This is to certify that the
thesis entitled

Side-Information Enhanced Localized Puncturing for Rate-Adaptive, Reliable and Stable Wireless Protocols

presented by

Ahmed Majeed Khan

has been accepted towards fulfillment
of the requirements for the

M.S degree in Electrical Engineering

Major Professor's Signature

August 9, 2010
Date

MSU is an Affirmative Action/Equal Opportunity Employer

SIDE-INFORMATION ENHANCED LOCALIZED PUNCTURING FOR RATE-ADAPTIVE, RELIABLE AND STABLE WIRELESS PROTOCOLS

By

Ahmed Majeed Khan

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Electrical Engineering

2010

ABSTRACT

SIDE-INFORMATION ENHANCED LOCALIZED PUNCTURING FOR RATE-ADAPTIVE, RELIABLE AND STABLE WIRELESS PROTOCOLS

By

Ahmed Majeed Khan

Time varying wireless channels are inherently susceptible to more errors than classical (wired) media due to their vulnerable nature. The errors which are not corrected by the physical layer are called *Residual Errors,* and residual errors result in link layer packet drops. Design of wireless protocols that exploits *useful data,* the data which was received correctly, in a received corrupted packet can benefit network's throughput substantially. A common way to control errors depends on feedback from channel and several previous works made use of *Rate-Compatible Punctured Codes* to adapt code rate to channel conditions but none of these took into account the bursty nature of wireless channels. The primary objective of this thesis is to fill this gap.

In our work we consider side-information enhanced localized puncturing to adapt the rates of channel codes to cope with time-varying wireless channel conditions. To give proof of our idea, we employ rate-adaptive hybrid framework in the domain of LDPC codes and embed them in wireless networking protocols. To the best of our knowledge, this is the first effort to enhance puncturing efficiency using side-formation from PHY/MAC layer and perform the puncturing operation in a localized manner. The proposed approach does not require any modification in subsequent layers, and it achieves significant gains in throughput efficiency, while ensuring reliable and stable transmission of data.

To my parents

ACKNOWLEDGMENTS

All glory be to the most praiseworthy who showered His countless blessings on me throughout the span of my life. With the prolific praise of the Owner of Honor, I desire to begin a limitless praise, with which He is pleased and I cite His own words; *"All praise to Allah, the Lord of the worlds"*. **The Quran (39:75).**

I am honored to have one of the most distinguished scholars, Professor Hayder Radha, as my advisor. I would like to thank him for his support throughout my stay at MSU. I found him an extremely intelligent researcher, exceptionally intellectual advisor and an excellent teacher. This work could not have been possible without his ever critical attitude towards my neophyte research ideas, followed by his tremendous encouragement to heighten my confidence.

I am obliged to my family, especially my parents, for their selfless support throughout my life, and particularly during the span of this work. I am running short on words to express my sentiments towards my parents for their gracious deeds and self-sacrificing exertions to empower my advancement. Accomplishment of this work is an implicit consequence of my Fathers vigorous attitude to boost my motivation and enthusiasm, and my Mothers affectionate prayers. I am also thankful to my Baba Jee, Nani Amma, Ayesha, Amenii, Rabii, Salahudin, Marium and Iqra for their unbounded yet unconditional love, and having absolute faith in me.

Life in the US could not have been easier in the absence of steadfast friends and wonderful colleagues and they deserve a special mention here. Stating in alphabetical

order, I found companions like Awais, Brig. Qaiser, Hassan Aqeel, Jawad, Khawar, Sajjad bhai, Shahzad, Tariq bhai, Usman bhai, Waqar, and Zubair who kept my morale high and made my stay a superb experience, here at MSU.

Lastly I want to extend my gratitude to my advisory committee members, Professor Jack Deller and Professor Jonathon Hall, for their uninterrupted patronization during the course of this work. Their extensive feedbacks not only opened new horizons of research for me, but also endowed me with insights into intricate engineering concepts. I am fortunate to have their persistent contribution and involvement in my work.

Table of Contents

List of Tables

List of Figures

Abbreviations and Acronyms

ACE	Automatic Code Embedding protocol
ACK	Acknowledgement
ADC	Analog to Digital Convertor
ARQ	Automatic Repeat Request
BEC	Binary Erasure Channel
BER	Bit Error Rate
BP	Belief Propagation algorithm
BSC	Binary Symmetric Channel
c-node	Check Node
CDF	Cumulative Density Function
CRC	Cyclic Redundancy Check
CS	Conventional Standard protocols
CSI	Channel State Information
DAC	Digital to Analog Convertor
dB	Deci Bell
DH	Data Host PC

DE	Density Evolution
DG	Data Ghost computer
E2E	End-to-End
Eb/No	Energy per Bit to Noise power spectral density ratio
ED	Error Detection
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
GHz	Giga Hertz
HARQ	Hybrid Automatic Repeat Request
HEEP	Hybrid Erasure-Error Protocols
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
LDPC	Low-Density Parity-Check codes
LLR	Log Likelihood Ratio
LODE	LDPC Online Density Evolution
LOPT	LDPC Online Optimization Tool
LoS	Line of Sight

MAC	Medium Access Control layer
MIT	Massachusetts Institute of Technology
MS/s	Million Samples per Second
NACK	Negative Acknowledgement
PDF	Probability Density Function
PHY	Physical Layer
PER	Packet Error Rate
pmf	Probability Mass Function
PRAISE	Puncturing for Rate-Adaptability of Reliable and Stable wireless protocols
RA	Repeat Accumulative
RASE	Reliable and Stable protocol
RF	Radio Frequency
RHC	Rate-adaptive Hybrid (erasure-error) Codes
RS	Reed Solomon codes
SDR	Software Defined Radio
SEFEC	Side-information Enhanced transmission in presence of Forward Error Correction

SNR	Signal to Noise Ratio
TCP	Transmission Control Protocol
USB	Universal Serial Bus
UDP	User Datagram Protocol
USRP	Universal Software Radio Peripheral
v-node	Variable Node
var	Variance

Chapter 1 - Introduction

1.1- Motivation:

Despite the tremendous promise of computer networks, they suffer from errors and losses in the presence of network congestion and transmission medium degradation. This results in decreased end-to-end network bandwidth utilization. Wireless channels are more prone to errors than wired media due to their highly time-varying nature. The design considerations for the wireless networks differ substantially from the wired media due to the increased error-rate. The designers of the wireless networks introduce enhanced robustness at the physical (PHY) and MAC layers of the protocol stack in order to cater for these errors. Some of these errors are not corrected by the physical layer, called as *Residual Errors,* and such errors result in link layer packet drops. Traditional network protocols like IEEE 802.11 ARQ schemes drop packets and request retransmission even if a single bit is in error. This design strategy leads to the wastage of *useful* data in the packet, the data which was received correctly. Design of wireless protocols that exploits *useful data* in a received corrupted packet can benefit network's throughput substantially and can result in significant improvement of end-to-end network utilization.

Some previous works like Hybrid ARQ I/II (HARQ-I/II) [38, 39] use some sort of error correction to improve wireless networks bandwidth. In fact more recent works like Automatic Code Embedding (ACE), Reliable and Stable (RASE), and ZIPTX [34, 40, 37]

exploit partial packet recovery to harness the power of *useful* data in a corrupted packet. All of these works rely on some sort of error correction scheme and require feedback from the receiver about channel conditions.

A common way to control errors is to rely on the *Channel State Information,* and uses it to predict suitable code rate under a given channel condition. In this way, controlled redundancy is added to data based on predicted channel conditions. So the worse the channel conditions are, lesser is the code rate and vice versa. Several previous works [29, 30, 31, 32, 33] used puncturing to adapt to the time-varying network conditions and the resultant codes are called as *rate-compatible punctured codes*. Jeongseok et al. [36] applied this idea to LDPC codes to find rate-compatible punctured LDPC codes with a single decoder for 'mother' and 'daughter' codes. Puncturing can be viewed as an erasure channel; hence, puncturing over a wireless channel leads to the reception of hybrid erasure-error codes at the decoder end. But the above-mentioned works [29, 30, 31, 32, 33, 36] focus on inherent code properties to design channel codes, and do not take into account the channel properties while designing puncturing patterns. In this way, there is a room to design puncturing patterns, based on wireless channels characteristics and inherent properties, to adapt channel code rates to time-varying wireless channel conditions. The primary objective of this thesis is to fill this gap.

We define *a channel to be in 'On' ('Off') state if packets transmitted through it experience small (large) number of errors*. Wireless channels are bursty in nature and a channel in 'On' ('Off') state has higher probability of staying in 'On' ('Off') state than to

go into 'Off' ('On') state. In other words, wireless channels have memory such that next state is dependent on previous state(s), which determines the depth of Markov (hidden Markov) chains used to model them. So, if a channel is in 'Off' state, it experiences errors at all (bit/byte/packet) levels in a bursty fashion. We take into account this inherent property of wireless channels to design puncturing patterns to adapt code rates to cope with bursty channel errors. Furthermore, we rely on side-information in the form of reliabilities of source symbols received from PHY/MAC layer to design these puncturing patterns and the retransmission data. In more explicit words, this thesis focuses on designing localized puncturing patterns for rate-adaptability of wireless channel protocols to cope with vulnerable, error-prone, and time-varying wireless channel conditions.

1.1- Research Problem:

In our work, we consider localized puncturing for channel codes, particularly focusing on LDPC codes. We show that localized puncturing can lead to construction of rate-adaptive hybrid erasure-error codes (RHCs), which in turn, can be used to adapt rate of channel codes in network protocols to perform well in varying network conditions. To the best of our knowledge, this is the first effort to enhance puncturing efficiency by performing it in a localized manner, leading to RHCs, using side-formation from PHY/MAC layer. To give proof of our idea, we employ rate-adaptive hybrid framework in the domain of Low-Density Parity-check (LDPC) codes and embed them in wireless networking protocols.

A discrete random process with the property that the next state depends only on the current state is called as ***Markov Chain***. The Markov property states that the conditional probability distribution for the system at the next step, and in fact at all future steps, given its current state depends only on the current state of the system, and not additionally on the state of the system at previous steps. We define *a channel to be in 'On' ('Off') state if packets transmitted through it experience small (large) number of errors*. Wireless channels are bursty in nature and a channel in 'On' ('Off') state has higher probability of staying in 'On' ('Off') state than to go into 'Off' ('On') state. In other words, wireless channels have memory such that next state is dependent on previous state(s), which determines the depth of Markov (hidden Markov) chains used to model them. So, if a channel is in 'Off' state, it experiences errors at all (bit/byte/packet) levels in a bursty fashion. We take into account this inherent property of wireless channels to design puncturing patterns to adapt code rates to cope with bursty channel errors. Furthermore, we rely on reliabilities of source symbols received from PHY/MAC layer to design these puncturing patterns and the retransmission data.

In order to keep the discussion generic and not dependant on a particular implementation or standard, we consider three rather abstract communication schemes: 1) transmission over error/erasure channels, which represents the **C**onventional **S**tandard (CS) protocols, like IEEE Automatic Repeat Request (ARQ) schemes; 2) transmission in the presence of a **F**orward **E**rror-**C**orrection (FEC) based schemes, like Hybrid ARQ [38, 39] and more recent schemes like ZIPTX [37], Automatic Code Embedding (ACE) [34], and Reliable and Stable (RASE) [40]; 3) **S**ide-information

Enhanced (transmission in presence of both, erasures and errors,) using a **F**orward-**E**rror Correction scheme (SEFEC) like the proposed scheme, **P**uncturing for **R**ate **A**daptability of Reliable and **S**tabl**e** (PRAISE) Wireless Protocols.

The three communication schemes considered by us can be explained by considering two generic aspects: 1) forward error correction; 2) feedback mechanism. Forward-error correction part can simply be segregated into two categories consisting of presence or absence of FEC. Feedback mechanism can have many variants, ranging from simple binary feedback about reception of correct/false packet like in case of ARQ, to feedback with flags requesting additional parity data for corrupted packets like in case of recent protocols like ACE [34] and RASE [40]. So under these two aspects, the following happens.

1) CS: the conventional standard protocol contains no forward error correction and requests the same packet even if a single bit is in error, example being IEEE802.11 ARQ scheme. In IEEE802.11 ARQ protocol, error-detection information (ED) bits are added to data to be transmitted (such as cyclic redundancy check, CRC) and retransmissions are solution for losses. If a corrupted packet is received, it is discarded without regard to the number and location of errors. This methodology ensures that the packet would eventually be recovered. However, this approach leads to a great deal of throughput degradation as even a single bit error leads to packet drops resulting in discard of a large number of correctly received data bits, ending up 'wasting' a lot of

useful data. Thus, network utilization deteriorates rapidly as channel bit error rate (BER) increases.

2) FEC: FEC based protocols, which represent schemes like IEEE802.11 Hybrid ARQ schemes [38, 39] and more recent schemes like ZIPTX [37], ACE [34] and RASE [40], contain some sort of forward error correction and request additional parity data in case of reception of a corrupted packet. Hybrid ARQ (HARQ) protocols are proposed as an alternative to CS. In HARQ protocols, forward error correction (FEC) bits are also added to the existing Error Detection (ED) bits, such as Reed-Solomon code, low-parity density-check code or Turbo code. In this way, these schemes reduce the number of re-transmissions as required in case of ARQ by making use of incremental channel codes. However, both ARQ and HARQ based approaches do not address the throughput stability issues in varying channel conditions, and thus lead to a great deal of throughput inefficiency. Other protocols like ZIPTX and ACE address the issues of reliability and/or stability in varying channel conditions but these fall short on many fronts. ZIPTX, though provides a working system, ignores any aspect of stability. ACE takes into account both the issues of reliability and stability by embedding channel codes using well-defined code rates but ACE, i) lacks a practical demonstration system, ii) do not consider rate-adaptability to address throughput efficiency in changing network conditions.

3) SEFEC: SEFEC is an alternative to above schemes. Similar to FEC based schemes, an SEFEC scheme requests additional parity using feedback flags, but the requested parity data is localized and request depends upon side-information received from physical (MAC) and link layers. So SEFEC is a cross-layer design which extracts beliefs associated with received data either at MAC/physical layer, or at link layer and decides to requests additional parity data for a particular subset of code depending on values of beliefs for a particular subset of code. Some previous works [35, 41, 42] have proposed cross-layer designs for multimedia applications to overcome throughput degradations and performance limitations imposed by traditional protocols. An example of such a work is UDP Lite [41], which relies on the error-resilient nature of multimedia content and makes adjustments to the protocol stack at the transport and the link layers in order to improve the bandwidth utilization. A major drawback of the existing cross-layer protocols is that their implementations require key modifications in transport and application layers. However the proposed cross-layer SEFEC design requires no modification in the subsequent layers since a single decoder is required for 'mother' and 'daughter' codes and side-information used to request parity data is received from physical (MAC) and link layers. Therefore, and unlike receivers of FEC- based schemes, SEFEC receiver has RHCs, is rate-adaptive depending on varying channel conditions, distinguishes between erasures and errors in a received packet, and does not require any modification in transport and application layers.

