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

December 2020

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 Symbols

$\begin{pmatrix} k \\ l \end{pmatrix}$	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

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

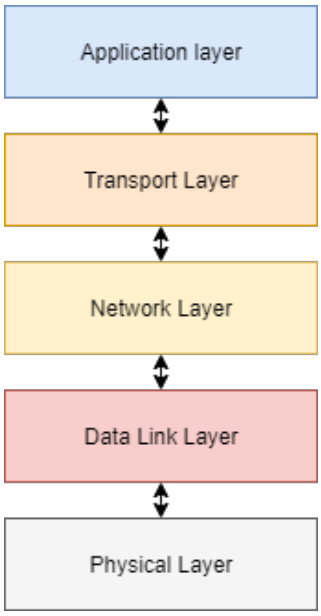

Figure 1.1: Various layers of the TCP/IP protocol. The Application Layer sits at the top

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.

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.

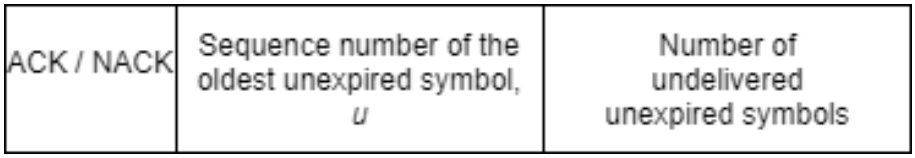

Figure 2.1: Structure of the feedback packet

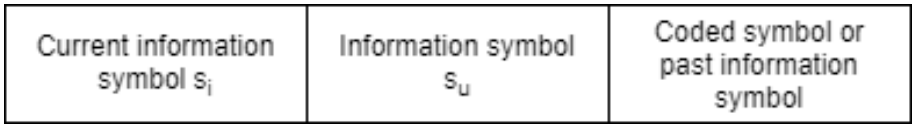

Figure 2.2: Packet structure of size 3 when feedback has been received for the previous packet

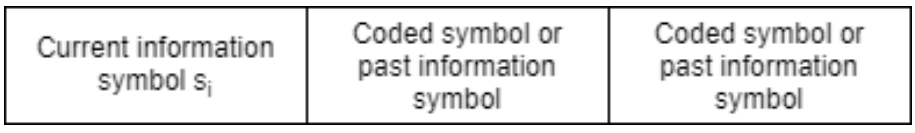

Figure 2.3: Packet structure of size 3 when feedback has not been received for the previous packet

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 $O(\delta_{max}^2)$. 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.

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

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

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

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

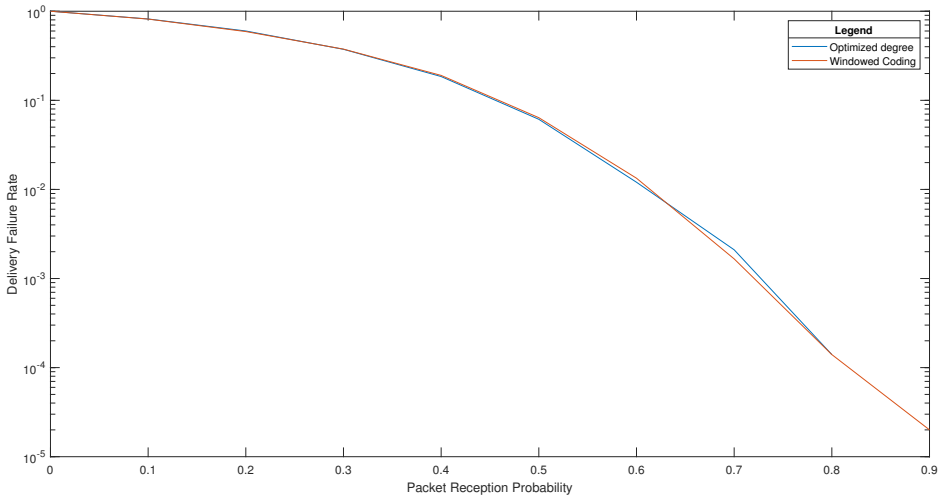

Figure 3.1: The results obtained when simulating the original windowed coding scheme (red) and the same coding scheme with better degree optimization (blue). It is clearly visible that the curves are very similar.

