothing extra needs to be sent.

   (b) $u < i, i - u \leq b - 1, \beta \geq 1$: This would mean that all the $i - u$ symbols of interest can fit inside the remaining $b - 1$ space of the payload $\mathbf{p}_i$. So all the symbols in $\mathbf{w}_i$ are appended to the payload $\mathbf{p}_i$.

   (c) $u < i, i - u > b - 1, 1 < \beta = i - u$: This would mean that all the $i - u$

symbols are missing and there is not enough space in the payload $\mathbf{p}_i$ to include everything. Since all the symbols in $\mathbf{w}_i$ are missing the first $b-1$ packets are appended to the payload because they are the symbols that will lose their relevance first as compared to others.

(d) $u < i, i-u > b-1, 1 < \beta < i-u$: Here, not all symbols in $\mathbf{w}_i$ are missing, but what symbols are missing is unknown, except for s_u. Only the number of missing symbols is known. It is only in this case that coding is performed. So, first the known missing symbol s_u is appended to the packet $\mathbf{p}_i$. Now we have $b-2$ free spaces and $\beta-1$ missing symbols. So the remaining $b-2$ symbols in the payload are coded symbols generated by random linear combination of d elements from the set $\mathbf{w}_i \backslash \{s_u\}$. The degree is chosen as

$$d = \underset{d'}{\operatorname{argmax}} \frac{\binom{\beta-1}{1} \cdot \binom{i-u-\beta}{d'-1}}{\binom{i-u}{d'}} \tag{2.3}$$

The above equation aims to choose a degree d which maximizes the probability of decoding of one packet at the receiver. Eq 2.3 is a crude estimate of the decoding success probability. Recall, that a symbol can be decoded if $d-1$ out of the d symbols are already received. So in Eq. 2.3, the number of ways 1 symbol from the $\beta-1$ missing symbols and the other $d'-1$ symbols from $i-u-\beta$ already received symbols can be chosen is calculated. To find the probability this number is divided by the total number of ways to choose d' symbols from $i-u$ symbols.

2. *Feedback for the packet* $\mathbf{p}_{i-1}$ *was not received*

If the feedback for the previous packet is not received the oldest undelivered symbol at the moment is unknown. So a set is defined as $\mathbf{b}_i = \{s_{i-1}, s_{i-2}, ..., s_{i-z}\}$ where $z = min(i-u_l, i-u_{max})$, u_l is the oldest undelivered sequence in the most recent feedback and $u_{max} = i - \delta_{max}$, the oldest unexpired information symbol at the current instant, i. This scenario can also be decomposed into similar cases:

(a) $z \leq b-1$: The entire vector $\mathbf{b}_i$ can be included in the payload $\mathbf{p}_i$ after the current symbol s_i.

(b) $z > b - 1$: This is similar to the case (d) in the feedback case. Here, again the missing symbols are unknown and not all symbols of interest can be incorporated in the payload. Coding of symbols is again employed here but since no other information about the state of the receiver is known, it is assumed that all symbols in the $\mathbf{b}_i$ are equally likely to be missing. So, coded symbols are generated by XORing D symbols from $\mathbf{b}_i$, where D is a uniform random variable between $[1, z]$.

2.3 Problems with the scheme

Though this scheme perform really well in practice, there are a few important shortcomings which need to be considered. The packet reception rate is a order of magnitude better when compared against two baselines—repetition redundancy and blind coding scheme. The scheme was evaluated on three erasure channels (Bernouli, Gilbert-Elliot, LoRa) against these baselines. More detailed results can be found in the paper [1]. Here, we focus more on three problems of this scheme and in the next chapter we will describe the steps taken to mitigate them.

2.3.1 Degree Selection when coding

The degree selection equation (Eq. 2.3) is computationally very intensive. As the delay tolerance δ_{max} will increase the value of d' will increase thus the number of times the $\binom{n}{k}$ operation needs to be done to calculate a single value of d will also increase. The complexity of one $\binom{n}{k}$ operation is approximately $O(min(n^k, n^{n-k})) \simeq O(n^k)$ so the complexity of selecting the degree once is approximately

$$O\left(\sum_{d'=1}^{d'=i-u-\beta+1} (i-u-\beta)^{d'-1} * (i-u)^{d'}\right)$$

when we only consider the upper bound.

This is unfathomably complex for a low power sensor node to calculate for every transmission. If such a computationally intensive equation is used to calculate the degree most of the battery of the sensor would be consumed in calculating this. One way to avoid this problem which the authors of [1] suggest is using a table of values in memory. This approach is feasible when the delay tolerance δ_{max} is low because the space complexity of this approach is $O\left(\delta_{max}^2\right)$. For the applications for which this method was first created this condition is usually met. But, in this work we want to do away of this requirement of low δ_{max} and make a more general coding scheme.

One more issue with the Eq. 2.3 is that it is only a crude estimate of successful decode probability and is only valid if the packet contains only one coded packet. If a packet contains more than one coded packet this estimate would further drift away from the actual way. To circumvent this, a joint optimization of degree should be done which was not done in [1] to avoid further complexity.

2.3.2 Feedback

The structure of the feedback packet is shown in the Fig. 2.1. Let us suppose our $\delta_{max} = 16$, which is a fairly common value in an industrial setting and the number of symbols we need to transmit is $10,000$. Then the number of bits required by our feedback would be $1 + 14 + 14 = 29$. One bit for acknowledgement, 14 bits each to represent the oldest undelivered sequence number and number of undelivered symbols. This approach is not very efficient. Even using 29 bits or more practically 32 bits, we could only know about the state of one symbol with surety i.e. s_u and just a cumulative estimate of the state of other symbols. In a way, the feedback is under utilized and more useful information can be packed in it.