Suppose we have k data-bits to be transmitted over a wireless channel and we make n bit codeword with those k bits such that rate of the code is:

$$r = \frac{k}{n}; \quad k < n \qquad \ldots\ldots\ldots (1.1)$$

It has been a tradition to define LDPC codes in terms of its degree-distribution pair $(\lambda(x), \rho(x))$ following [11] such that:

$$\lambda(x) = \sum_{i=2}^{d_l} \lambda_i x^{i-1} \qquad \ldots\ldots\ldots (1.2)$$

$$\rho(x) = \sum_{i=2}^{d_l} \rho_i x^{i-1} \qquad \ldots\ldots\ldots (1.3)$$

Then, the rate of the code in terms of its degree distributions pair become:

$$r(\lambda, \rho) = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} \qquad \ldots\ldots\ldots (1.4)$$

Puncturing increases code rate as we decrease n for fixed k bits while doing puncturing. Since the decoder for lowest rate 'mother' code is compatible with higher rate 'daughter' codes, no additional complexity is required for puncturing. We define $p^{(0)}$ to be the total puncturing fraction following [36],

$$p^{(0)} = \frac{the\ number\ of\ punctured\ variable\ nodes}{the\ total\ number\ of\ variable\ nodes} \qquad \ldots\ldots\ldots (1.5)$$

Then rate of 'daughter' code, $r(\lambda,\rho)^{\sim}$, is related to the rate of base/'mother' code as:

$$r(\lambda,\rho)^{\sim} = \frac{r(\lambda,\rho)}{1 - p^{(0)}} \qquad \ldots\ldots\ldots (1.6)$$

Several recent works tried to find bounds on code rates for reliable communication. Some recent efforts like ACE [34] and RASE [40] developed a framework for reliable and stable communication with guarantees on sustainable flows. If ε_i is BSC channel's error (cross-over) probability and α_m is the error-correction capability of a code, then ACE determines the code rate to be the following for reliable communication:

$$R = 1 - \frac{\varepsilon_i}{\alpha_m} \qquad \ldots\ldots\ldots (1.7)$$

So if a channel is in 'good' state, we can increase the code rate and still have a reliable and stable communication with same guarantees on sustainable flows. So for a 'daughter' code with error-correction capability α_d, we determine puncturing fraction as:

$$p^{(0)} = 1 - \frac{r(\lambda,\rho)}{1 - \frac{\varepsilon_i}{\alpha_d}} \qquad \ldots\ldots\ldots (1.8)$$

At the encoder end, we divided n-bit codeword into c chunks or partitions in a manner pre-determined by encoder and decoder. There are reliabilities associated with each variable bit as data is received at the decoder. Belief Propagation algorithm identifies l partitions of the lowest reliability and requests additional parity data for those l

partitions. Please note that we need $f = \left\lceil log_2 \binom{c}{l} \right\rceil$ feedback bits to identify l out of c chunks of lowest reliability. Based on this description, the flow chart of the proposed scheme can be shown as in figure I-I below:

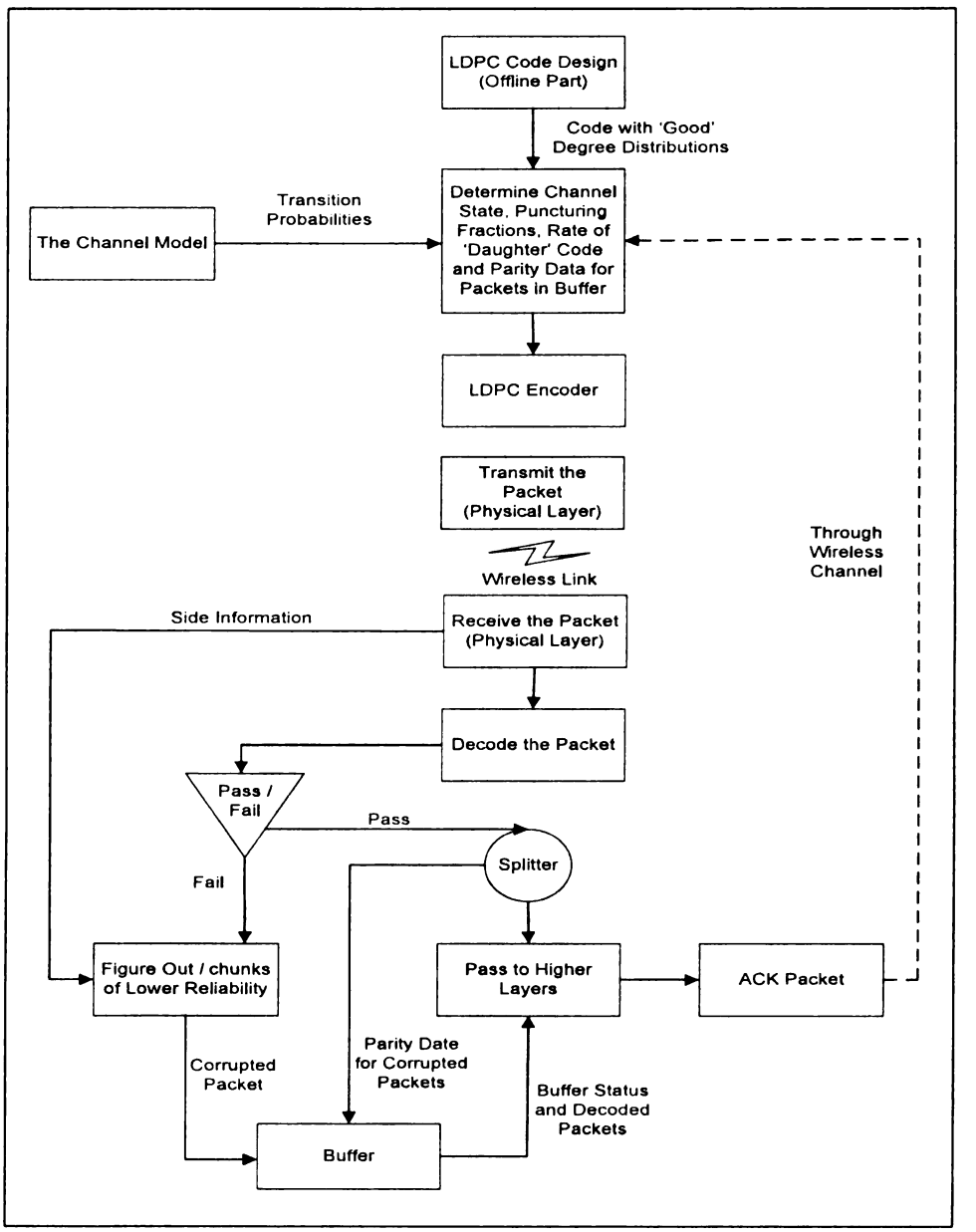

Figure I-I: Flow Chart of PRAISE

The basic idea is as follows: suppose a 'mother' code is designed with rate, $r(\lambda, \rho)$, with highest redundancy and containing c partitions. This code, with the largest number of parity bits, can be used in the worst channel condition. We do puncturing determined by channel error probability ε_i to transmit a 'daughter' code with error-correction capability α_d. If decoding fails, decoder identifies l partitions of lowest reliability and requests parity data for those l partitions using f additional feedback bits. Encoder encodes additional parity bits with new packet and transmitter transmits that. This process continues till whole data is successfully received and decoded.

So, our contributions can be summarized as follows:

1) We present new design architecture for rate adaptability of wireless protocols using side-information enhanced localized puncturing to cater for corrupted packets received in time-varying wireless channel conditions. This approach does not require any modification in subsequent layers, and it achieves significant gains in throughput efficiency, while ensuring reliable and stable transmission of data.

2) GNU Radio is an open-source, free software toolkit for deploying Software Defined Radio (SDR) based communication systems and Universal Software Radio Peripheral (USRP) is its RF front-end. We use GNU-USRP based platform to evaluate real-time wireless channel conditions using real-time traffic data, and carry out comparative performance analysis of PRAISE and other link-layer

protocols by implementing and testing these over the GNU-USRP based platform.

1.2- Thesis Organization:

Rest of the document is arranged as follows. Chapter 2 provides essential background information on LDPC codes and channel types under consideration. Chapter 3 begins with the code design aspects that we took into account to propose rate-adaptive codes using localized puncturing, and continues with the discussion of designing puncturing fractions for rate-adaptability of channel codes in time-varying channel conditions. Chapter 4 provides the details of the channel model to develop the proposed scheme. Chapter 5 begins with the discussion of Software-Defined Radios (SDRs) and explains GNU-USRP based implementation of an SDR to be used in our work. It also provides information about our experimental setup and our motivation to use GNU-USRP platform. Chapter 6 presents the analysis on the element of burstiness in collected traces, and in Chapter 7, we present results, conclusions and future directions.

Chapter 2 - Background

In this chapter we provide necessary background information on LDPC codes, puncturing, residual errors and channel types under consideration.

2.1- LDPC Codes:

A low-density parity-check code (LDPC code) is a linear error correcting code and is constructed using a sparse bipartite graph. LDPC codes are also known as Gallager codes, in honor of Robert G. Gallager, who developed the LDPC concept in his doctoral dissertation at MIT in 1960 [2]. LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close (or even arbitrarily close on the BEC) to the theoretical maximum (the Shannon limit) for a symmetric memory-less channel. The noise threshold defines an upper bound for the channel noise up to which the probability of lost information can be made as small as desired. Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block length.

2.1.1- Encoding of LDPC codes:

LDPC codes are linear codes, and can be represented in the following two ways. First way is the graph representation. Suppose a bipartite graph G has n left nodes and r right nodes. The graph G can give rise to a linear code of block length n and dimension at least $(n - r)$ if the n coordinates of the code-words are associated with the n message nodes. Here, the left nodes are called as *Message Nodes* while the right nodes are called as *Check Nodes*. The code-words are those vectors $(C_1, C_2, \ldots, C_n)$ such that

for all check nodes the sum of the neighboring positions among the message nodes is zero.

Another way of looking into the representation of LDPC codes is the matrix representation. Suppose $\mathbf{H}$ is a binary $(r * n)$ matrix in which the entry $(i; j)$ is 1 if and only if the i^{th} check node is connected to the *jth* message node in the corresponding graph. So in this case, the LDPC code defined by the graph is the set of vector $C = (C_1, C_2, \dots, C_n)$ such that ($\mathbf{H} \,.\, C^T = 0$). In this case, the matrix $\mathbf{H}$ is called the *Parity Check Matrix* for the code. Conversely speaking, any binary $(r * n)$ matrix gives rise to a bipartite graph between n message and r check nodes, and the null space of $\mathbf{H}$ defines the code associated to this graph.

Many specific constructions can be encoded in time approximately or exactly linear in the code size. For example, Repeat-Accumulate (RA) or serial-repeat-accumulate codes have an explicit linear-time encoding method. For LDPC codes design, the following approaches are noteworthy:

- The original regular LDPC codes due to Gallager with an $\mathbf{H}$ matrix [2]
- MacKay Codes with sparse $\mathbf{H}$ matrices [21, 22]
- Irregular LDPC codes by Richardson et al. [11] and Ruby et al. [3]
- Finite Geometry Codes [20, 23]
- Repeat Accumulate (RA) by D. Divalsar et al. [24]
- Irregular Repeat Accumulate (IRA) by H. Jin et al. [25]
- Extended Irregular Repeat Accumulate (eIRA) by Yang and Ryan [5, 6, 7]

- Array Codes by Fan [26, 27] and Eleftheriou [9]
- Combinatorial LDPC Codes [13, 14, 18, 19, 20]

2.1.2- Decoding of LDPC Codes:

LDPC codes are often decoded using some type of iterative message-passing decoding algorithms which are iterative in nature. An example of such an algorithm is *Belief Propagation (BP)*, and many other message-passing algorithms can be viewed as approximations to BP. The reason for their name is that at each round of the algorithm, messages are passed from message nodes to check nodes, and from check nodes back to message nodes. The messages from a check node to message nodes are computed based on the observed value of the check node and some of the messages passed from the neighboring message nodes to that check node. It is important to note that the message that is sent from a check node c to a message node v must not take into account the message sent in the previous round from v to c. The same is true for messages passed from message nodes to check nodes.

One important and the most commonly used subclass of message passing algorithms is the *Belief Propagation (BP) Algorithm*. This algorithm is present in Gallager's work [2], and it is also used in the Artificial Intelligence community [10]. In BP, the messages passed along the edges in this algorithm are probabilities, or beliefs. So specifically, the message passed from a message node v to a check node c is the conditional probability that v has a certain value, given the observed value of that message node, and all the values communicated to v in the prior round from check nodes incident to v other than

c. Similarly, the message passed from c to v is the probability that v has a certain value given all the messages passed to c in the previous round from message nodes other than v. Mostly authors prefer working with likelihoods, or sometimes log-likelihoods instead of probabilities and we adapt the log-likelihood approach.

For a binary random variable x, let $L(x) = \Pr[x = 0] / \Pr[x = 1]$ be the likelihood of x. Given another random variable y, the conditional likelihood of x denoted $L(x|y)$ is defined as $\Pr[x = 0 \mid y] / \Pr[x = 1 \mid y]$. Similarly, the log-likelihood of x is $\ln L(x)$, and the conditional log-likelihood of x given y is $\ln L(x \mid y)$. By Baye's rule, we know that $L(x|y) = L(y|x)$ if x is an equi-probable random variable. Therefore, if $(y_1, \ldots, y_d)$ are independent random variables, then we have:

$$\ln L\,(x \mid y_1, \ldots, y_d) = \sum_{i=1}^{d} \ln\, L\,(x \mid y_i) \qquad \ldots\ldots\ldots (2.1)$$

[3] expresses log-likelihoods as a function of hyperbolic-tangents. Suppose that $(x_1, \ldots, x_l)$ are binary random variables and $(y_1, \ldots, y_l)$ are random variables. Denote addition over F_2 by ‡. We proceed to calculate $\ln L(x_1 \ddagger \cdots \ddagger x_l \mid y_1, \ldots, y_l)$ as follows.

Note that if:

$$p = (2 \Pr[x_1 = 0 \mid y_1] - 1), \text{ and} \qquad \ldots\ldots\ldots (2.2)$$

$$q = (2 \Pr[x_2 = 0 \mid y_2] - 1) \qquad \ldots\ldots\ldots (2.3), \text{ then,} \qquad \ldots\ldots\ldots (2.3)$$

$$2 \Pr[x_1 \ddagger x_2 = 0 \mid y_1, y_2] - 1) = pq \qquad \ldots\ldots\ldots (2.4)$$

Therefore,

$$2\Pr[x_1 \ddagger \cdots \ddagger x_l = 0 \mid y_1, \ldots\ldots, y_l] - 1)$$

$$= \prod_{i=1}^{l} (2\Pr[x_i = 0 \mid y_i] - 1) \quad \ldots\ldots\ldots (2.5)$$

Furthermore, since:

$$\Pr[x_i = 0 \mid y_i] = L(x_i \mid y_i) / (1 + L(x_i \mid y_i)) \quad \ldots\ldots\ldots (2.6)$$

$$\Longrightarrow 2\Pr[x_i = 0 \mid y_i] - 1 = {(L-1)}/{(L+1)} = \tanh(l/2) \quad \ldots\ldots\ldots (2.7)$$

where $L = L(x_i \mid y_i)$ and $l = \ln L$. Therefore, we obtain:

$$\ln L(x_1 \ddagger \cdots \ddagger x_l \mid y_1, \ldots, y_l) = \ln \frac{1 + (\prod_{i=1}^{l} \tanh(l_i/2))}{1 - (\prod_{i=1}^{l} \tanh(l_i/2))} \quad \ldots\ldots\ldots (2.8)$$

Where $l_i = \ln L(x_i \mid y_i)$.

For the derivation of the BP algorithm for LDPC codes [3], let $m_{vc}^{(l)}$ be the message passed from message node v to check node c at the l^{th} round of the algorithm. Similarly, define $m_{cv}^{(l)}$. At round 0, $m_{vc}^{(0)}$ is the log-likelihood of the message node v conditioned on its observed value, which is independent of c. Let us denote this value by m_v. Then the update equations for the messages under belief-propagation can be described as:

$$m_{vc}^{(l)} = \begin{cases} m_v, & if\ l = 0 \\ m_v + \sum_{c' \epsilon\, C_v \setminus \{c\}} m_{c'v}^{l-1}, & if\ l \geq 1 \end{cases} \quad \ldots\ldots\ldots (2.9)$$

$$m_{cv}^{(l)} = \ln \frac{1 + \prod_{v' \epsilon\, V_c \setminus \{v\}} \tanh(m_{v'c}^{(l)} / 2)}{1 - \prod_{v' \epsilon\, V_c \setminus \{v\}} \tanh(m_{v'c}^{(l)} / 2)} \quad \ldots\ldots\ldots (2.10)$$

Where, C_v is the set of check nodes incident to message node v, and V_c is the set of message nodes incident to check node c.

The computations at the check nodes can be simplified further by performing them in the log-domain, in the form of log-likelihood ratios. Shokorollahi [4] suggests since the value of $\tan h(x)$ can be negative, we need to keep track of its sign separately. As per his method, let γ be a map from the real numbers $[-\infty, \infty]$ to ($F_2 * [0, \infty]$) defined by:

$$\gamma\,(x) = \left(sgn(x), -\ln \tanh\left(\frac{|x|}{2}\right)\right) \quad \ldots\ldots\ldots (2.11)$$

So in this case, $sgn(x)$ can be set as:

$$sgn(x) = 1, if\ x \geq 1 \text{ and } sgn(x) = 0,\ otherwise)$$

It is clear that γ is bijective, so there exists an inverse function γ^{-1}. Moreover,

$$\gamma(xy) = \gamma(x) + \gamma(y) \quad \ldots\ldots\ldots (2.12)$$

Where, addition is component-wise in F_2 and in $[0, \infty]$. Then it can be shown that:

$$m_{cv}^{(l)} = \gamma^{-1}\left(\sum_{v' \in V_c \setminus \{v\}} \gamma\left(m_{v'c}^{(l-1)}\right)\right) \quad \ldots\ldots\ldots (2.13)$$

In practice, belief propagation may be executed for a pre-specified maximum number of rounds or until the passed likelihoods are close to certainty under pre-determined thresholds, whichever is achieved first. Some authors use the term C*ertain Likelihood* to denote beliefs, where a *certain* likelihood is a likelihood in which $\ln L\ (x \mid y)$ is either ∞ or $-\infty$. If it is ∞, then $\Pr[\, x = 0 \mid y\,] = 1$, and if it is $-\infty$, then $\Pr[\, x = 1 \mid y\,] = 1$.