(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.

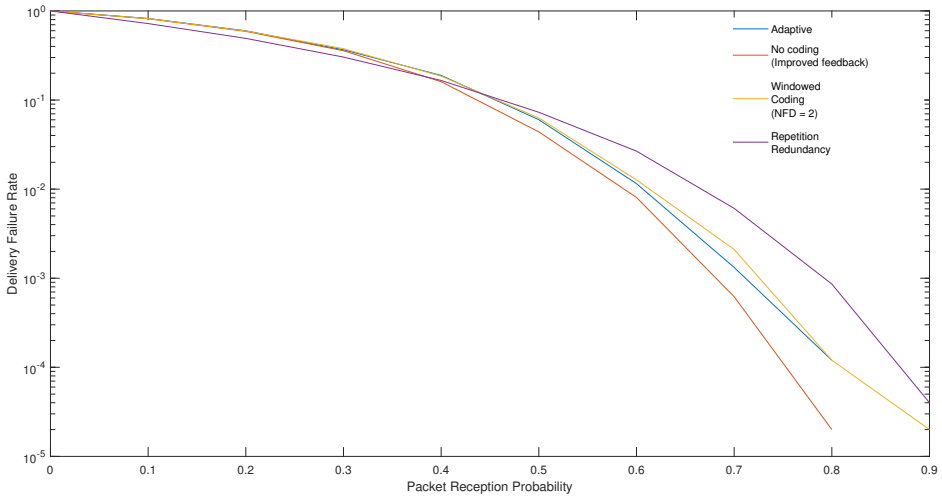

Figure 3.2: Comparison between baseline Repetition Redundancy (purple), Windowed Coding (yellow), Adaptive scheme (blue) and No coding or improved feedback (red). The simulation is done when delay tolerance $\delta_{max} = 4$ *and* $l_b = 4$. The switching to the original scheme in adaptive takes place 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.

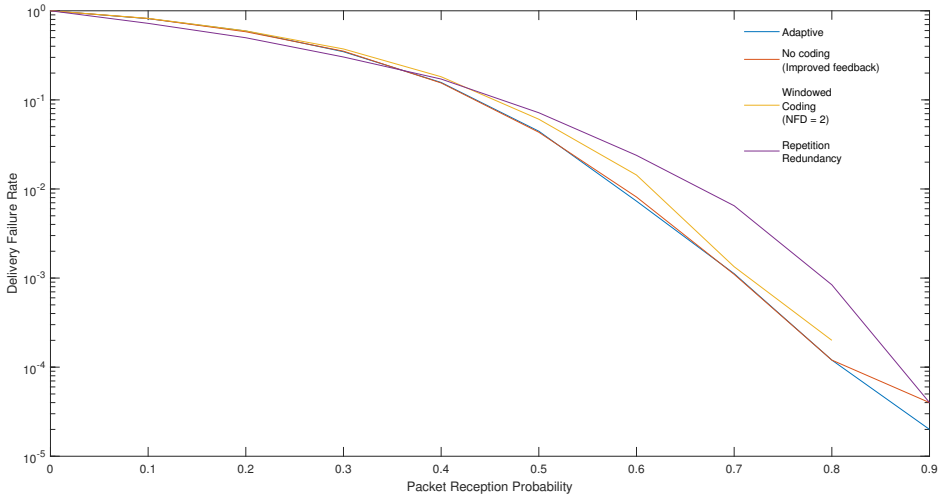

Figure 3.3: Similar simulation as in Fig. 3.2 but with $\delta_{max} = 16$ i.e $\delta_{max} > l_b$. The no coding performance deteriorates because now most of the missing symbols are outside the range in which it can provide information i.e. in set $[s_{u+l_b+1}, s_i]$.