2.3.3 No-feedback Coding Degree

The coding degree chosen when there is no feedback is D, which is a uniform random variable between $[1, z]$. The idea behind this is the assumption that all symbols are equally likely to be missing, which is actually not the case. Moreover, in our experiments, choosing this degree randomly underperformed when compared to a fixed degree. In our case, we could get the best performance when this degree was fixed at 2. This calls to question the theoretical reasoning behind this approach. Further study is needed to get to the actual reason behind this and come up with better methods to choose this degree.

2.4 Summary

In this chapter we described the windowed coding scheme and the various issues associated with its design and implementation. In the next chapter we will describe some methods through which we can deal with some problems and thus make the algorithm better.

Chapter 3

Improved Windowed Coding

In this chapter, we discuss the modifications made to the windowed scheme introduced in chapter 2 to improve upon the shortcomings in section 2.3. In the proceeding sections we take up each issue and describe the modifications and their effects. In some cases, since the work is in progress, we describe our thought process and preliminary results.

3.1 Experimental Setup

As an initial step, we describe our experimental setup. The windowed coding scheme was simulated in both Python and MATLAB. We are only working on the windowed coding scheme of the paper [1] and will later evaluate the *selective coding* scheme. The feedback implementation was modified to augment the information content along with the improvements to the degree selection already made. A notable difference as stated by the authors of [1] is that there exists a buffer at the receiver in their implementation which stores some recent coded symbols even if no information symbol can be decoded from them. This helps in decoding symbols later when some extra information symbols have been received. Currently, our implementation does not have a buffer at the receiver. But, in our experiments we find the role of the buffer to be minimal but further evaluation needs to be done. One initial roadblock which we faced was similar code in Python and MATLAB diverged on their results. Though, later we decided to stick to MATLAB which provided better results.

3.2 Improvements

In this section we describe the modifications made to solve some of the issues in section 2.3. We first describe improved degree selection process followed by improved feedback structure and no feedback degree selection.

3.2.1 Improved Coding Degree Selection

As stated in section 2.3.1 the process of choosing a suitable coding degree is computationally very demanding and the energy and space requirements are too high to be met by any small sensor node. Additionally, the time consumed in calculating the degree is also very high which may prove to be disastrous in high speed applications.

The optimal degree is chosen by:

$$f(x, y) = \underset{d}{\operatorname{argmax}} \left(y \frac{\binom{x-y}{d-1}}{\binom{x}{d}} \right) \tag{3.1}$$

where $x = i - u - 1$, $y = \beta - 1$ and f is the optimal degree.

Now, we need to find the maxima of the $f(x, y)$ function which can give us the optimum degree. So instead of searching over the entire domain space of d we use calculus to find the maxima. This optimization though a little involved but the final results are very simple and worth it. We can expand the function $f(x, y)$ as

$$f(x,y) = \operatorname*{argmax}_d \left(y \frac{\frac{(x-y)(x-y-1)...(x-y-d+2)}{(d-1)(d-2)...1}}{\frac{x(x-1)(x-2)...(x-d+1)}{d(d-1)...1}} \right)$$

$$f(x,y) = \operatorname*{argmax}_d \left(d \cdot y \frac{(x-y)(x-y-1)...(x-y-d+2)}{x(x-1)(x-2)...(x-d+1)} \right)$$

Let us assume we need to maximize $g(x,y)$ with

$$g(d) = d \cdot y \frac{(x-y)(x-y-1)...(x-y-d+2)}{x(x-1)(x-2)...(x-d+1)} \tag{3.2}$$

This is only a function of d because x and y are constant at a given time. Take log of this for finding the derivative. In that case:

$$h(d) = \ln d + \ln y + \sum_{k=0}^{d-2} \ln(x-y-k) - \sum_{k=0}^{d-1} \ln(x-k) \tag{3.3}$$

where $h(d) = \ln(g(d))$. We know the values of x, y and d are integers. So the above function $ln(g(d))$ is discrete and the derivative cannot be calculated in a continuous fashion. We know derivative in discrete domain is

$$y[n] = x[n] - x[n-1] \tag{3.4}$$

Use 3.4 to find derivate of 3.3. This will give the derivate as $h'(d)$ as:

$$h'(d) = \ln(\frac{d}{d-1}) + \ln(x-y-d+2) - \ln(x-d+1) \tag{3.5}$$

For calculating the maxima we need to find the value of d where $h'(d)$ is zero.

$$\ln(\frac{d}{d-1}) + \ln(x-y-d+2) - \ln(x-d+1) = 0$$

$$\frac{d}{d-1}.\frac{(x-y-d+2)}{(x-d+1)} = 1$$

$$dx - dy - d^2 + 2d = dx - d^2 - x + 2d - 1$$

$$-dy = -x - 1$$

$$d = \frac{x+1}{y} \tag{3.6}$$

Hence, the optimal degree

$$d = \frac{i-u}{\beta - 1} \tag{3.7}$$

where, i is the current sequence number, u is the oldest undelivered packet and β is the number of undelivered packets. One more caveat of implementation is that the maximum value of degree cannot be greater than the window size - number of undelivered and degree should be integer. To incorporate both the conditions we transform 3.7 as:

$$d = \min(i - u - \beta, \lfloor\frac{i-u}{\beta-1}\rfloor) \tag{3.8}$$

It is clear now that Eq 3.8 provides a better way to calculate the degree. It removes all the complex calculations and now both the space and time complexity of calculating the degree once is $O(1)$. Fig. 3.1 shows the result of the simulation when we compare the original degree selection method (Eq. 2.3) with our optimized degree selection