To estimate the running time of the belief propagation algorithm, it is important to note that the algorithm traverses the edges in the graph and the graph is sparse. So, the number of edges traversed in each round of the algorithm is very small. Moreover, if the algorithm runs for a pre-specified constant number of times, then each edge is traversed a constant number of times, and the algorithm uses a number of operations that is linear in the number of message nodes. So, the running time is directly proportional to the block length of the code. Another important note about belief propagation is that the algorithm itself is entirely independent of the channel used, though the messages passed during the algorithm are completely dependent on the channel. It can be noticed that belief propagation is in general less powerful than maximum likelihood decoding. [15]

2.1.3- LDPC Decoding on BEC:

Belief propagation's application to LDPC codes over the BEC with erasure probability σ constitutes a very typical example of the algorithm. Because of the binary nature of the

messages, belief propagation on the erasure channel can be described much easier in the following:

1- **Initialization:** This step constitutes the initialization of the values of all the check nodes to zero.
2- **Direct Recovery:** This step is performed for all message nodes v. In this step, if the node is received, then add its value to the values of all adjacent check nodes and remove v together with all edges originating from it from the graph.
3- **Substitution Recovery:** For check nodes, if there is a check node c of degree one, substitute its value into the value of its unique neighbor among the message nodes, add that value into the values of all adjacent check nodes and remove the message nodes and all edges originating from it from the graph.

This algorithm was first proposed in [12]. It is clear that the number of operations that this algorithm performs is proportional to the number of edges in the graph. So for sparse graphs the algorithm runs in time linear to the block length of the code.

2.1.4- Asymptotic Analysis of Belief Propagation and Density Evolution:

The messages passed at each round of the belief propagation algorithm are random variables. The update equation correctly calculates the corresponding log-likelihood based on the observations, if the incoming messages are statistically independent at every round in the algorithm. Many authors have questioned the validity of this assumption especially when the number of iterations is large. In fact, the independence

assumption is true for the first l rounds of the algorithm only if the neighborhood of a message node up to depth l is a tree. [12]

Belief propagation can be analyzed using a combination of tools from combinatorics and probability theory. The first analysis for a special type of belief propagation and its application to hard decision decoding of LDPC codes appeared in [4]. The analysis was vastly generalized in [8] to belief propagation over a hefty class of channels.

The analysis states that for fixed l in random bipartite graphs, if n and r are large enough then the neighborhood of depth l of most of the message nodes is a tree. Therefore, the belief propagation algorithm on these nodes correctly computes the likelihood of the node for l rounds. Let us call these nodes the 'good' nodes following [11]. Then the expected behavior of belief propagation is calculated by analyzing the algorithm on the tree, and a martingale is used to show that the actual behavior of the algorithm is mostly concentrated around its expectation since the conditional expected value of an observation at some time t, given all the observations up to some earlier time s, is equal to the observation at that earlier time s.

So it has been proved that a heuristic analysis of belief propagation on trees correctly depicts the actual behavior on the full graph for a fixed number of iterations, using the martingale arguments and the tree assumption, which holds for large graphs and a fixed iteration number l [3]. Furthermore, the behavior of belief propagation can be used to calculate the probability of error among the good message nodes in the graph. This

shows that the error probability of the good message nodes in the graph can be made arbitrarily small for appropriate degree distributions.

Since we assume the channel to in 'good' state most of the times, the "bad" message nodes will contribute only a sub-constant term to the error probability and their effect will disappear asymptotically since their fraction is smaller than a constant. This means that they are not relevant for an asymptotic analysis.

It should be noted that there is recursion for the density function of the messages passed along the edges in the analysis of the expected behavior of belief propagation on trees. So asymptotically speaking, the general machinery shows that the actual density of the messages passed is very close to the expected density. Thus, tracking the expected density during the iterations gives a very good picture of the actual behavior of the algorithm. This method is called *Density Evolution*. We extensively used density evolution to find thresholds of our puncturing patterns.

2.1.5- Degree Distributions for Nodes in LDPC Code:

An LDPC code is called low-density code since it's a linear block code for which the parity check matrix H is sparse and has a low density of 1's. A regular LDPC code is a linear block code whose parity check matrix H contains exactly w_c 1's in each column and exactly $\omega_r = \omega_c\ (n/m)$ 1's in each row, where;

n = Number of bits in the code-word

k = Number of bits in data to be coded

$$m = n - k$$

$$w_c \ll m$$

The code rate $R = k/n$ is related to these parameters as, $R = (1 - \frac{w_c}{w_r})$, assuming $\mathbf{H}$ is a full rank matrix. If $\mathbf{H}$ is low-density but number of 1's in each column or row is not constant, then the code is an irregular LDPC code. For irregular LDPC codes, the parameters w_c and w_r are function of the column and row numbers and are independent of each other.

It is usual in the literature (following [11]) to specify the v-node and c-node *Degree Distribution Polynomials,* denoted by $\lambda(x)$ and $\rho(x)$ respectively. In the polynomial,

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1} \quad \ldots\ldots\ldots (2.14)$$

λ_d denotes the fraction of all edges connected to a degree-d v-node and d_v denotes the maximum v-node degree. Similarly in the polynomial:

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1} \quad \ldots\ldots\ldots (2.15)$$

ρ_d denotes the fraction of all edges connected to a degree- d v-node and d_c denotes the maximum c-node degree.

A typical LDPC code can be shown as in Figure II-I. Here, the number of left (message) nodes is 10 and the number of right (check) nodes is 5.

Maximum *v*-node Degree, d_v = 4

Maximum c-node Degree, d_c = 8

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1} = 0.2\,x^2 + 0.4\,x^4 + 0.2\,x^5 + 0.2\,x^7$$

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1} = 0.6\,x^1 + 0.2\,x^2 + 0.2\,x^3$$

The parity check matrix representation of above code be:

$$\begin{bmatrix} 1&1&1&0&1&0&0&1&0&0 \\ 0&1&0&1&0&1&1&0&1&1 \\ 1&1&0&1&1&1&0&1&1&1 \\ 0&0&1&0&1&0&0&1&0&0 \\ 0&1&0&0&1&0&1&1&0&1 \end{bmatrix}$$

Please note that the degree distribution equations specify an ensemble of code, for which following graph representation is showing a particular code.

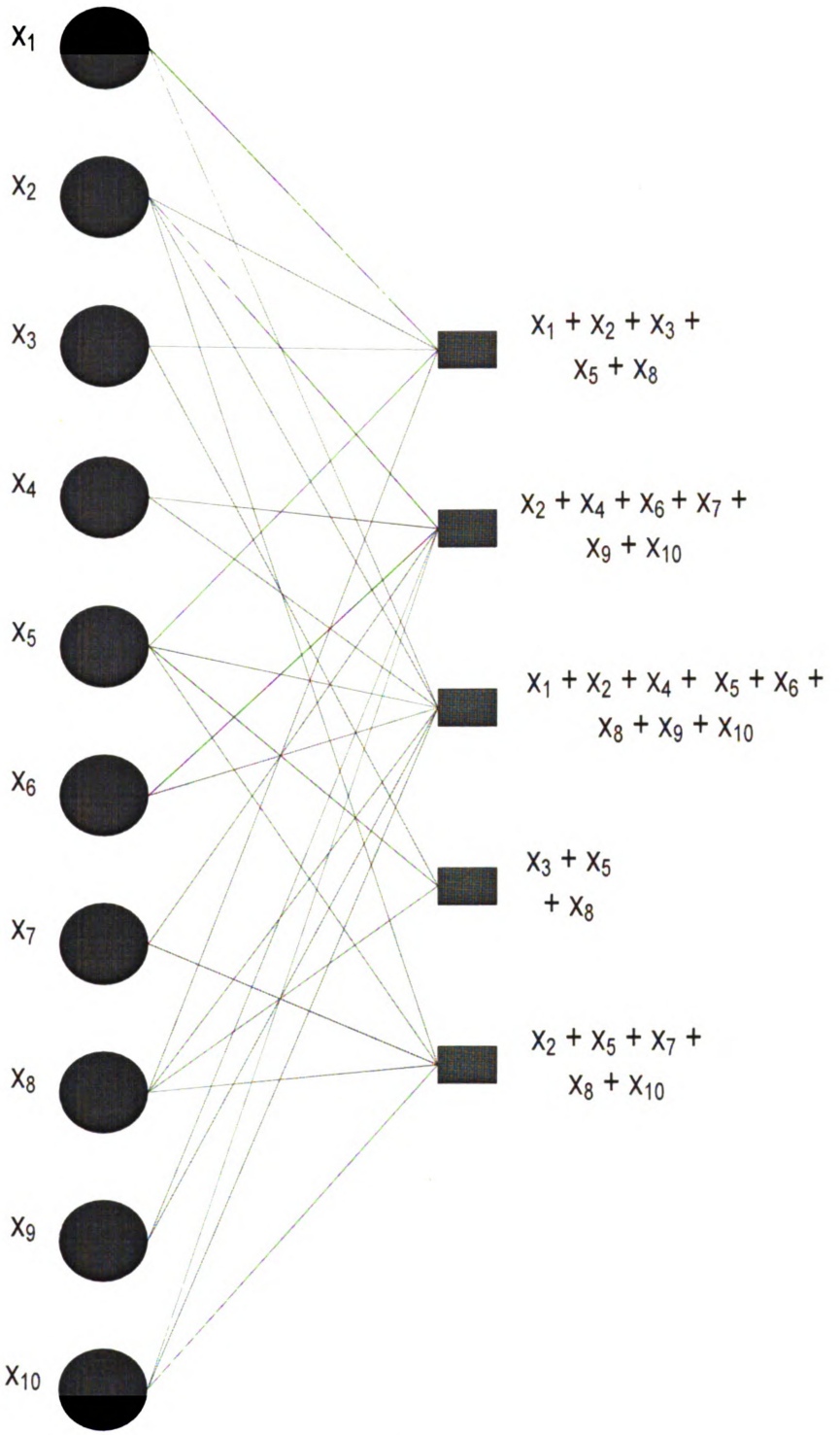

Figure II-I: Example of an LDPC code

2.2 Puncturing:

Puncturing is the process of removing some of the parity bits after encoding the data with an error-correction code and it is one of the basic techniques to modify the code rate. This has the same effect as encoding with an error-correction code with lesser redundancy, thus puncturing leads to a higher code rate.

A code C is characterized by three fundamental parameters; its length n, its dimension k, and its redundancy $x = n - k$. The code rate r can be defined as:

$$r = {}^{k}/_{n}; k < n \quad \ldots\ldots\ldots (2.16)$$

While doing puncturing, we decrease n, for a fixed k and thus increase the code rate. In this way, the code dimension remains fixed but its length and redundancy is varied as some redundancy symbols are deleted. Puncturing may cause the minimum distance to decrease because to puncture the code, we delete columns from its generator matrix.

Suppose we have a codeword $C_i(K_i, X_i)$, where, K_i data symbols are coded using X_i parity symbols. The receiver attempts to retrieve K_i data symbols by utilizing X_i parity symbols embedded in C_i. Depending on the decoding algorithm, the receiver can correct up to a certain threshold of error directly proportional to the number of parity symbols embedded in the message. For instance, for an error correcting code with error correction capability α, for X_i parity symbols, the receiver is capable of correcting up to $(\alpha * X_i)$ errors out of $|C_i|$ symbols in the message. Here α measures the expected error-correcting capability of a particular rate decoder. For example, the error-

correcting capability of Reed-Solomon codes is half as many as redundant symbols (i.e., $= 0.5$). The parameter α, thus, provides and upper bound on the error correction capability of a decoder and it is approximated by having thresholds over cumulative density function (CDF) of error distribution. With puncturing, we decrease the number of X_i parity symbols to $X_i^{'}$, assuming the receiver is capable of correcting up to $(\alpha * X_i^{'})$ errors out of $| C_i^{'} |$ symbols in the message. In this way, code rate is increased and since the same decoder can be used regardless of how many bits have been punctured, puncturing considerably increases the flexibility of the system without significantly increasing its complexity.

2.3- Residual Errors:

Traditional network protocols like IEEE 802.11 ARQ standard add some sort of forward error correction at PHY/MAC layer, such as Reed-Solomon codes etc., to recover from errors especially given the high data rates. This design strategy naturally prevents PHY/MAC layer to pass faulty packets to higher layers and these schemes drop packets and request retransmission even if a single bit is in error in the data passed from PHY/MAC layer to the higher layers. This leads to the wastage of *useful* data in the packet, the data which was received correctly. The errors which are not corrected by the physical layer are called *Residual Errors,* and residual errors result in link layer packet drops. This can be shown as figure-II-II below:

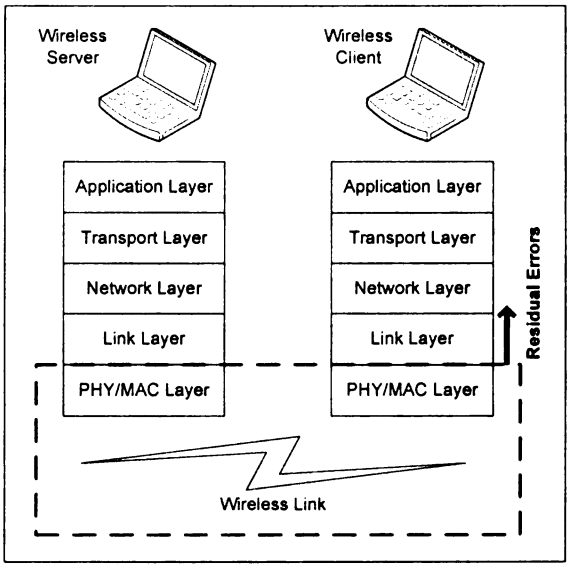

Figure II-II: Residual Errors

We define throughput over a transmission channel as:

$$throughput = \#\ of\ bits\ received\ correct\ /total\ \#\ of\ bits \quad \ldots\ldots\ldots (2.17)$$

And we define *goodput* over a transmission channel as:

$$goodput = \#\ of\ correct\ bits\ containing\ actual\ data/total\ \#\ of\ bits \quad \ldots\ldots (2.18)$$

So for a session, goodput becomes ratio of total number of bits that contained actual data and are received correctly to the total number of bits transmitted for that session. In this way, the goodput parameter excludes parity bits from calculations of throughput values. We measure the values of both, throughput and goodput, at link layer as shown in figure II-III. From figure II-III, we can see that link layer throughput corresponds to the number of bits that are getting relayed to the network layer, that is, ratio of the number of bits received by the link layer to the number of bits received by the network layer.

Similarly link layer goodput corresponds to the number of bits containing actual data that are getting relayed to the network layer, that is, ratio of the number of bits containing actual data received by the link layer to the number of bits received by the network layer.