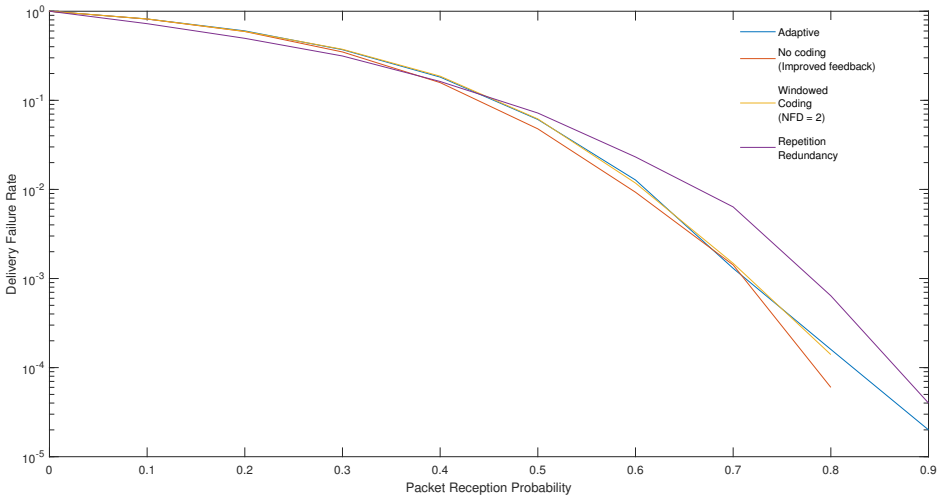

Figure 3.4: Similar simulation as in Fig. 3.3 but with $\alpha_0 = 1.5$. The no coding performance is similar to what was before because now also most of the missing symbols are outside the range in which it can provide information i.e. in set $[s_{u+l_b+1}, s_i]$. The adaptive scheme's performance becomes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

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.

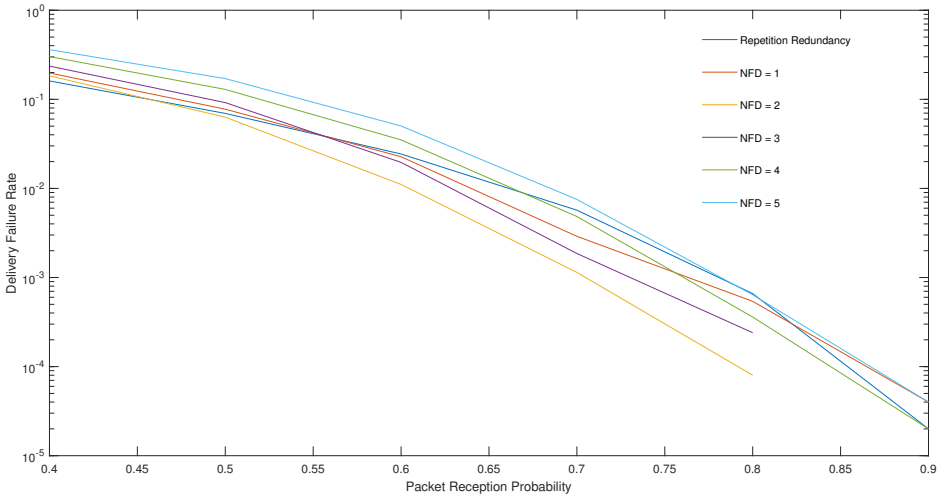

Figure 3.5: Delivery Failure Rate v/s Packet Reception Probability with different values of No Feedback Degree (NFD).

We conduct some experiments with various fixed degrees while coding when no feedback is present. The *Delivery Failure Rate* 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. On initial thought, the best performance of $NFD = 2$ can be explained by the decoding process. 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 deocde and thus the delivery failure rate is higher.

3.3 Future Works

In the previous section we have described the analysis that has been completed yet. There are many other things to explore. In coming months our focus will be on the

adaptive scheme and how it can be further improved. At the moment, there are some issues with the scheme. Furthermore, there is no theoretical ground to degree selection when there is no feedback. We would explore how to quantify the no feedback state and how to choose optimal degree based on the information which we might have from previous feedback. Even when feedback is present the estimate for optimal degree is suboptimal because when a single payload contains more than one coded symbol joint optimization of degree over both the symbols should be looked into. In this work, the focus has been on the *windowed coding* scheme whereas the paper [1] lists one more algorithm—*selective coding*. *Selective coding* should also be looked into to improve its performance. Apart from all this we also need to evaluate our methods on different channel erasure models as currently we are focusing on Bernouli and Gilbert-Elliot channels. Finally, the role of a buffer at the receiver should also be investigated.

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.