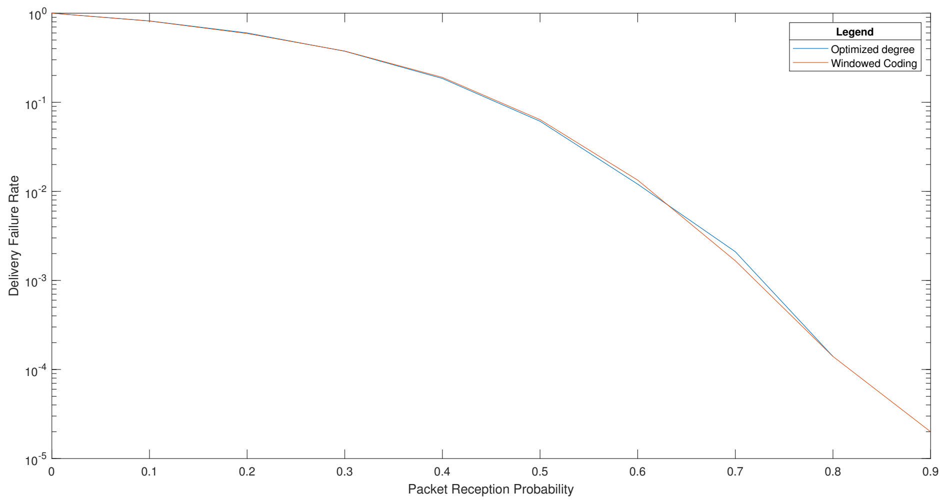


Figure 3.1: The results obtained when simulating the original windowed coding scheme (red) and the same coding scheme with better degree optimization (blue). It is clearly visible that the curves are very similar.

(Eq. 3.8). Both the approaches give similar results as there is no theoretical difference between the two but our optimized approach is computationally very efficient.

3.2.2 Improvised Feedback

In an earlier section 2.3.2, we stated that the feedback is not fully utilized. Even if we 32 bit feedback we could only know the state of one symbol with certainty. Suppose, we know which other symbols are missing we could directly send them in the next transmission. Each symbol whose state we know will alleviate the need of coding for it. This will greatly reduce the energy consumption at both the receiver's and transmitter's end, which is one of the key requirements of our use case.

We keep the basic structure of the feedback packet (see Fig. 2.1) intact. In our approach, we only modify the "number of unexpired undelivered symbols" section of the feedback, because it is section which is not fully utilized. We use the 15 bits for a 32 bit feedback, assigned to that section to pass on information about the other symbols back to the sender. The first 16 bit section of the feedback provides the sequence number of the oldest undelivered unexpired symbol, u, then we use the next 15 bit segment to denote whether symbols $s_{u+1}, s_{u+2}, ..., s_{u+15}$ have been received or not. If any of that symbol is missing that bit would be set. This can help us know about the exact state of the symbols from $[s_u, s_{u+15}]$ so depending on the packet size b we can send the missing symbols directly without coding. The major advantage of this approach is that if the delay tolerance δ_{max} is less or equal to 16 we do not need to code for any symbol in the payload $\mathbf{p}_i$, if we have received feedback for the packet $\mathbf{p}_{i-1}$. Coding still needs to be done if we don't receive feedback for the previous packet. This scheme can still work for higher δ_{max} if we assign more bits to the feedback.

In our use case, the feedback bits also cannot increase arbitrary and they have to follow a limit to respect the duty cycle and energy constraints. So what can be done if δ_{max} is greater than the feedback bits assigned to state of other symbols? We propose an adaptive scheme in that case. Let us assume the number of bits assigned to the improvised feedback section be l_b. Now the set of relevant symbols at time instant i, $[s_u, s_i]$ can be broken into two disjoint sets $[s_u, s_{u+l_b}]$ and $[s_{u+l_b+1}, s_i]$ if $i - u > l_b$. The receiver knows the number of missing symbols in both of the sets. The first set contains older packets and they should be transmitted first to ensure that they don't expire. To do that we can use the improvised feedback as the state of all symbols in the first set can be incorporated into the feedback l_b bits. This works if majority of missing symbols are in the first set, but if the majority of symbols are missing in the second set we cannot send information about that set through our improvised method because then the older symbols would stand the risk to expire before delivery. So in this case we would shift back to the original *windowed coding* scheme transmitting the feedback as in Fig. 2.1. We can control the level of adaptability by defining a constant α as

$$\alpha = \frac{number\ of\ missing\ symbols\ in\ set\ [s_{u+l_b+1}, s_i]}{number\ of\ missing\ symbols\ in\ set\ [s_u, s_{u+l_b}]} \tag{3.9}$$

We can then set a threshold α_0. If $\alpha > \alpha_0$ we switch to the original *windowed coding* scheme's feedback structure, else we use the improvised feedback structure to pass information about symbols in the set $[s_u, s_{u+l_b}]$. The results of our experiments after making these modifications are shown in the next set of figures.