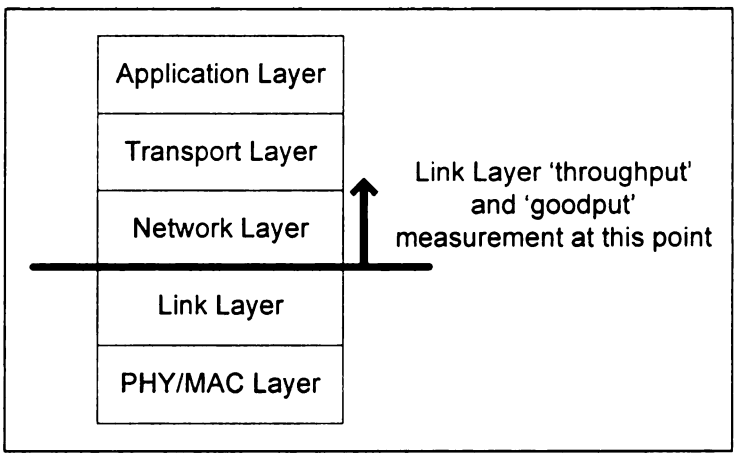

Figure II-III: Throughput and goodput measurement at link layer of the protocol stack

2.4- Channel Characterization:

Most common wireless communication channels are of two types: 1) Binary Erasure Channel (BEC); 2) Binary Symmetric Channel (BSC). In both cases the input alphabet is binary, and the elements of the input alphabet are called bits. These channels can be described as:

1) Binary Erasure Channel (BEC): A binary erasure channel with erasure probability σ is a channel with binary input, ternary output, and probability of erasure σ. Let X be the transmitted random variable with alphabet $\{0, 1\}$ and Y be the received variable with alphabet $\{0, 1, e\}$, where e is the erasure

symbol. Then, the channel is characterized by the following conditional probabilities:

a. $\Pr(Y = 0 \mid X = 0) = 1 - \sigma$

b. $\Pr(Y = e \mid X = 0) = \sigma$

c. $\Pr(Y = 1 \mid X = 0) = 0$

d. $\Pr(Y = 0 \mid X = 1) = 0$

e. $\Pr(Y = 1 \mid X = 1) = 1 - \sigma$

So, in the case of the binary erasure channel the output alphabet consists of $(0, 1)$, and an additional element denoted *e* and called *Erasure*. Each bit is either transmitted correctly (with probability $1 - \sigma$), or it is erased (with probability σ). The capacity of this channel is:

$$C_{BEC} = 1 - \sigma \quad \ldots\ldots\ldots (2.19)$$

2) Binary Symmetric Channel (BSC): A binary symmetric channel with cross-over probability denoted by ε is a channel with binary input and binary output and probability of error ε. So if X is the transmitted random variable and Y the received variable, then the channel is characterized by the conditional probabilities as below:

a. $\Pr(Y = 0 \mid X = 0) = 1 - \varepsilon$

b. $\Pr(Y = 0 \mid X = 1) = \varepsilon$

c. $\Pr(Y = 1 \mid X = 0) = \varepsilon$

d. $\Pr(Y = 1 \mid X = 1) = 1 - \varepsilon$

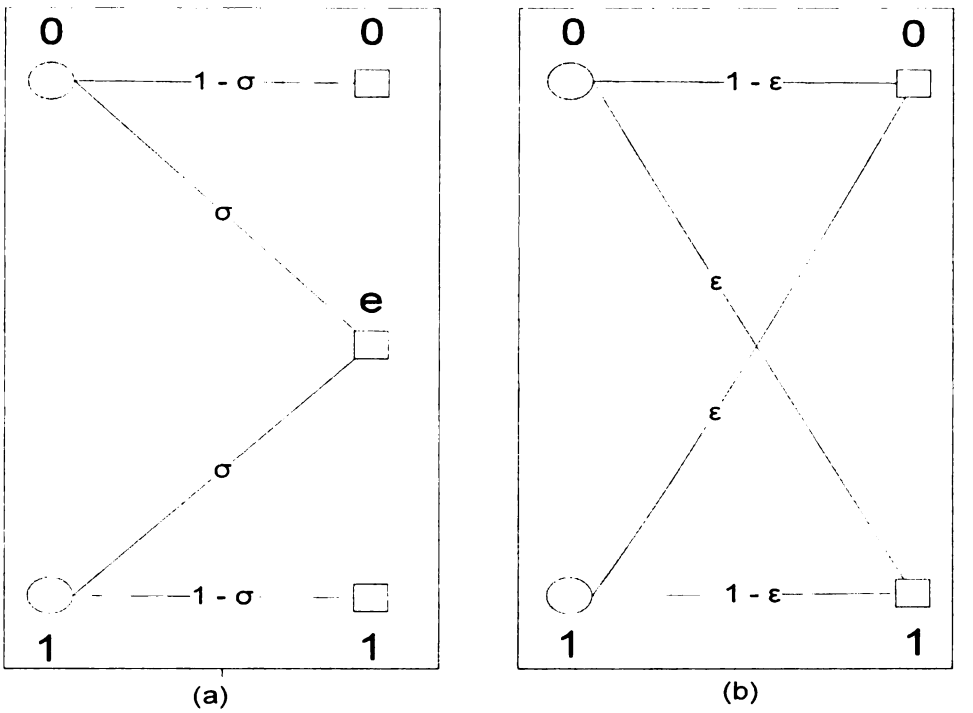

Figure II-IV: Two examples of channels: (a) The Binary Erasure Channel (BEC) with erasure probability σ , and (b) The Binary Symmetric Channel (BSC) with error probability ε

So, In the case of the BSC both the input and the output alphabet is F_2. Each bit is either transmitted correctly with probability $(1-\varepsilon)$, or it is transitioned to other with probability ε. The capacity of the channel is $1 - H(\varepsilon)$, where $H(\varepsilon)$ is the *Binary Entropy Function*:

$$C_{BSC} = \left(1 + \varepsilon log_2(\varepsilon) + (1-\varepsilon)log_2(1-\varepsilon)\right) \quad \ldots\ldots\ldots (2.20)$$

Maximum likelihood decoding for this channel is equivalent to finding, for a given vector of length n over F_2, a codeword that has the smallest Hamming distance from the received word.

Chapter 3 - Code Design and Working of PRAISE

This chapter focuses on the code design aspects that we took into account to propose rate-adaptive codes using localized puncturing. In the beginning, we provide details about density evolution tools (LOPT and LODE) that we used to,

1) identify appropriate degree distributions with maximum noise variances for given allowed degrees under specified parameters, and

2) calculate noise thresholds for the degree distributions that we actually used to design our codes under practical constraints.

In the latter part, we continue with the discussion of designing puncturing fractions for rate-adaptability of channel codes in time-varying channel conditions.

3.1- Density Evolution Tools:

LDPC Online Optimisation Tool (LOPT) is an online optimization tool and LDPC Online Density Evolution (LODE) is an online density evolution calculator developed by Signal Processing and Microelectronics Lab, School of Electrical Engineering and Computer Science at University of Newcastle. We extensively used LOPT to find 'good' degree distributions for LPDC code generation. This gave us an idea about ratios for allowed degrees for a given code rate. Moreover, we also used LODE to use density evolution and find noise thresholds for the code with allowed degrees and given degree distributions.

3.1.1- LDPC Online Optimisation Tool:

LOPT is an online optimization tool, which uses density evolution to find the ensemble which returns the best possible threshold for a given code-rate and allowed degrees. To recall degree distributions as explained in Chapter 2, the variable (v-node) and check (c-node) d*egree* distribution polynomials, denoted by $\lambda(x)$ and $\rho(x)$ respectively. In the polynomial,

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1}$$

λ_d denotes the fraction of all edges connected to a degree-d v-node and d_v denotes the maximum v-node degree. Similarly in the polynomial:

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1}$$

ρ_d denotes the fraction of all edges connected to a degree- d v-node and d_c denotes the maximum c-node degree.

In this way, code designer can look for the specified degree distributions to get the best possible performance for a specified code rate and allowed degrees.

We extensively used LOPT to find the degree distributions and noise thresholds in code designing process for a particular code rate. Since, it was not always possible to confine to the degree distributions requirements specified by LOPT due to practical constraints; we tried to design codes with degree distributions as near as possible to the suggested

ones. Later on, we used LODE to find noise thresholds corresponding to actually used degree distributions for code designing.

Parameters for LOPT	
Code Rate	0.500000
Min LLR Value	-20.000000 dB
Max LLR Value	20.000000 dB
Number of Iterations	1000
Minimum BER	1.000000e-10
λ Degrees	2, 3, 4
ρ Degrees	4, 5, 6
Results for LOPT	
$\lambda(x)$	0.352288, 0.127617, 0.520095
$\rho(x)$	0.059358, 0.082206, 0.858436
Noise Variance	0.9050000 ≤ Var ≤ 0.910000

Table III-I: LOPT Parameters and Results

Theoretically speaking, the Probability Density Functions (PDFs) of density evolution are continuous and defined for all possible log-likelihood ratios (LLRs). But, this is not possible practically in the implementation process. So to avoid this, both LOPT and LODE limit the LLRs considered to those between Min LLR and Max LLR and chooses Number of LLR points equally spaced samples between Min LLR and Max LLR. We chose Min LLR, Max LLR and Number of LLR points as -20 dB, 20 dB and 511 respectively.

Furthermore, in theory the sum-product decoder is allowed to run for an infinite number of iterations and we expect the bit error rate to be zero for a successful decoding run, it is again not possible in practice. Both LOPT and LODE limit the simulations to the Number of iterations specified and consider BERs below Minimum BER to be equivalent to a BER of zero. We restricted the number of iterations to 1000 and Minimum BER to 1.0E-10, whichever is achieved first. A sample use of LOPT is as shown in Table III-I

3.1.2- LDPC Online Density Evolution:

LODE is an online density evolution calculator, and can be used to find the threshold of an LDPC ensemble. To use LODE, we specified the degree distributions corresponding to ρ's and λ's for our ensemble, where by definition, summation of $\lambda(x)$ and $\rho(x)$ over all i equals 1. We extensively used LODE to find BER vs. Thresholds for our LDPC ensembles.

Here again, due to practical constraints as in case of LOPT, we needed to specify Min LLR, Max LLR, Number of LLR points, Number of iterations, and Minimum BER and chose the same values as in case of LOPT. Furthermore, here we were able to select the output format and there are two output formats available.

1- Threshold Option

2- Eb/No Option

In our work, we used the threshold option, which searches between Minimum (dB) and Maximum (dB) to find the noise threshold of the specified ensemble. In this

case, the result is accurate to within Minimum difference (dB) and the output is the threshold, given as both the signal-to-noise ratio in dB and the corresponding noise variance (Var).

Parameters for Threshold search in LODE	
Code Rate	0.460197
Min LLR Value	-20.000000 dB
Max LLR Value	20.000000 dB
Number of Iterations	1000
Minimum BER	1.000000e-10
λ Degrees	2, 3, 4
$\lambda(x)$	0.2939, 0.0923, 0.6141
ρ Degrees	4, 5, 6
$\rho(x)$	0.0921, 0.1323, 0.7756
Threshold Min (dB)	0
Threshold Max (dB)	10
Minimum Difference	1e-5
Results for Threshold search in LODE	
Eb/No Threshold (dB)	0.731077
Var	0.958207

Table III-II: LODE Parameters and Results for Threshold Search

3.2- Designing Puncturing Fractions:

An LDPC code is called low-density code because it's a linear block code for which the parity check matrix H is sparse and has a low density of 1's. A regular LDPC code is a linear block code whose parity check matrix H contains exactly w_c 1's in each column and exactly $\omega_r = \omega_c (n/m)$ 1's in each row, where:

n = Number of bits in the code-word

k = Number of bits in data to be coded

$$m = n - k$$

$$w_c \ll m$$

The code rate in case can be defined as:

$$r = \frac{k}{n}; \quad k < n \qquad \ldots\ldots\ldots (4.1)$$

If H is a full rank matrix, then the code rate r is related to these parameters as,

$$R = (1 - \frac{w_c}{w_r})$$

If H *is low-density but number of 1's in each column or row is not constant then the code is called an Irregular LDPC Code*. For irregular LDPC codes, the parameters w_c and w_r are function of the column and row numbers and are independent of each other. We consider only regular ensembles of LDPC codes in our work.

In Chapter 2, we denote the variable-node (v-node) and check-node (c-node) degree distribution polynomials by $\lambda(x)$ and $\rho(x)$ respectively. In the polynomial,

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1}$$

λ_d denotes the fraction of all edges connected to a degree-d v-node and d_v denotes the maximum v-node degree. Similarly in the polynomial:

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1}$$

ρ_d denotes the fraction of all edges connected to a degree-d v-node and d_c denotes the maximum c-node degree.

Following these notations, the rate of the code in terms of its degree distributions pair become:

$$r(\lambda, \rho) = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} \qquad \ldots\ldots\ldots (3.1)$$

Following [36], the total puncturing fraction $p^{(0)}$ can be defined as,

$$p^{(0)} = \frac{the\ number\ of\ punctured\ variable\ nodes}{the\ total\ number\ of\ variable\ nodes} \qquad \ldots\ldots\ldots (1.5)$$

In Chapter 4, we develop the channel model and introduce the notation of *Transmission Intervals* $[\mathrm{T}_i;\ for\ i = 1, 2, \ldots, \infty]$, where the index i refers to the i^{th} time slot. We also

assume that packets are transmitted over the channel under consideration during discrete time slots. For i^{th} transmission interval (T_i), we define $p^{(i)}$ to be the total puncturing fraction in the i^{th} interval,

$$p^{(i)} = \frac{\textit{the \# of punctured variable nodes in the } i^{th} \text{ transmission interval}}{\textit{the total \# of variable nodes}} \quad \ldots (3.2)$$

Puncturing increases code rate because we decrease n for fixed k bits in doing puncturing operation. To recall the operating procedure, a 'mother' code is designed with rate, $r(\lambda, \rho)$, with highest redundancy. This code, with the largest number of parity bits, is used in the worst channel condition. We do puncturing as determined by the (σ, ε) pair for channel characterization, as discussed in Chapter 4, to transmit a 'daughter' code with error-correction capability α_d and code rate $r(\lambda, \rho)^{(\sim)}$. An important aspect to mention here is that the (σ, ε) pair value at the i^{th} time interval is used to design puncturing fraction $p^{(i+1)}$ and thus, the code-rate for the packet to be transmitted at $(i+1)^{th}$ time interval.

The rate of 'daughter' code in the i^{th} interval, $r(\lambda, \rho)^{(\sim i)}$, is related to the rate of base/'mother' code as:

$$r(\lambda, \rho)^{(\sim i)} = \frac{r(\lambda, \rho)}{1 - p^{(i)}} \qquad \ldots\ldots\ldots (3.3)$$

Several recent works tried to find bounds on code rates for reliable communication. Examples of such works include ACE [34] and RASE [40], which developed a framework

for reliable and stable communication with guarantees on sustainable flows. ACE defines stability as *"System is stable when higher layers are neither starved for information packets nor is there a glut of packets leading to buffer overflow"*. If ε_i is BSC channel's error (cross-over) probability and α_m is the error-correction capability of a code, then ACE determines the code rate for reliable and stable communication as:

$$R = 1 - \frac{\varepsilon_i}{\alpha_m} \qquad \ldots\ldots\ldots (3.4)$$

This means that if a channel is in 'good' state, we can increase the code rate and still have a reliable and stable communication with the same guarantees on sustainable flows. So PRAISE's working in terms of code-designing can be summarized as follows:

At the encoder end, we divide n-bit codeword into c chunks or partitions in a manner pre-determined by encoder and decoder. During T_i, a sender transmits a message which is represented by the tuple $M_i = (C_i(k_i, X_i); Y_i)$, where k_i represents the number of data symbols which are not being retransmitted in the i^{th} transmission interval. In each T_i, a transmitter encodes k_i with parity symbols X_i creating a codeword $C_i(k_i, X_i)$ with c chunks. We refer to these parity symbols as *Type-I Parity* following [34]. The receiver utilizes X_i to decode C_i and in case of successful decoding, k_i data symbols are passed up to the higher layers. In the case of error-correction (and thus, decoding) failure, the receiver stores C_i in its buffer, identifies l chunks of the lowest reliability out of c chunks with the help of side-information received from PHY/MAC layer, and requests additional parity data for those l partitions. The

transmitter also sends additional parity symbols for l chunks of the lowest reliability, denoted by Y_i. We call these *Type-II Parity Symbols* following [34]. The receiver utilizes Y_i symbols to recover old corrupted code-words accumulated in the buffer (e.g., $C_j; j = 1, 2, \ldots, i-1$).

The transmitted codeword after reception at the decoder end is called the Received Word. Received word has both, errors and erasures in it along with, if any, correctly received data-bits. The decoder identifies the location of erasures in received word caused by puncturing operation at the encoder end because it can determine the code-rate of the received word. Moreover, since we used LDPC codes, whenever a received word is decoded correctly, we are able to figure out exactly the number and position of errors in the received word at the decoder end. The decoder uses the BEC erasure probability σ_i as determined by the puncturing fraction, and BSC error probability ε_i as introduced by wireless channel to figure out the (σ, ε) pair value of the channel at the i^{th} interval. For details on the channel model, please refer to Chapter 4 of the thesis.

The PRAISE's receiver has a finite buffer which can accommodate up to b corrupted messages waiting additional parity data from encoder. If a newly corrupted packet finds all the spaces in the buffer occupied, it does not enter the buffer and is dropped. The status of the receiver is reported to the transmitter via certain flags in an acknowledgment message which are called *Buffer Status Flags*. Specifically, b flags are encapsulated in every acknowledgement packet. Let buffer status flags in ACK_i be represented as:

$$[F_i[k];\ k = 1, 2, \ldots, b], and$$

$$[F_i[k] = [\ 1 + f], bits$$

$$f = \left\lceil log_2 \binom{c}{l} \right\rceil$$

Please note that we need $f = \left\lceil log_2 \binom{c}{l} \right\rceil$ feedback bits to identify l out of c chunks of lowest reliability and the additional 1 bit is required to flag the status of the k^{th} space. In this way, each buffer status flag is associated with a particular packet-space in the buffer and represents the status of that space, along with the number of the lowest reliability l chunks, to request additional parity data for. In addition, the receiver estimate of channel condition, determined by (σ, ε) pair in T_i is also encapsulated in acknowledgment message. The ACK_i value for the buffer status flags is transmitted back to encode a new packet with:

1- puncturing fraction $p^{(i+1)}$ and thus, the code-rate for the packet to be transmitted at $(i+1)^{th}$ time interval. For a 'daughter' code with error-correction capability α_d, we determine puncturing fractions as:

$$p^{(i+1)} = 1 - \frac{r(\lambda, \rho)}{1 - \frac{\varepsilon_i}{\alpha_d}} \qquad \ldots\ldots\ldots (3.5)$$

2- additional parity data for l chunks of the lowest reliability for the corrupted code-words residing in the buffer at the receiver in the i^{th} transmission interval.