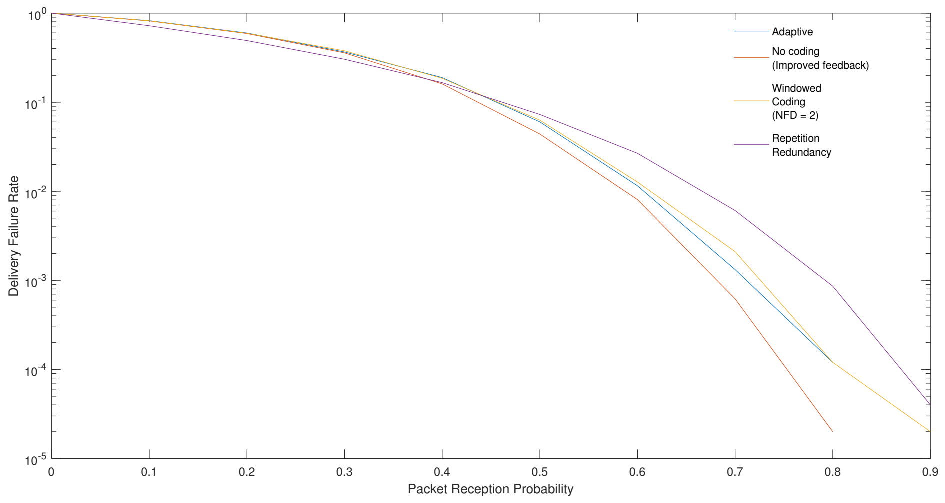


Figure 3.2: Comparison between baseline Repetition Redundancy (purple), Windowed Coding (yellow), Adaptive scheme (blue) and No coding or improved feedback (red). The simulation is done when delay tolerance $\delta_{max} = 4$ *and* $l_b = 4$. The switching to the original scheme in adaptive takes place instantaneously i.e. $\alpha_0 = 0$. It can be clearly seen that the no coding scheme performs better than everything and it does not employ any coding so is energy efficient too.

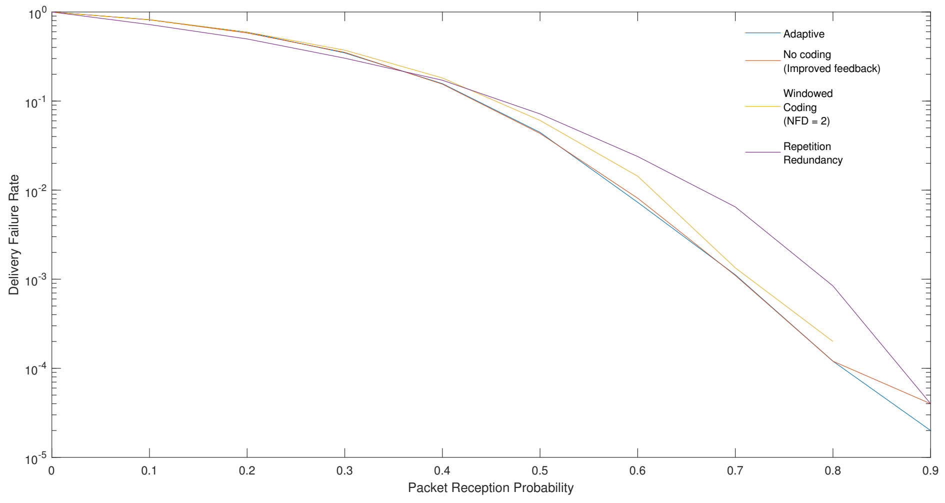


Figure 3.3: Similar simulation as in Fig. 3.2 but with $\delta_{max} = 16$ i.e $\delta_{max} > l_b$. The no coding performance deteriorates because now most of the missing symbols are outside the range in which it can provide information i.e. in set $[s_{u+l_b+1}, s_i]$.

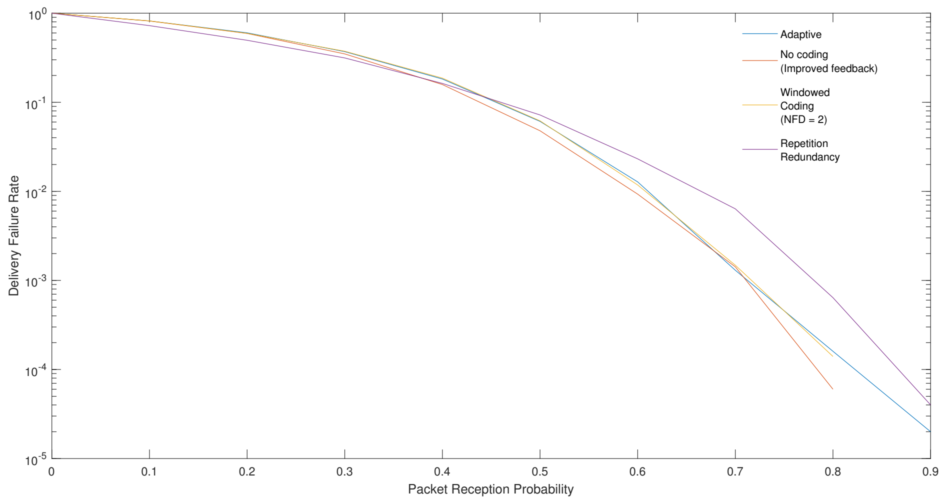


Figure 3.4: Similar simulation as in Fig. 3.3 but with $\alpha_0 = 1.5$. The no coding performance is similar to what was before because now also most of the missing symbols are outside the range in which it can provide information i.e. in set $[s_{u+l_b+1}, s_i]$. The adaptive scheme's performance becomes very similar to *windowed coding* because of the transmission take place in the windowed coding mode rather than the improvised feedback mode.

On subsequent experiments we found that their was no situation or a scenario with different values of δ_{max}, l_b or α_0 where the adaptive method would perform better than the *No Coding* method. In all the experiments we ran the adaptive method was bounded between *Windowed Coding* and *No Coding.* The idea that if more symbols are missing in the range $[s_{u+l_b+1}, s_i]$ we change to *windowed coding* which would allow us to send those symbols as part of the current packet, fails to look into the recency of the missing symbols. The *Window Coding* scheme gives equal priority to all missing symbols but we know that the symbols near s_u will expire earliest and thus should be given higher priority. This is what the *No Coding* method does and thus its performance is better or equal to the *Adaptive* scheme in majority of the cases.

3.2.3 No feedback degree selection

According to the algorithm of *windowed coding* if feedback for payload $\mathbf{p}_{i-1}$ is not received, then the current payload $\mathbf{p}_i$ will contain all coded symbols. The degree of coding is chosen randomly from a uniform distribution between $[1, z]$ where $z =$

$min(i - u_l, i - u_{max})$, u_l is the oldest undelivered sequence in the most recent feedback and $u_{max} = i - \delta_{max}$, the oldest unexpired information symbol at the current instant, i. The idea behind this that since no information about the state of symbols is known an assumption is made that all the symbols are equally likely to be missing so a uniform random variable is used to choose the coding degree. But on further thought it is clear that not all symbols have the same probability of missing. The symbols which are older have got more chances in the past to be delivered than recent symbols. In a way the probability of missing should increase for a more recent symbol, but this assumption is only valid if channel erasure is uniform. For Gilbert-Elliot channel where burst errors occur it may be possible that any set of symbols may be lost in a burst and thus no conclusion can be drawn about the missing probability of symbols.

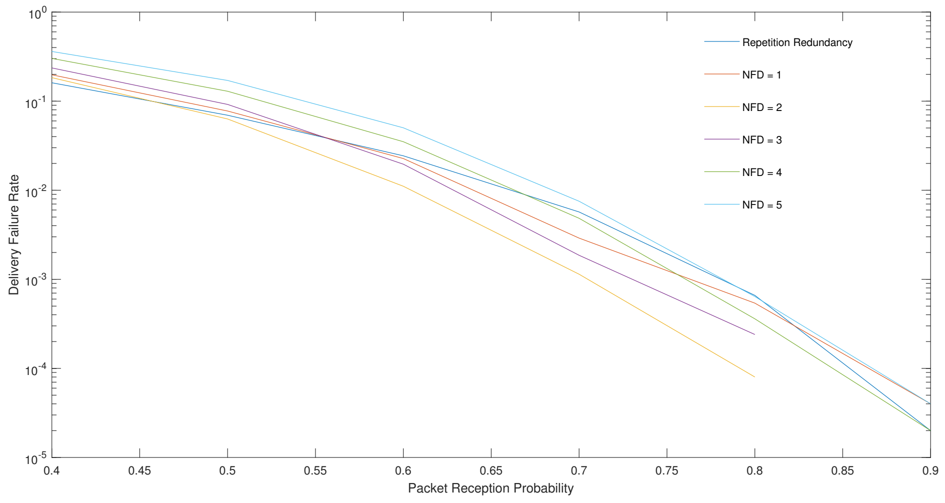


Figure 3.5: Delivery Failure Rate v/s Packet Reception Probability with different values of No Feedback Degree (NFD). *'NFD = 1'* does not transmit any redundancy as no coded packet is transmitted.

We conduct some experiments with various fixed degrees while coding when no feedback is present. The *Delivery Failure Rate* for *Windowed Coding* is quite erratic and sometimes performs even worse than Repetition Redundancy. The best performance is seen when the no feedback coding degree is 2. There is no clear theoretical understanding why this happens and what should be the optimal no feedback degree for different scenarios. Intuitively, the best performance of $NFD = 2$ can be explained by the decoding criteria. We know that for successful decoding $d-1$ out of the d combined symbols must already be received. So when the degree is 2 if anyone among the two is already present decoding is successful but for degree 3 we would any two of them to be already received and so on for higher degrees. It becomes improbable that atleast $n-1$ symbols are already received for n-degree coded symbol as n increases. So most of the transmissions with higher coding degree would not result into a successful decoding of the message and thus the delivery failure rate is higher for higher degree.

3.3 Summary

In this chapter we introduced several modifications and improvements to the *Windowed Coding* scheme which improve the *Delivery Failure Rate* as well as computationally cheaper. We introduced the *No coding* scheme (we will refer to it as ***Uncoded Scheme*** just to get make notation less ambiguous) which due to its feedback structure is more reliable, faster and energy efficient as the number of coded packets are reduced. Secondly, we also improved the nascent *Windowed Coding* scheme by reducing the complexity of degree selection. Though all these modifications are for single point to point communication, we will look into the effect of adding a relay in the subsequent chapters.

Chapter 4

Relay

Our analysis in preceding chapters has been limited to the study of single source to gateway point to point communication. In this chapter we introduce a relay node between the source and the gateway and we evaluate the performance gain and constraints in such a setting.

4.1 Basic Relay Operations

A relay node in a network is usually tasked with increasing the redundancy and routing the packets generated at the source, when direct transmissions between source and receiver may be hampered or impossible. The sender sends the message packet both to the receiver directly and simultaneously to the relay. Now, if the direct transmission between the source and receiver fails, the relay node may decide to retransmit that packet some time in the future. This leads to the development of an additional redundant path from the source to the receiver. The additional link established is more reliable as the probability of packet loss between source to relay, and, relay to receiver, is smaller than direct source to receiver transmission. This is because the relay is usually placed closer to the source than the receiver is from the source.

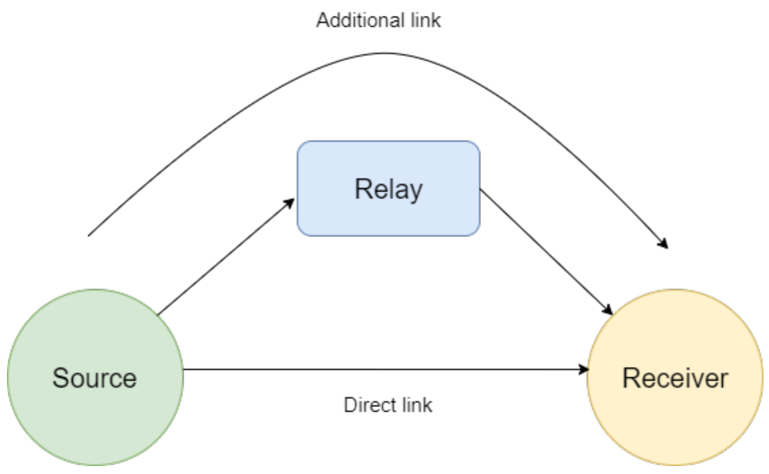


Figure 4.1: The additional link established due to relay node.