This process continues till whole data is successfully received and decoded.

Chapter 4 - The Channel Model

This chapter provides the details of the channel model that we used to develop the proposed scheme for rate-adaptability of wireless protocols in time-varying channel conditions. *A channel model describes the process under which errors are introduced in a transmitted packet over a wireless link.* For modeling purposes, we assume that packets are transmitted over the channel under consideration during discrete time slots which we refer to as *Transmission Intervals,* T. We denote transmission intervals by $[\mathrm{T}_i;\ for\ i = 1, 2, \ldots, \infty]$, where the index i refers to the i^{th} time slot. To proceed with our channel model, we divide our discussion in two levels. The modeling begins with developing channel model in a particular transmission interval and then continues with a complete channel model based on all transmission intervals.

During the i^{th} transmission interval, a packet is transmitted over cascaded BEC and BSC channels. Here, the BEC channel with erasure probability σ models the puncturing operation to puncture the input codeword. The input codeword to the BEC channel is encoded with the code rate for 'mother' code, $r(\lambda, \rho)$, and the output is the 'daughter' code with code rate $r(\lambda, \rho)^{(\sim)}$. Hence we model the puncturing fraction $p^{(o)}$ by BEC erasure probability σ, and the puncturing operation by a BEC channel. Please refer to Chapter-3 for more details on code rates and their design aspects.

In our channel model, the BSC with cross-over probability ε models the wireless channel transmitting the punctured codeword. These two channels are parameterized by the following:

σ: BEC erasure probability (as determined by the puncturing fraction)

ε: BSC error probability

Hence, the (σ, ε) pair characterizes the channel during each transmission interval T. *The transmitted codeword after reception at the decoder end is called the Received Word.* Received word has both, errors and erasures in it along with, if any, correctly received data-bits. The decoder identifies the location of erasures in received word caused by puncturing operation at the encoder end because it can determine the code-rate of the received word. Moreover, since we used LDPC codes, whenever a received word is decoded correctly, we are able to figure out exactly the number and position of errors in the received word at the decoder end. In this way, the decoder uses the $(\sigma_i, \varepsilon_i)$ pair to figure out the channel state at the i^{th} interval.

It is important to note that puncturing causes erasures in a transmitted packet. Strictly speaking, it's not a part of the channel over which we are transmitting a packet as it is performed at the encoder end before transmission. Still we can assume it is a part of the channel for modeling purposes as at the decoder end, because decoder finds both, errors and some erasures in the received word. So for the i^{th} transmission interval, the channel model can be as shown in the figure IV-I below:

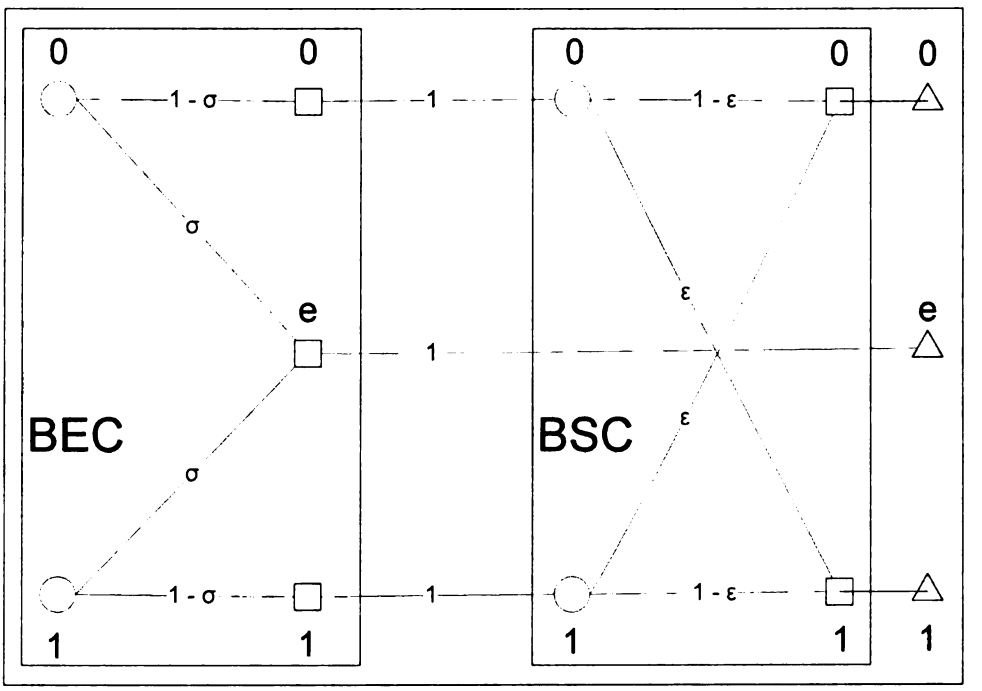

Figure IV-I: Cascaded BEC and BSC Channels with erasure and error probabilities σ and ε respectively.

A discrete random process with the property that the next state depends only on the current state is called as ***Markov Chain***. The Markov property states that the conditional probability distribution for the system at the next step, and in fact at all future steps, given its current state depends only on the current state of the system, and not additionally on the state of the system at previous steps:

$$Pr\,(X_{n+1} \mid X_1, X_2, \ldots, X_n) = Pr\,(X_{n+1} \mid X_n) \qquad \ldots\ldots (4.1)$$

We call the process under-discussion "*a discrete random stochastic process*" because the wireless channel, which is in a certain state characterized by its particular $(\sigma_i, \varepsilon_i)$ pair at i^{th} transmission interval, changes randomly between intervals. Moreover in our

proposed scheme to adapt rate of wireless protocols in different network conditions, the decoder transmits the (σ, ε) pair value back to the encoder via feedback and the encoder uses the $(\sigma_i, \varepsilon_i)$ pair value at the i^{th} time interval to design puncturing fraction $p^{(i+1)}$ and thus, the code-rate for the packet to be transmitted at $(i+1)^{th}$ time interval. This suggests a *Markov-based* channel characterization.

Hence, to derive a channel model for all transmission intervals, we assume that each $(\sigma_i, \varepsilon_i)$ pair value of a particular T_i is valued from a finite set $\mathbb{F}_N$ with length N i.e.

$$(\sigma_i, \varepsilon_i) \in \mathbb{F}_N;\ |\mathbb{F}_N| = N \qquad \ldots\ldots (4.2)$$

As a result, we can consider the channel model as a combination of N various cascaded BECs and BSCs with unique (σ, ε) pairs, i.e.,

$$(\sigma_m, \varepsilon_m) \neq (\sigma_n, \varepsilon_n); for\ m \neq n; m, n = 1, 2, \ldots, N \qquad \ldots\ldots (4.3)$$

In every T_i, the channel is in one of the N possible states $(S_1, S_2, \ldots, S_N)$ where each state corresponds to a particular cascade of BECs and BSCs as depicted in figure above.

The changes of state of a Markovian system are called as Transitions, and *the probabilities associated with various state-changes are called Transition Probabilities.* The set of all states and transition probabilities completely characterizes a Markov chain. By convention, we assume all possible states and transitions have been included in the definition of the processes, so there is always a next-state and the process goes on forever. In our proposed scheme, the (σ, ε) pair value is calculated at the receiver, which reports it back for training of the Markovian channel and predicting code rate for

next word to be transmitted. Since our state space is finite, the transition probability distribution can be represented by a matrix, called the *Transition Matrix*, with the $(s,t)th$ element of the matrix $P_{s,t}$ equals to:

$$p_{s,t} = Pr\,(X_{n+1} = s \mid X_n = t) \qquad \ldots\ldots (4.4)$$

For simplicity of modeling, we can assume that probability of transition between states is independent of the state index n.This makes the Markov chain a *Time-homogeneous Markov chain*, and in this way, the process is described by a single, time-independent matrix, $P_{s,t}$. For a time-homogeneous Markov chain:

$$Pr\,(X_{n+1} = s \mid X_n = t) = Pr\,(X_n = s \mid X_{n-1} = t) \qquad \ldots\ldots (4.5)$$

In such a case, the vector $л$ is called a *Stationary Distribution,* and the associated probabilities as *Steady-state Probabilities*, if the entries $л_t$ of the vector $л$ sum to 1 and it satisfies:

$$л_t = \sum_{s\,\varepsilon\,S} л_s \cdot p_{s,t} \qquad \ldots\ldots (4.6)$$

Based on the above-mentioned channel model, the wireless channel for the proposed *Side-Information Enhanced Forward Error Correction (SEFEC)* scheme is modeled by a discrete Markov chain with N possible states where each Markov state is represented by a cascade of BECs and BSCs with a particular (σ, ε) pair. We assume a homogenous and stationary Markov chain with transitional probability matrix $P_{s,t}$ and the limiting state (stationary) probabilities $(л = л_1, л_2, \ldots, л_N)$. The Markovian channel model is

trained on real channel traces by using the statistics of previous transmission intervals. This captures the effects of multipath fading and interference on the (σ, ε) pair in every transmission interval using a single aggregated model. This can be shown in figure IV-II. It's well known that the capacity of the cascaded BEC(σ) and BSC(ε) channel is $(1-\sigma).(1-H(\varepsilon))$. Using the steady state probabilities, the average channel capacity in the i^{th} transmission interval is determined as follows:

$$C_T = \sum_{i=1}^{N} \pi_i \, (1-\sigma_i)(1-H(\varepsilon_i)) \qquad \ldots\ldots (4.7)$$

The channel capacity gives an upper bound on the average (reliable) information transmission rate for the wireless channel under consideration in the i^{th} transmission interval.

We explained the *Conventional Standard (CS)* schemes like IEEE 802.11 ARQ in Chapter-I. Some prior works [35, 41, 42] have proposed cross-layer designs to achieve improvements in end-to-end bandwidth utilization and overcome performance limitations imposed by CS. We have included details of such works in Chapter-II of the thesis, with the fact that their primary drawback is that their implementation requires major modifications in higher layers. However, the analysis of the Hybrid Erasure-Error Protocols (HEEPs) [42] shows that cross-layer protocols in general can significantly improve the overall performance as measured by video quality and provide capacity improvement in many realistic scenarios. The proposed SEFEC scheme is a cross-layer

design with all advantages of a cross-layer design and does not require any modification in higher layers for extraction of side-information.

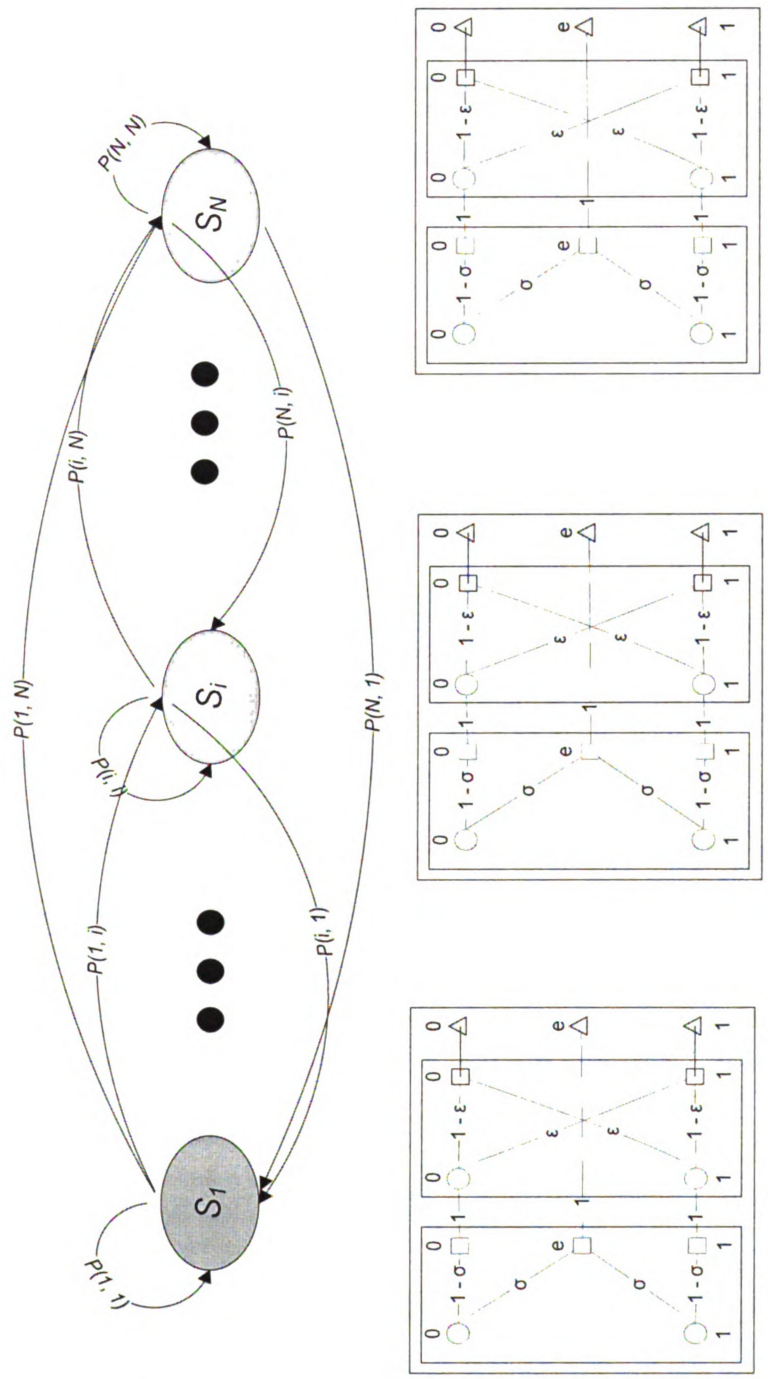

Figure IV-II: Markovian Channel Model

Chapter 5 - Experimental Setup

This chapter begins with giving a brief idea about Software-Defined Radios (SDRs) and explains GNU-USRP based implementation of a *Software Defined Radio (SDR)* to be used in our work. We continue with our motivation to use GNU-USRP platform to collect real-time channel traces and provide details about our experimental setup. In the end, we report parameters to collect data.

5.1- Software-Defined Radios:

The term *Software Defined Radio* was first used in 1991 by Joseph Mitola, who published the first paper on the topic in 1992. A software-defined radio system, or SDR, is a radio communication system where components of a typical radio communication system, like mixers, filters, amplifiers, modems (modulators/demodulators), detectors, etc., that have been typically implemented in hardware are instead implemented by means of software on a personal computer or on embedded computing devices. Hence in an SDR, software defines shape of waveforms, modulation/demodulation schemes and other parameters of transmission like packet size, data rate and bursty modes. This brings before user a whole new area of design by transforming typical hardware problems into software ones. In this way, a user is more flexible to play with these parameters by changing different software building blocks which was not possible earlier due to hardware limitations.

A very basic SDR communication system may consist of a personal computer equipped with a sound card, or other analog-to-digital (A/D) converter, preceded by some form

of RF front end. Typically major part of signal processing is handed over to the general-purpose processor, rather than being done in special-purpose hardware. Such a design produces a radio which can receive and transmit widely different radio protocols based solely on the software used.

SDRs have found significant utility for the military and cell phone services, as both of these always look to serve a wide variety of changing radio protocols demands in real time. In the long term, SDRs are expected to become the dominant technology in radio communications systems. SDRs, along with software defined antennas are the enablers of the cognitive radio computing and are being used extensively by research community and academia.

5.2- GNU Radio:

GNU Radio is an open-source, free software toolkit for learning about, building, and deploying SDR communication systems. The project started in 2001 and is now an official GNU project, released under the GPL version 3 license. Philanthropist John Gilmore initiated and has sustained GNU Radio as of now with the funding of $320,000 (US) to Eric Blossom for code creation and project management duties.

GNU Radio is a signal processing package with the goal to give ordinary software people the ability to understand the electromagnetic radio spectrum and think of clever ways to use it. So it makes possible for a user to explore many domains of cognitive computing using software modules only.

GNU Radio signal processing blocks are written in C++, while creating flow graphs and connecting signal blocks is done using Python. So, the developer is able to implement real-time SDR with high-throughput capability, in a simple-to-use and rapid-application-development environment.

As with all SDR communication systems, re-configurability is the key feature of GNU Radio. Instead of purchasing multiple expensive radios, a single more generic hardware is purchased, which feeds into powerful signal processing software and serves the purpose. Currently only a few forms of radio are implemented in GNU Radio, but one can reconfigure GNU Radio to receive any radio transmission system by understanding the math behind it.

5.3- Universal Software Radio Peripheral:

The Universal Software Radio Peripheral (USRP) is a high-speed USB-based board for making software radios, and acts as an RF front-end for an SDR. USRP is a comparatively inexpensive hardware device (under $1K) facilitating the building of an SDR. USRP has got open source drivers and software to incorporate GNU Radio. Furthermore, the USRP board schematics and the associated Verilog code are also freely available for download. It is also designed to be flexible, allowing developers to make their own daughter-boards for specific needs with regard to connectors, different frequency-bands, etc.