4.1.1 Relay Baseline

It is obvious that if we add a relay in any network the net performance in terms of packet delivery will increase due to the additional path and redundancy. So to evaluate our methods (which would describe later) we need to create a relay baseline against which we would compare our results.

First, we describe a few assumptions on the relay node which we introduce in the network:

- **Memory size:** The relay has fixed memory which we describe as the number of maximum source messages it can store. In our case we only keep the uncoded messages which have not yet expired for the relay i.e. in $[s_{i-\delta_{max}}, s_i]$
- **Transmitted packet size:** The relay will transmit a packet which will contain only 1 message (coded or uncoded). We fix this to 1 to make sure that energy consumption at the relay can be kept at a minimum.
- **Relay transmission interval:** This is the minimum number of messages which the relay must receive before it can transmit a packet.
- **Packet Success Probability:** It it already known that the channel between the source and relay is more reliable than the channel between source and receiver due to closeness of the relay to the source. But how much reliable? In these

experiments we assume that channel between source and relay is two times better than the channel between source and receiver i.e. the packet erasure probability becomes half for the channel between source and relay. Let's assume the packet success probability between source and receiver is p_s then the packet success probability between source and relay will become $1 - \frac{1-p_s}{2}$.

We finally define our relay baseline as the normal windowed coding employed at the source and a relay node which does simple forwarding of one message that it receives. The relay waits to receive one packet from the source and once it is received successfully, the relay transmits it to the gateway. We also abstain from coding at the relay node and thus relay transmits only uncoded message. The relay transmission rate for this node is 1.

4.2 Windowed Coding at Relay

We now employ message coding at the relay intelligently. We create a similar network structure with a source, a relay node and a receiver as in the case of relay baseline. Here both the source and relay node follow the *Windowed Coding* scheme. The relay also receives the feedback from the gateway and since the channel between the relay and gateway is also more reliable the relay receives feedback more frequently as compared to the source.

4.2.1 Modifications to Windowed Coding for Relay

The generic *Windowed Coding* behaves differently when the sender receives a feedback from the receiver and when the feedback is missing. However, the relay does not have all the source symbols in its memory so we can't directly use the nascent Windowed Coding. It may so happen that even if the relay receives a feedback, it may not have required source symbols in its memory to take the desired actions. The general feedback structure as in Fig. 2.1 is used here. The feedback has two important parameters: the *oldest undelivered* symbol, s_u and the *number of missing symbols*, β in $[s_u, s_i]$. Based on these values the algorithm decides the degree of coding and choose the symbols to code.

Algorithm 1: Nascent Windowed Coding

```
Result: Message delivery at the receiver
Generate all the messages at the sender;
for all messages at the sender do
    if previous feedback is received then
        create a packet while choosing a coding degree using the 3.8;
    else
        create a packet using the no feedback degree of 2;
    end
end
```

Algorithm 2: Modified Windowed Coding at Relay

```
Result: Message delivery at the receiver
Receive the uncoded messages from the source and store in the memory;
Discard old messages which do not fit in the memory;
if number of messages in memory greater than equal to relay transmission
  interval then
    if previous feedback is received and oldest undelivered message in memory
      then
        create a packet while choosing a coding degree using the 3.8;
    else
        create a packet using the no feedback degree of 2;
    end
end
```

It is quite possible that the relay may not have all the symbols in this range. Suppose the oldest packet which is in relay's memory is s_{relay} such that s_u is older than s_{relay}, then is clear that we cannot send s_u directly like we do in nascent *Windowed Coding*. Secondly, the number of missing packets β in the interval $[s_u, s_i]$ does not make sense for the interval $[s_{relay}, s_i]$ which is a subset of $[s_u, s_i]$. Thus, the received feedback in this case turns out to be practically useless for the relay to make better decisions. So when this happens we do not utilize the feedback and assume it to be missing and thus create coded packets with degree 2. Finally the relay, sends packets and preform coding following the Algorithm 2 listed above.

4.3 Results

After setting up our system model we simulate it for various values of relay parameters like *'relay memory size'* and *'relay transmission interval'*, we discover some very interesting correlations. In this section we will only discuss the effects of relay parameters as role of other parameters has already been discussed in previous chapters.

4.3.1 Delivery Failure Rate

Delivery Failure Rate (DFR) is the ratio of number of messages undelivered to the total number of messages transmitted. We observe that as we increase the *relay memory size* to a limit (a little higher than δ_{max}) the DFR reduces sharply and the coding scheme performs better than the baseline. For lower values of *memory size* the baseline performs very similar to the relay (coding). Similarly, when the *relay transmission interval* is reduced the DFR reduces owing to higher number of transmissions from the relay. Our goal is to get better performance while keeping the number of relay transmission to a minimum. So, the higher the *relay transmission interval* the more energy efficient is the overall scheme. We get better performance than the baseline for reasonably high *relay transmission interval* but the performance deteriorates for very high values. This trend holds irrespective of the type of channel.

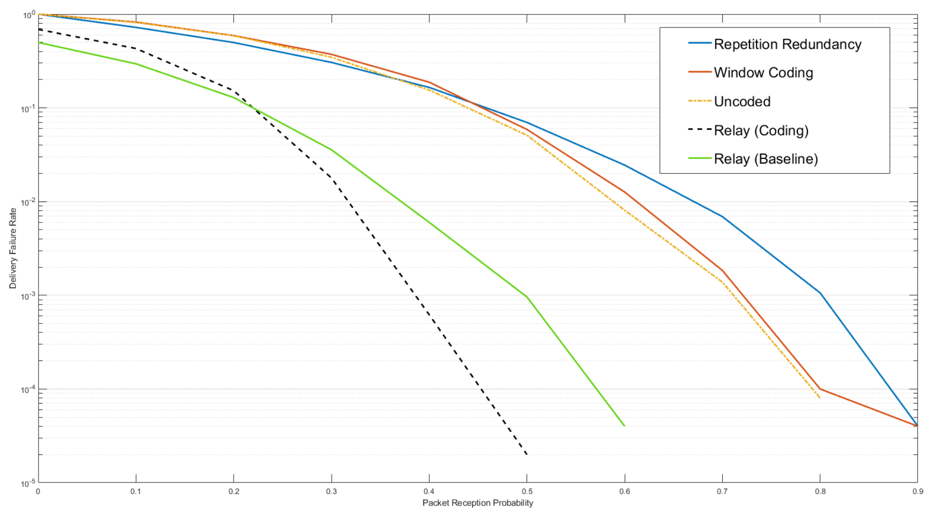


Figure 4.2: Delivery Failure Rate v/s Packet Reception for a Bernouli Erasure Channel: The *relay memory size* is 20 and *relay transmission interval* is 1. It is evident that even adding a simple relay will improve the performance as shown in the baseline. However, our scheme performs better than a simple forwarding relay.

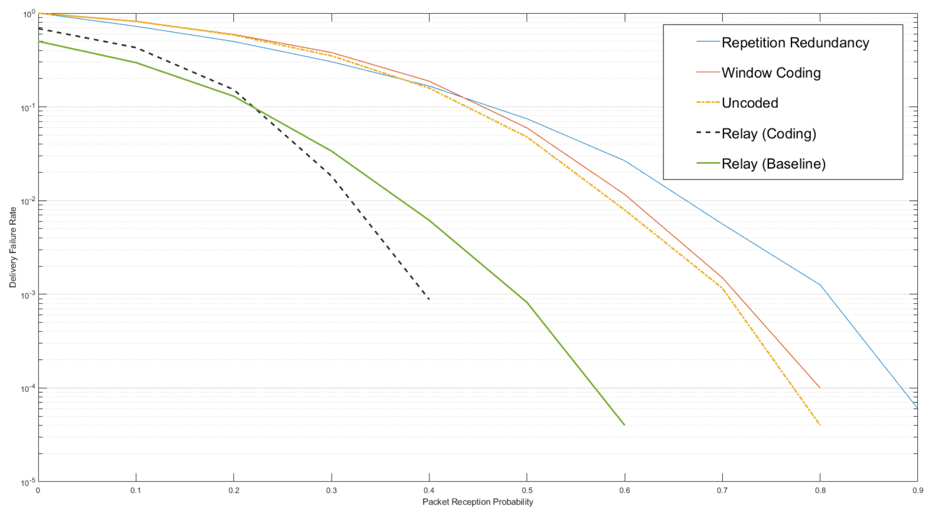


Figure 4.3: Delivery Failure Rate v/s Packet Reception for Bernouli Erasure Channel: The *relay memory size* is 20 and *relay transmission interval* is 10. Increasing the *relay transmission interval* hampers the performance slightly.

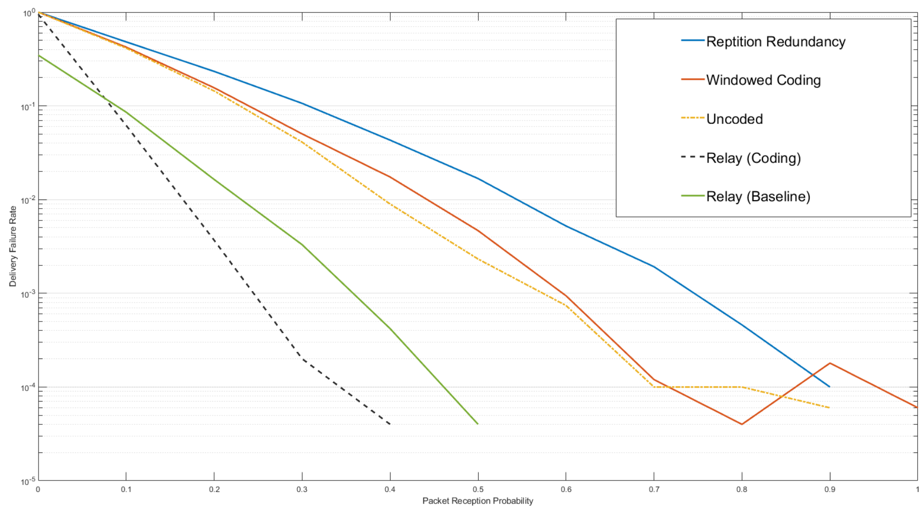
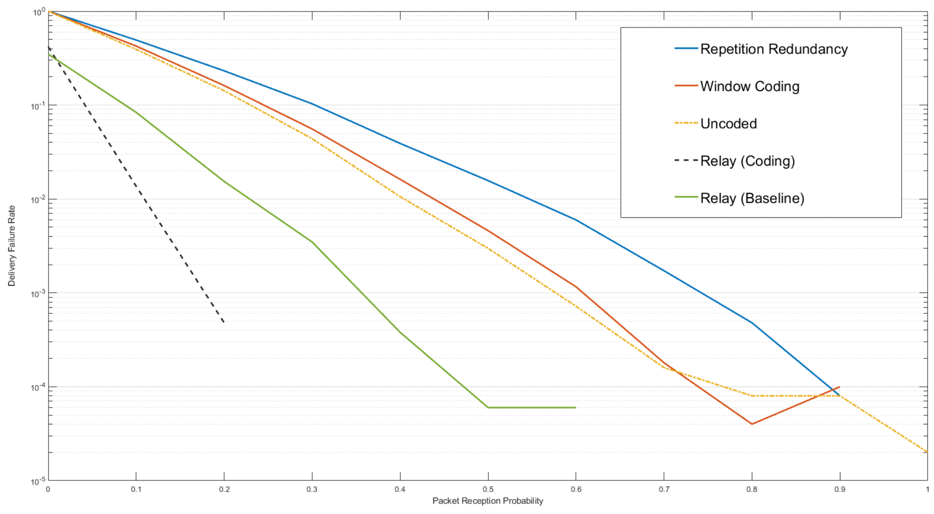