The USRP is developed by GNU Radio project and a team led by Matt Ettus. USRP is a digital acquisition system containing A/D and D/A converters, and support circuitry including a high-speed USB 2.0 interface. The USRP is capable of processing signals with

bandwidth up to 16 MHz. Several transmitter and receiver plug-in daughter boards are available covering various bands between 0 and 5.9 GHz. Technical details of USRP are as under:

- Four high-speed 64 MS/s A/D converters, each having a resolution of 12 bit
- Four high-speed 128 MS/s D/A converters, each having a resolution of 14 bit
- An Altera Cyclone EP1C12Q240C8 FPGA
- A Cypress EZ-USB FX2 High-speed USB 2.0 controller
- 4 extension sockets (2 TX, 2 RX) in order to connect up to 4 daughter-boards
- 64 general purpose I/O pins available through four BasicTx/BasicRx daughterboards (16 pins each)

5.4- Daughter-Boards with USRP:

Daughter-boards serve as the RF frontend for SDR. They allow the output signal to be modulated to a higher frequency and an input signal to be demodulated of its carrier. USRP comes with a variety of daughter-boards and three different types of boards exist:

1. Receivers
2. Transmitters
3. Transceivers

1- Receivers: Receivers are a type of daughter-boards that only support RX and consume only one RX port:

- BasicRX, 1 - 250 MHz Receiver, for use with external RF hardware.
- LFRX, DC - 30MHz Receiver
- TVRX, 50 MHz - 870 MHz Receiver

- DBSRX, 800 MHz - 2.4 GHz Receiver
- BURX, 300 MHz - 4 GHz Receiver

2- Transmitters: Transmitters are a type of daughter-boards that only support TX and consume one TX port:

- BasicTX, 1 - 250 MHz Transmitter, for use with external RF hardware
- LFTX, DC - 30MHz Transmitter

3- Transceivers: Transceivers are a type of daughter-boards that are both TX and RX and consume 2 ports, one for transmission and the other for reception:

- WBX0510, 50 MHz - 2.2 GHz Transceiver
- RFX400, 400 - 500 MHz Transceiver
- RFX900, 800 - 1000 MHz Transceiver
- RFX1200, 1150 - 1450 MHz Transceiver
- RFX1800, 1.5 - 2.1 GHz Transceiver
- RFX2400, 2.3 - 2.9 GHz Transceiver
- XCVR2450, Dual-band Transceiver, i.e. 2.4 - 2.5 GHz and 4.9 - 5.85 GHz

5.5- Motivation for GNU-USRP Platform:

Wireless network refers to any type of computer network that is wireless, and is implemented using remote information transmission system. Common types of wireless networks include Wireless Personal Area Networks (e.g., Bluetooth, Zigbee), Wireless Local Area Networks (e.g., Wi-Fi, Fixed Wireless Data), Wireless Metropolitan Area Networks (e.g., WiMAX), Wireless Wide Area Networks (e.g., Gaiacom Wireless Network), and Mobile devices networks (with development of smart phones, like GSM,

PCS and D-AMPS). Wireless market is having many options to establish communication among different users in various different environments.

Current wireless network devices such as USB network adaptors (e.g., Samsung WIS-09ABGN, Linksys WUSB54G), Peripheral Component Interconnect (PCI) wireless adaptor cards for computer desktops (e.g., D-Link DWL-G520, Linksys WMP54G) and notebooks (e.g., Linksys WPC54G, TP-Link TL-WN360G), wireless Ethernet bridges (e.g., TiVo AN0100 , Linksys WET54G), and Wireless Compact Flash Card Adapters for PDAs (e.g., Linksys WCF54G, Netgear MA701) are utilized to establish communication channels among different wireless users in various wireless environments. By design, these devices use IEEE802.11 standard to attain channel access, contention, and error control at the Physical (PHY) and MAC layers.

IEEE 802.11 standard has some sort of basic error correction at PHY layer, like Reed Solomon codes etc. and this design strategy naturally leads to preventing PHY layer to pass distorted packets to the higher layers. The distorted packets are dropped right away at PHY layer, wasting a lot of useful data. So we faced the major challenge of utilization and alternation of link-layer corrupted packets, containing residual errors, at the receiver to:

1- Capture and measure the behavior of an error process of a wireless channel,

2- Analyze the residual errors, at link-layer, in a wireless channel to design suitable puncturing fractions in localized manner for rate-adaptability of error control protocols,

3- Implement error control protocols at the link-layer that employ and manipulate received corrupted packets, and

4- Extract side information from MAC/PHY layers to request localized parity for subsets with potentially highest number of errors.

To cope with this issue, we need to modify the configuration of these devices to attain the distorted packets at the PHY/MAC layer. But, these off-the-shelf wireless devices are not open source products, thus modification of the source code functionality is not possible.

To overcome this problem, we use Software-Defined Radio (SDR) technology. Specifically, we use GNU-USRP platform as the communication system front-end on each sender and receiver node in our experiments. Since, GNU Radio is an open-source software development toolkit, we are able to easily reconfigure and modify its building blocks. In this way, we have complete control over the communication system configurations down to the FPGA/antenna level, and down to PHY/MAC layers and we are able to capture distorted packets received at the PHY/MAC layer and pass packets with residual errors to link layer. Furthermore, we are able to extract side information associated with each packet.

5.6- Layered Architecture:

Figure V-I shows the layered architecture of the experimental realization of the wireless communication for this thesis. At the top level, *Application Layer* resides which is designed to generate data packets and pass them down to the link-layer. It is important

to note that we are interested in point-to-point link-layer wireless communication enhanced by side-information from MAC/PHY layer, thus TCP and Network modifications are irrelevant for this work.

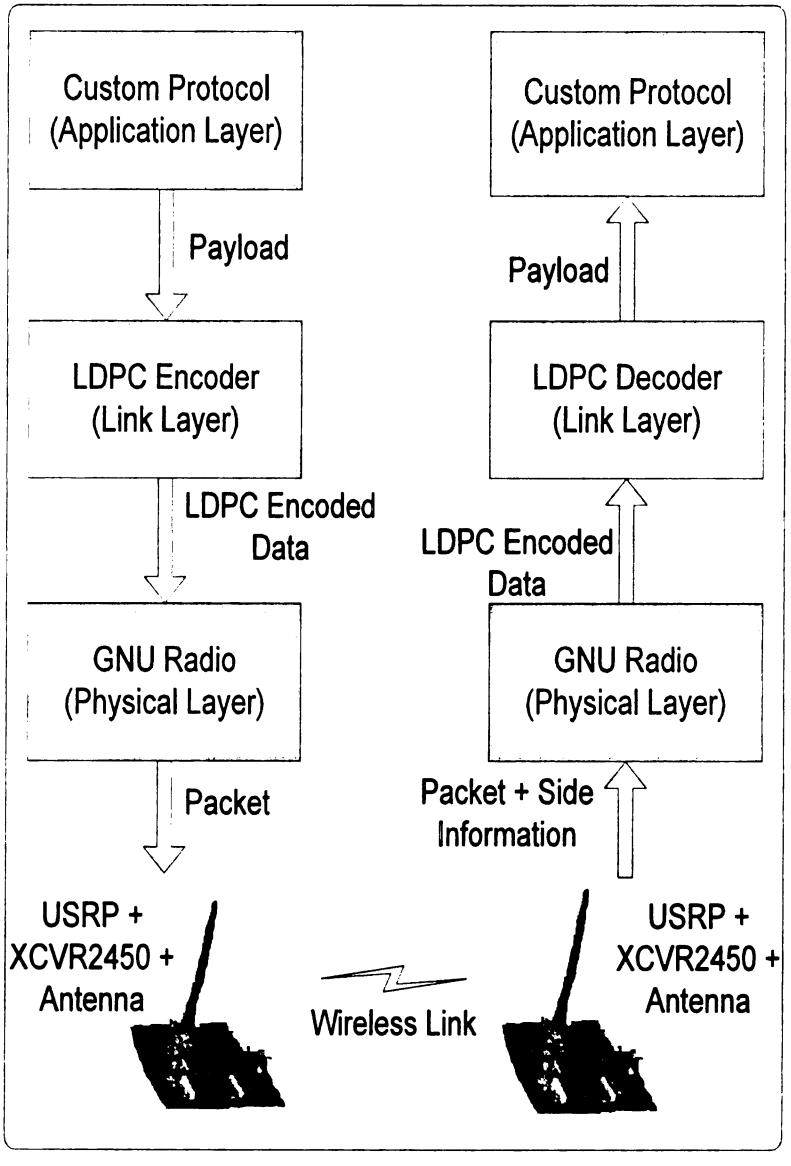

Figure V-I: Layered Architecture of our Experimental Setup

At the link-layer, the data packets are encoded to link-layer packets using a particular error control protocol. We used LDPC codes for this task. These packers are then sent to the PHY layer (i.e., GNU Radio) where they are transmitted over the wireless channel. We use Universal Software Radio Peripheral (USRP) as an RF frontend of GNU Radio. USRP can simultaneously receive and transmit on two antennas. Daughter-boards mounted on URSP provide flexible, fully integrated RF front-ends. USRP has an open design, with freely available schematics and drivers that can be integrated with GNU Radio. This enables us to modify MAC/PHY layer of our communication system and extract side information for error localization in received packets with residual errors.

Similarly, at the receiver end, packets are received by USRP. Here packet, along with side information is passed on to link-layer. LDPC decoder at link-layer decodes the packet and passes it on to Application layer on the top.

5.7- Experimental Test-Bed:

We use GNU-URSP platform to conduct experiments, collect traces and do analysis over real-time traffic data. In our work, we use XCVR2450 daughter-board which is a Dual-band Transceiver and transmits at 100+mW output at 2.4-2.5 GHz and 50+mW output 4.9-5.85 GHz. The XCVR2450 covers both the ISM band at 2.4 GHz and the entire 4.9 to 5.9 GHz band, including the public safety, UNII, ISM, and Japanese wireless bands.

Our experimental test-bed consists of two USRP devices: one connected to a desktop PC which we refer to as the *Data-Host (DH)* PC and the other connected to the *Data-Ghost (DG)* laptop computer. DH is hosting the data to be transmitted and DG is streaming

data from DH. In our experiments, the application layer in DH and DG respectively generates packet data and the acknowledgment packets. We set up the wireless communication system between DH and DG using the parameters as listed below.

We conducted our experiments in two phases. In the first phase, we collected traces to analyze the element of burstiness in the packet and to train our Markovian model. The detailed analysis for burstiness and the details of the deployed channel model can be found in Chapter-4 of the thesis. To train our Markovian channel model and to measure the number of errors introduced in every packet during a transmission session over a specific channel, in this phase of the experimental data collection, DH transmits predefined 100, 000 packets with 1500-byte packet payloads in every transmission session. Accordingly, DG captures the received packet and performs XOR operations with the predefined payload. Through this, we measure the number of byte (and bit) errors introduced in each packet.

In the second phase of experiments, using our GNU-USRP based wireless platform, we conducted several communication scenarios to capture the behavior of wireless error in the presence of fading, interference and mobility. We characterize wireless channel condition with average Bit Error Rate (BER) and average Packet Error Rate (PER). Our analysis consists of measuring 100 wireless channel conditions which are categorized in four groups:

1- *Environment factor*: To cater for the environments effects on wireless channel conditions, we carried out extensive set of experiments in both, open-spaced

out-door environment, i.e. outside the building, and in closed-space in-door environment, i.e. inside the building. For in-door setting, several factors contributed in wireless channel conditions as explained in coming lines. However, for out-door setting, the wireless communication was mostly line-of-sight (LOS) communication and we noticed that distance was the only metric for wireless connection failure and packet dropping.

2- *Fading factor*: In this set of scenarios, the DG laptop is located within certain distances of DH. The distance between DG and DH governs the impact of fading on the wireless communication. In particular, the error conditions for channels where DH and DG are relatively close to each other (e.g., less than 3m) are remarkably better than those with DG and DH are farther. It is also important to note that placing DG and DH very close to each other (e.g., less than ½ m) causes a lot of missing packet at DG due to interference caused by DH, so we did not consider such close distances.

3- *Interference factor*: To capture the impact of interference on wireless channel condition, we establish wireless communication in the presence of other wireless networks. These networks include Wi-Fi in our lab (i.e. WAVES lab), Wi-Fi in the building (i.e. College of Engineering), and in the presence of other hot-spots established over some laptop computers. It is evident that as more active wireless networks are presented in the environment, the likelihood of packet collisions increases leading to frequent packet errors due to interference.

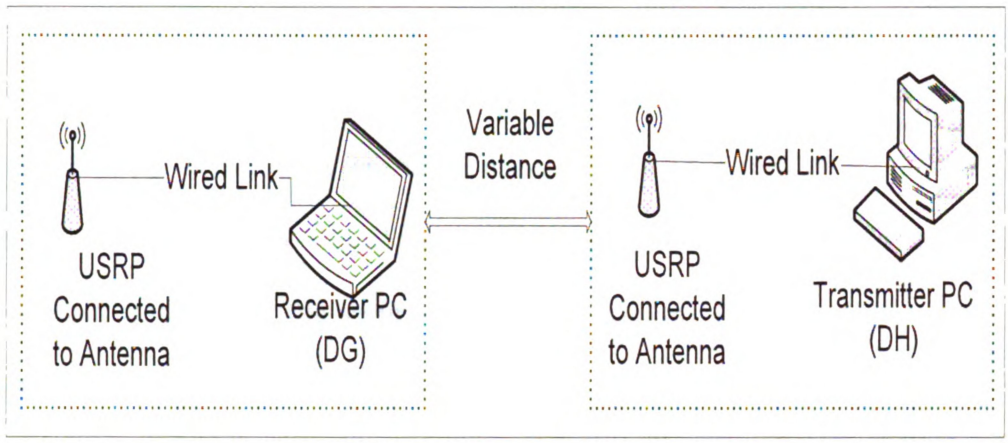

Figure V-II: Experimental Setup to see Effects of Environment and Fading Factors

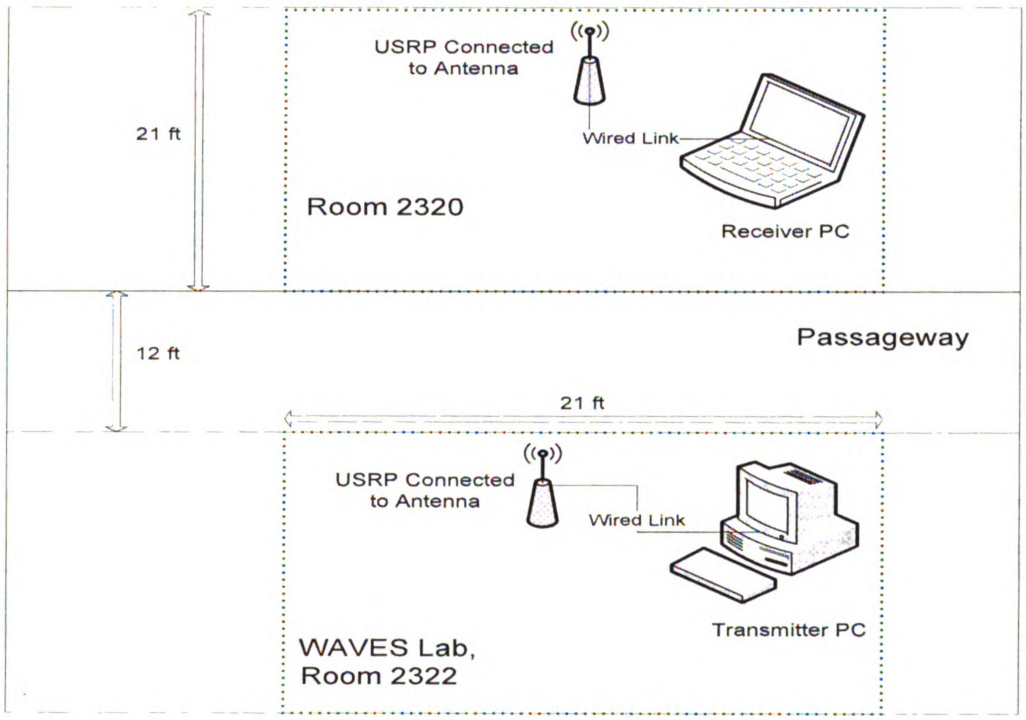

Figure V-III: Experimental Setup to see Effects of Mobility Factor

4- *Mobility factor*: The DH transmits data packets to the DG laptop which is a mobile node and changes its location frequently. Please figure III-III to get an idea of the room locations. In this set of experiments, the mobility is conducted within a room (i.e. WAVES lab), within the hallway (next to the room where DH resides), and within the room opposite to hallway. However the wireless connection never fails; all the transmitted packets are received by DG.