Figure 4.4: Delivery Failure Rate v/s Packet Reception for a Gilbert-Elliot Erasure Channel: The *relay memory size* is 20 and *relay transmission interval* is 5. The performance of our method is even better than the Bernouli Channel.



(a)

b

Figure 4.5: Delivery Failure Rate v/s Packet Reception for Gilbert-Elliot Erasure Channel: The *relay memory size* is 20 and *relay transmission interval* is 10.

4.3.2 Energy Consumption

Energy consumption is a vital parameter for any wireless communication network since most of the wireless devices run of a battery so it is critical to ensure we no energy is wasted during transmission. Most of the energy is consumed transmission of a packet and is proportional to the length of the packet. Every transmission in our design is of fixed packet length so if we count the total number of symbols we transmit we can get an approximate value of the total energy consumed. The total energy consumed for the relay setup would obviously be higher due to the presence of two elements which can transmit so comparing it with the cases when there is no relay will not be fair. Thus, we calculate the average energy consumed per transmitting source by dividing the total energy consumed by 2. We find that the average energy consumed per transmitting source on introducing the relay is smaller when there was no relay (i.e. only a single source). The extra transmissions made by the relay actually reduce the number of direct transmissions made by the source and thus reduce the average consumed by it too.

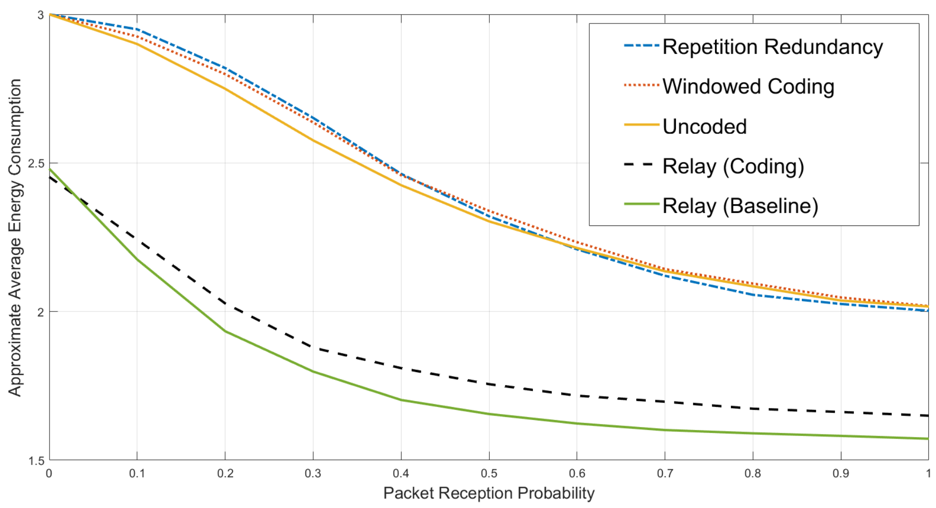


Figure 4.6: Average Energy Consumed v/s Packet Reception for Gilbert-Elliot Erasure Channel: The *relay memory size* is 20 and *relay transmission interval* is 10.

4.4 Summary

In this chapter we introduced a relay node in our setup and we found that if we do modified windowed coding at the relay with the symbols available at it we can vast improvements in Delivery Failure Rate and Average Energy Consumption per transmitting source. Introducing a relay in any network improves performance but we demonstrate that our method performs better as compared to introducing a naive relay node in the network.

The goal of this thesis was to improve upon the existing *Windowed Coding Scheme* introduced first in [1]. By the end of it we were able to take substantial steps towards our objective. Finally we introduced a coding scheme or a message transmission protocol which can be utilized in any wireless sensor network to reliably transmit information while being energy and computationally efficient. There are still multiple avenues for improvement namely joint optimization of degree, storing coded messages at relay, buffer at receiver, relay node using the uncoded algorithm, adaptive packet sizes during transmission etc. which are still under our consideration and may help us improve our methods in the future.

Bibliography

[1] S. S. Borkotoky, U. Schilcher, and C. Raffelsberger, “Application-layer coding with intermittent feedback under delay and duty-cycle constraints,” in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[2] M. Luby, “Lt codes,” in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, 2002, pp. 271–280.

[3] A. Shokrollahi, “Raptor codes,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[4] E. Drinea, L. Keller, and C. Fragouli, “Real-time delay with network coding and feedback,” *Physical Communication*, vol. 6, pp. 100 – 113, 2013.

[5] S. L. Fong, S. Emara, B. Li, A. Khisti, W.-T. Tan, X. Zhu, and J. Apostolopoulos, “Low-latency network-adaptive error control for interactive streaming,” in *Proceedings of the 27th ACM International Conference on Multimedia.* Association for Computing Machinery, 2019, p. 438–446.

[6] P. J. Marcelis, V. S. Rao, and R. V. Prasad, “Dare: Data recovery through application layer coding for lorawan,” in *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2017, pp. 97–108.

[7] “Sx1272/3/6/7/8: Lora modem designer’s guide,” *ANI200.13 Semtech Corporation*, 2013.