Using the GNU Radio-USRP based platform, we determine the performance efficiency of different link-layer protocols over above-mentioned channel conditions. In particular, we implement five link-layer protocols. One belongs to the Conventional Standard (CS) protocol category (IEEE802.11 ARQ), three belong to Forward Error Correction (FEC) protocol category (Hybrid ARQ type-I (HARQ-I), Hybrid ARQ type-II (HARQ-II), Reliable and Stable (RASE)), and one for Side-information Enhanced Forward Error Correction (SEFEC) protocol category (PRAISE) on our platform. The details of CS, FEC and SEFEC can be found in Chapter-1 of the thesis. These protocols reside in *link-layer* section of the layered architecture shown in Figure III-I and are implemented using C++ modules and glued in GNU-USRP platform with Python.

Though two identical USRPs are connected to both PCs, we will use the terms DH and DG for USRP and the associated PC jointly. Unless otherwise specified, following are the default parameters for conducting experiments and collecting channel traces:

Frequency:	5.1G
Packet Size:	1500 Bytes (Range = 0-4096 bytes)
No. of Packets in a session:	100,000
Modulation Scheme:	GMSK
Bursty Mode:	Disabled

We analyzed the performance of five network protocols; 1) Puncturing for Rate Adaptability of Reliable and Stable Wireless Protocols (PRAISE), 2) Reliable and Stable (RASE), 3) Hybrid ARQ-II, 4) Hybrid ARQ-I, and 5) Automatic Repeat Request (ARQ) over GNU-USRP platform. To keep discussion generic, we divided these protocols in three categories; 1) Conventional Standard (CS, like ARQ), 2) Forward Error Correction enabled Protocols (FEC, like HARQ-I, HARQ-II, and RASE), 3) Side-Information Enhanced Forward Error Correction enabled Protocols (SEFEC, like PRAISE). The details of CS, FEC and SEFEC can be found in Chapter-1 of the thesis.

To gauge the performance of these protocols over the wireless channel, we make use of two protocol-dependent parameters namely, 1) throughput; 2) goodput. We define throughput over a transmission channel as:

$$throughput = \#\ of\ bits\ received\ correct\ /total\ \#\ of\ bits \quad \ldots\ldots (5.1)$$

And we define *goodput* over a transmission channel as:

$$goodput = \#\ of\ correct\ bits\ containing\ actual\ data/total\ \#\ of\ bits \quad \ldots\ldots (5.2)$$

So for a session, goodput becomes ratio of total number of bits that contained actual data and are received correctly to the total number of bits transmitted for that session. In this way, the goodput parameter excludes parity bits from calculations of throughput values.

We compare the values of the above-mentioned protocol-dependent parameters against two wireless channel metrics, 1) Packet Error Rate (PER); 2) Bit Error Rate (BER). *Bit Error Rate (BER) is defined as the rate at which errors occur in a transmission system.* BER is a unit-less performance quantity and is expressed as a percentage number. The definition of bit error rate can be expressed as:

$$BER = number\ of\ bits\ in\ errors/total\ number\ of\ bits \quad \ldots\ldots (5.3)$$

And, t*he Packet Error Rate (PER) is the number of incorrectly transferred data packets divided by the number of transferred packets*. A packet is assumed to be incorrect if at least one bit is incorrect. PER is a unit-less performance quantity and is expressed as a percentage number. Mathematically, PER can be expressed as:

$$PER = number\ of\ packets\ in\ errors/total\ number\ of\ packets \quad \ldots\ldots (5.4)$$

Hence our results (explained in Chapter-7) fall in two categories depending upon wireless channel metrics and we measure each of the two protocol-dependent parameters against each of the channel metric.

Depending on channel conditions, we vary the code rate from (10-50)% and collected several traces. From the first phase of experiments, we concluded that the code-rate of

33.33% is optimum for performance in varying channel conditions for fixed rate FEC schemes. It is neither too much to affect goodput in good channel conditions nor that bad to cause a lot of retransmissions in bad channel conditions.

Chapter 6 - Analysis of Burstiness in a Packet

An Error Burst over a data transmission channel can be defined as a contiguous sequence of n symbols such that the first and last symbols are in error. In this way, there does not exist any contiguous subsequence of *m* correctly received symbols within the error burst of *n* symbols, where $m < n$.

The Length of a Burst of Bit Errors in a Frame is defined as the number of bits from the first error to the last, both inclusive. The integer parameter *m* is used to specify the length of the burst and is referred to as the *Guard Band* of the error burst. The last symbol in a burst and the first symbol in the following burst are accordingly separated by *m* correct bits or more.

We explained our experimental setup in Chapter-5 of the thesis. Our experimental test-bed consists of two USRP devices: one connected to a desktop PC which we refer to as the *Data-Host (DH)* PC and the other connected to the *Data-Ghost (DG)* laptop computer. DH is hosting the data to be transmitted and DG is streaming data from DH. In our experiments, the application layer in DH and DG generates packet data and the acknowledgment packets respectively. We set up the wireless communication system between DH and DG using the parameters detailed in Chapter-5.

Using our GNU-USRP based wireless platform, we conducted several communication scenarios to capture the behavior of wireless error in the presence of fading, interference and mobility. As we explain in Chapter-5, we conducted our experiments in two phases. In the first phase, we collected traces to analyze the element of burstiness

in the packet and to train our Markovian model. To measure the amount of error introduced on every packet during a transmission session over a specific channel, in this phase of the experimental data collection, DH transmits 100, 000 packets with predefined 1500 byte packet payloads in every transmission session. Accordingly, DG captures the received packet and performs XOR operations with the predefined payload. Through this, we measure the number of byte (and bit) errors introduced in each packet. Our proposed scheme enhances end-to-end bandwidth utilization in case of bursty errors in a packet. So burstiness analysis for data in a received packet in this phase of date collection constitutes an important part of the thesis. This burstiness analysis is divided into three following subtopics:

1- Distribution of Errors in a Packet

2- Bit Error Rate (BER) vs. Cumulative Density Function (CDF)

3- End-to-End (E2E) Burst in a Packet

6.1- Distribution of Errors in a Packet

We define *Distribution of Errors* as *the arrangement of continuing and successive errors in space*. Hence the error distribution describes the range of possible values that an error variable can attain in succession and the probability that the value of the error variable is within any (measurable) subset of that range. In our experimental setup, space consists of a packet length and the unit to specify errors is taken as bytes. We show plots of the error distributions of the collected traces in the following pages to get

proof of the proposed idea. We are including three error distributions in the thesis for reference purposes.

Each of the error distribution plots burst length vs. number of received packets in a session. In this way, x-axis corresponds to number of successive error bytes in a packet of length 1500 bytes, and y-axis refers to the %age of received packets containing specific error burst lengths. To explore burstiness of errors in a packet, we scaled our burst length axis from 2-1500 because a burst length of zero refers to a packet without errors (i.e. a packet that was received correct) and a burst length of one shows an alternate occurrence of a pattern of zeros/ones referring to sporadically distributed error bytes. The resulting error distributions are as shown below in figures VI-I, VI-II, and VI-III.

Figure VI-I shows distribution of errors in traces with Packet Error Rate, PER, in the range of 30-40%. Since the channel in this case is in 'good' state, we notice values of burst lengths represented by a smaller percentage of received packets. For example, the highest burst length is around 80 bytes and only 4.2% of received packets have a burst length of 80 bytes. Furthermore, except the burst length of 85 bytes, represented by 3.4% of received packets, all other burst lengths can be seen in less than 1.5% of received packets. As we move-on to the traces with higher PER values, we notice that burstiness in a packet increases and we start observing packets with greater burst lengths. For example, in traces with PER values ranging between 60-70% and 80-90%,

the highest burst length values are again around 80 bytes, but this value is represented by 7.3% and 8% of the received packets respectively, as shown in figures VI-II and VI-III.

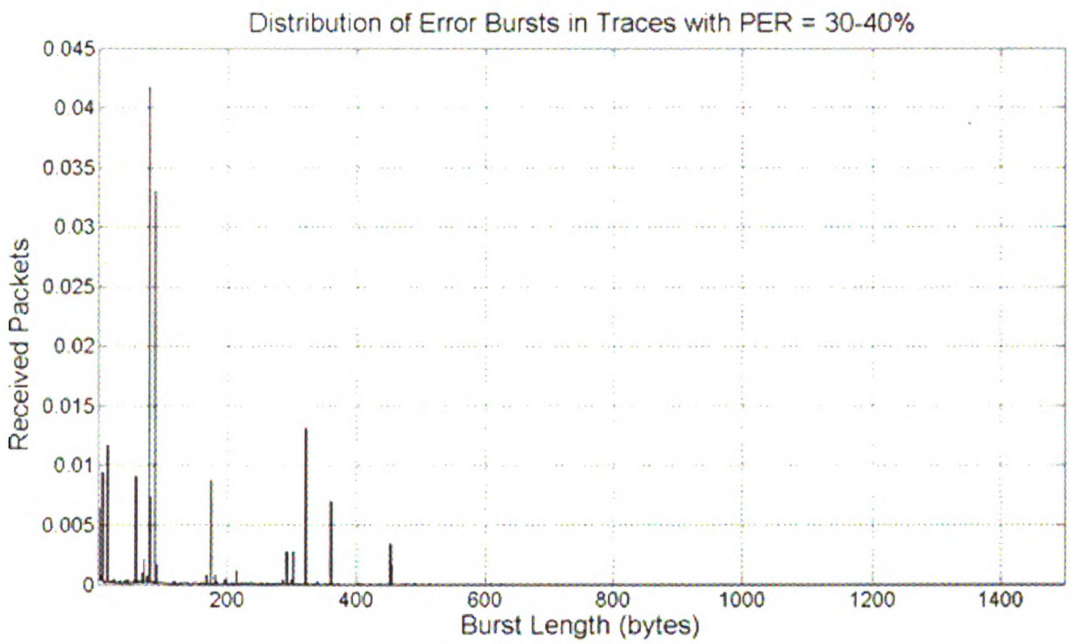

Figure VI-I: Distribution of Errors in Traces with PER = 30-40%

Moreover, we find significant peaks in burst distributions represented by over 2% and over 4% of received packets in traces with PER values 60-70% and 80-90% respectively.

From these error distributions, we see that burstiness at packet level increases with higher PER values. So we conclude here that as the channel state gets worse, the error distributions increase too resulting in increased burstiness effects at packet level. This gave us the motivation to design codes catering for bursty errors in a packet and coping with bursty nature of wireless channels by adapting code-rates in varying channel conditions.

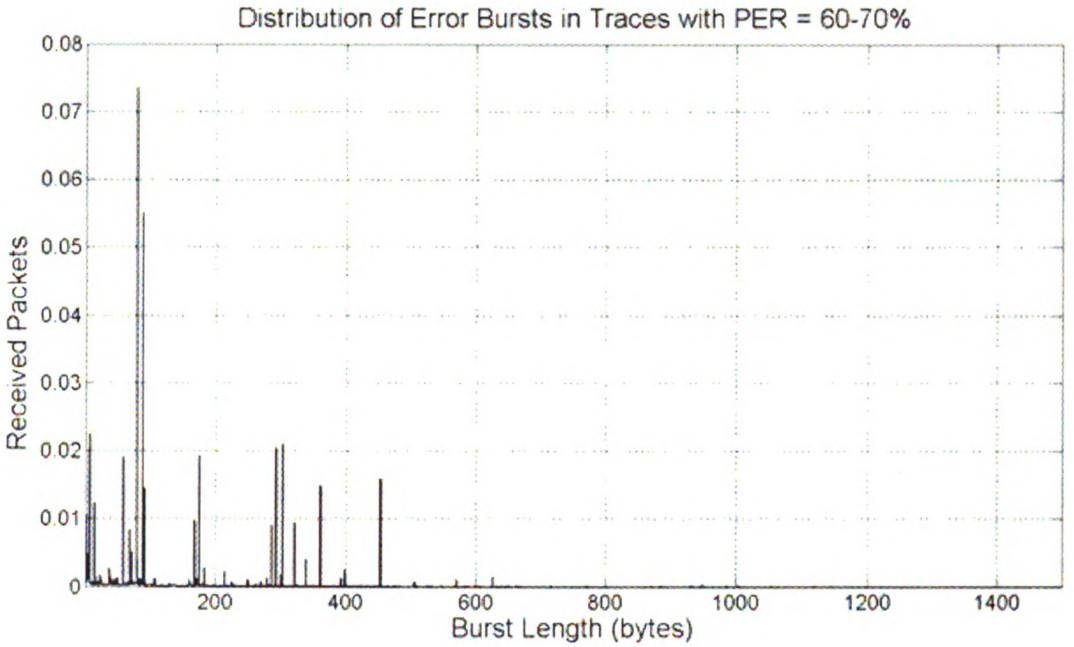

Figure VI-II: Distribution of Errors in Traces with PER = 60-70%

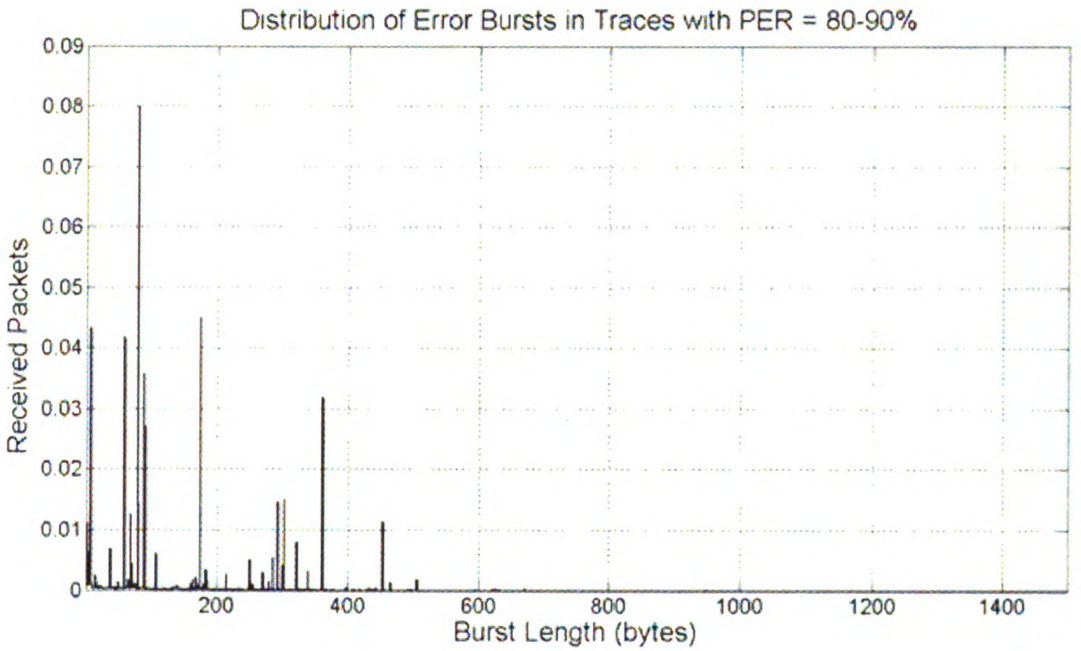

Figure VI-III: Distribution of Errors in Traces with PER = 80-90%

6.2- Bit Error Rate (BER) vs. Cumulative Density Function (CDF):

We know that a probability distribution can be completely described by its cumulative distribution function when a random variable takes values in the set of real numbers, whose value at each real value *x* is the probability that the random variable is smaller than or equal to *x*. *Cumulative Density Function* (CDF) of error distribution, plotted against *Bit error rate* (BER) is another useful metric to gauge burstiness in a packet.

Suppose we have a codeword $C_i(K_i, X_i)$, where, K_i data symbols are coded using X_i parity symbols. If we define *Distortion* as *an undesired change in the output signal resulting from imperfections in the wireless channel*, then the *Distortion Level* E_i is always random and unknown to the receiver and therefore the notion of partial recovery is unrealistic in the context of error correction. This means that the receiver can either correct all errors in C_i and declare successful decoding, or just states that no recovery is achieved.

The receiver attempts to retrieve K_i data symbols by utilizing X_i parity symbols embedded in C_i. Depending on the decoding algorithm, the receiver can correct up to a certain threshold of error directly proportional to the number of parity symbols embedded in the message. For instance, for an error correcting code with error correction capability α, for X_i parity symbols, the receiver is capable of correcting up to $(\alpha * X_i)$ errors out of $|C_i|$ symbols in the message. Here α measures the expected error-correcting capability of a particular rate decoder. For example, the error-

correcting capability of Reed-Solomon codes is half as many as redundant symbols (i.e., $= 0.5$).

The parameter α, thus, provides and upper bound on the error correction capability of a decoder and it is approximated by having thresholds over CDF of error distribution. Figure IV-VI plots BER vs. CDF for traces with shown PER values. For PER values of 10-15%, since the channel is in a 'good' state, we notice that up to 80% corrupted packets can be successfully decoded if provided *enough parity data* to correspond for a BER of 0.7. This means a decoder with $\alpha = 0.7$ can correct 80% packets in such a scenario. For other cases, to correct 80% of corrupted packets, the corresponding α-values are 0.78, 0.85 and 0.93 for PER values of 40-45%, 60-65% and 80-85% respectively.

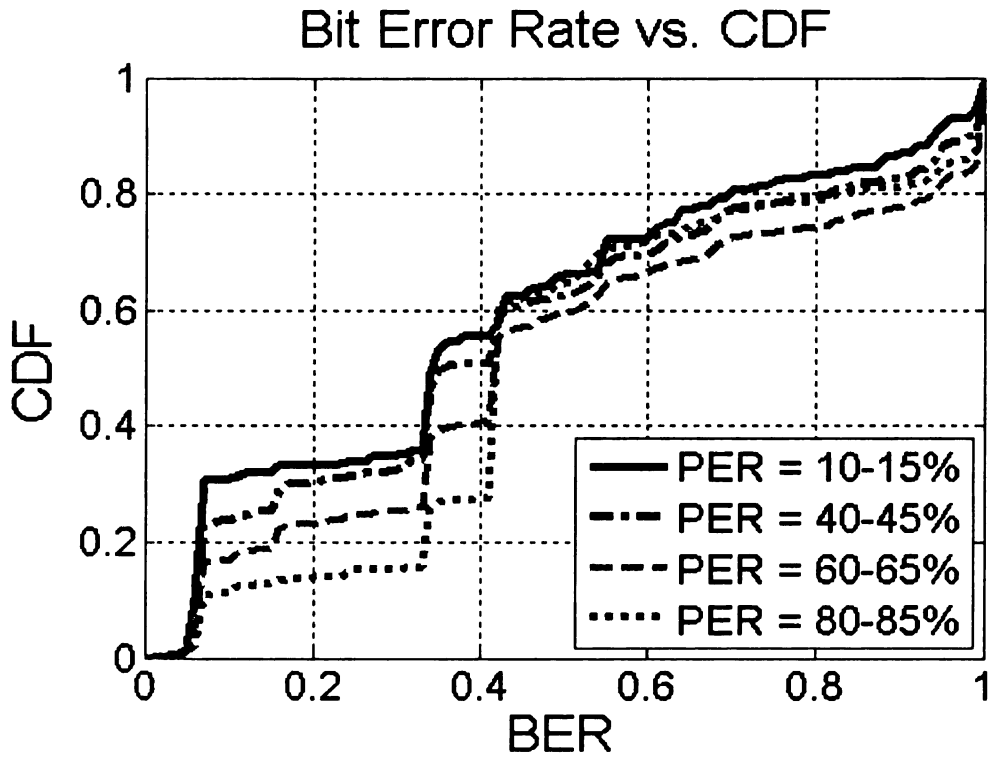

Figure VI-IV: BER vs. CDF

6.3- End-to-End (E2E) Burst in a Packet:

We define *End-to-End (E2E) burst* as *the number of bits between the first error bit and the last error bit, both inclusive, in a packet received over a transmission channel.* This definition provides room for existence of multiple contiguous subsequences of *m* correctly received symbols and error bursts of *n* symbols within E2E burst.

E2E burst is a useful way of looking into corrupted area of a packet. It gives an overall/big picture view of the faulty portion of the packet. This caters for even the sporadic distribution of errors inside a packet, providing the code designer an insight to design codes keeping in view the fact that errors may be randomly distributed inside a packet, aside from their occurrence in bursts. So, E2E burst length specifies the total corrupted portion of a received packet enabling one to design parity data keeping in view the maximum possible length of burst. We, in our work, used E2E burst length as a measure to analyze the traces and found the results as given below. For E2E burst length plots, y-axis shows percentage of the corrupted received packets and x-axis shows the percentage of the faulty portion in a packet.

Figure VI-V shows E2E burst length for traces with Packet Error Rate, PER, of 40-50%. We see that about 30% of the corrupted received packets have E2E burst length equal to 10% of the packet. This means, in case of decoding failure, if parity data is transmitted only for 10% corrupted portion of a packet, about 30% of the corrupted received packets can be recovered. The corrupted portion can easily be identified by the side-information, as in our case. Then, each of 20%, 30% and 40% E2E burst length is

found in 10% of the corrupted packets. These results are very favorable and give proof of our idea to divide a packet into chunks and transmit parity data with the help of side-information for selected subsets only, instead of:

1- discarding whole packet, as in case of Conventional Standard (CS) schemes, like IEEE 802.11 ARQ schemes, explained in Chapter-I, or

2- requesting parity data for whole packet like in case of Forward-Error-Correction (FEC) schemes, like Automatic Code Embedding (ACE), explained in Chapter-I

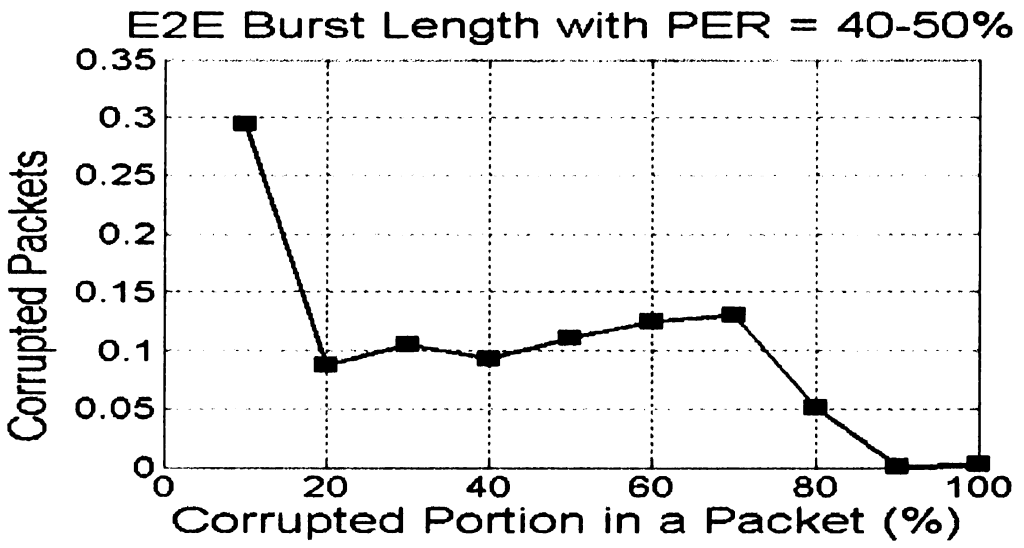

Figure VI-V: E2E Burst Length for Traces with PER = 40-50%

Figure VI-VI shows E2E burst length for traces with PER 70-80%. It is important to note that this PER value corresponds to a channel in very bad state. Only 3-4% of corrupted packets have 10% E2E burst length. Here, each of 60%, 70% and 80% E2E burst length is found in 10% of the corrupted packets. Still we see that about 3-4% of the corrupted

received packets can be recovered by requesting parity data for only 10% of the packet size.

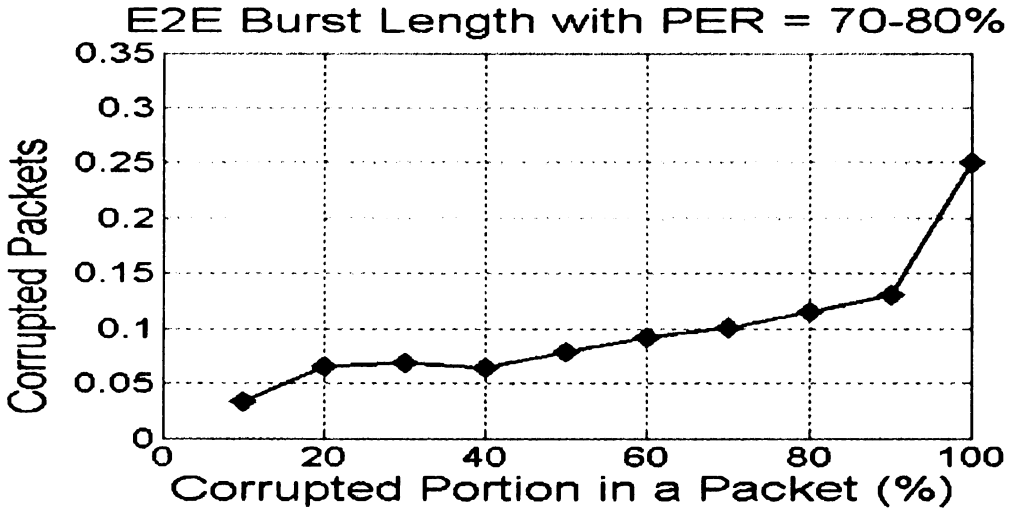

Figure VI-VI: E2E Burst Length for Traces with PER = 70-80%

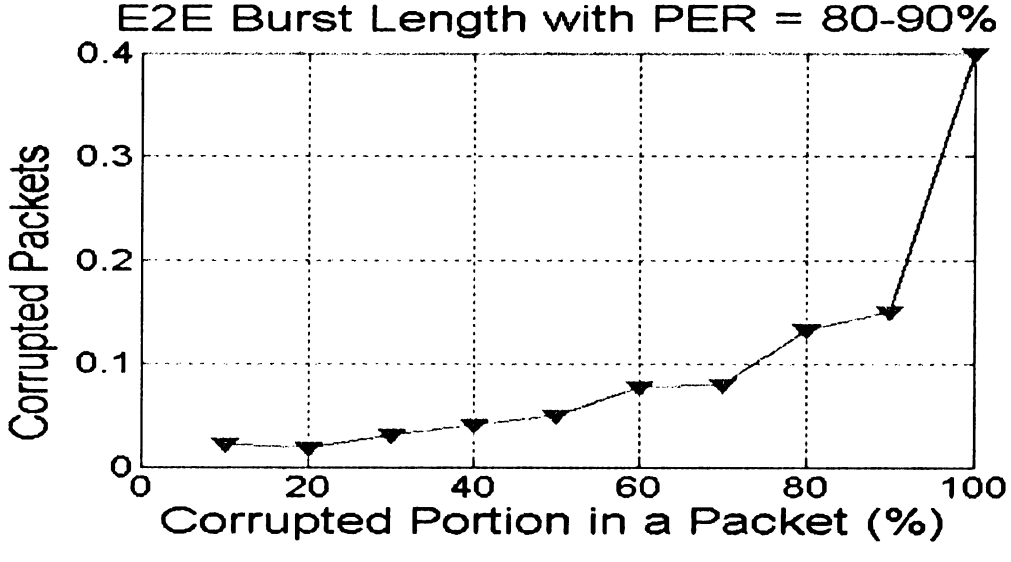

Figure VI-VII: E2E Burst Length for Traces with PER = 80-90%

Figure VI-VII shows E2E burst length for traces with PER 80-90%. Notice that this very high PER value corresponds to a channel in extremely bad state. Less than 1% of corrupted packets have 10% E2E burst length. Here, each of 70%, 80% and 90% E2E burst length is approximately found in 10% of the corrupted packets. Even in this scenario, over 10% packets require parity data for only 70% of the received packet.

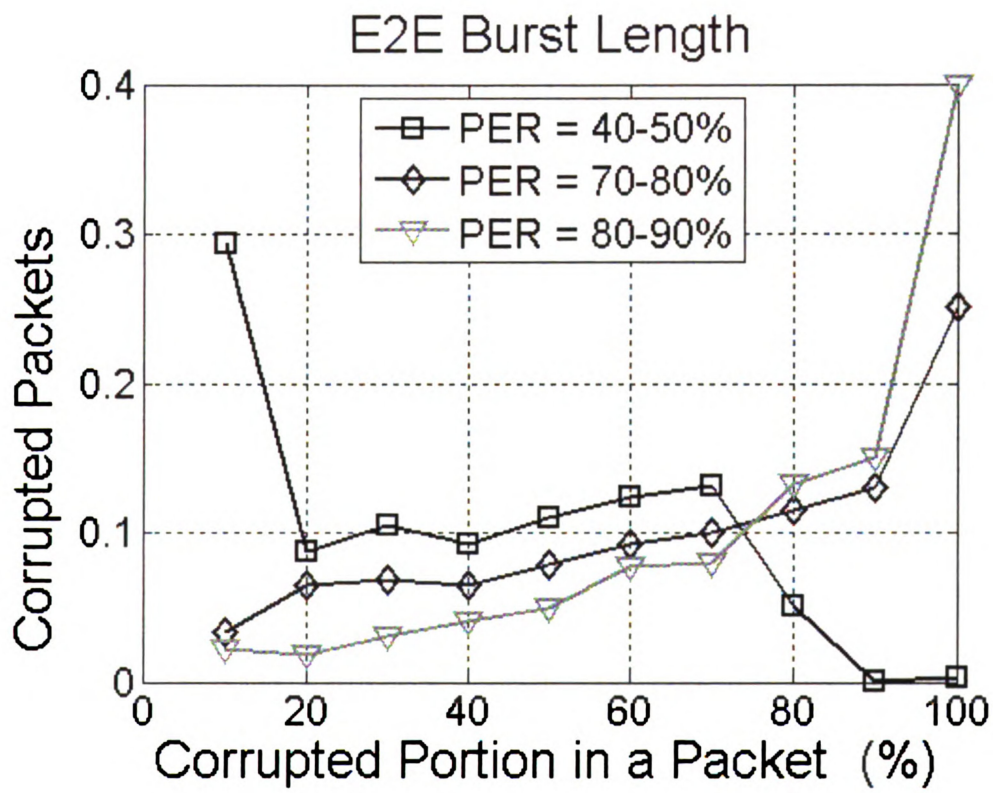

Figure VI-VIII: E2E Burst Length

Figure VI-VIII shows plots above in a single canvas for comparison purposes. We notice that as channel state gets worse and worse, E2E burst length increases on average. This reason for such results is that increasing PER values correspond to higher BER values, which contribute in increasing E2E burst lengths. As we mentioned in Chapter-I that

some previous works have made use of incremental parity data to cope with varying channel conditions. Examples of such works Automatic Code Embedding (ACE) and Reliable and Stable (RASE). But none of these works use side-information to enhance their performance. In fact, ACE requests incremental parity data for whole packet in all channel conditions. This leads to requesting parity data for subsets of a corrupted received packet which do not require any parity data for decoding. As a result, we see in Chapter-V that PRAISE outclasses state-of-art network protocols in terms of throughput, goodput and end-to-end bandwidth utilization in good and average network conditions. Even when the channel is in extremely bad states, PRAISE's performance is comparable to that of RASE as in extremely bad channel conditions, parity data for whole packet (i.e. all of the subsets) is needed to be transmitted.

Chapter 7 – Results, Conclusions and Future Work

Let us recall that in Chapter-1, in order to keep the discussion generic and not dependant on a particular implementation or standard, we consider three rather abstract communication schemes: 1) transmission over error/erasure channels, which represents the conventional standard (CS) protocols, like IEEE Automatic Repeat Request (ARQ) schemes; 2) transmission in the presence of a forward error-correction (FEC) based schemes, like Hybrid ARQ [38, 39] and more recent schemes like ZIPTX [37], Automatic Code Embedding (ACE) [34], and Reliable and Stable (RASE) [40]; 3) side-information enhanced transmission in presence of both, erasures and errors, using a forward-error correction scheme (SEFEC) like the proposed scheme, Parity localization for Rate Adaptability of Reliable and Stable Protocols (PRAISE).

The two generic aspects of 1) forward error correction; 2) feedback mechanism (FB), are enough to explain the three communication schemes as described above. Forward error correction part can simply be segregated into two categories consisting of presence or absence of any forward error correction mechanism. Feedback mechanism can have many variants, ranging from simple binary feedback about reception of correct/false packet like in case of ARQ, to the feedback with flags requesting additional parity data for corrupted packets like in case of recent protocols like ACE [34] and RASE [40]. So under these two aspects, the following happens.

1) CS: the conventional standard protocol contains no FEC and requests the same packet even if a single bit is in error, example being IEEE802.11 ARQ scheme. In

IEEE802.11 ARQ protocol, error-detection information (ED) bits are added to data to be transmitted (such as cyclic redundancy check, CRC) and retransmissions are solution for losses. If a corrupted packet is received, it is discarded without regard to the number and location of errors. This methodology ensures that the packet would eventually be recovered. However, this approach leads to a great deal of throughput degradation as even a single bit error leads to packet drops resulting in discard of a large number of correctly received data bits, ending up 'wasting' a lot of useful data. Thus, network utilization deteriorates rapidly as channel bit error rate (BER) increases.

2) FEC: FEC based protocols, which represent schemes like IEEE802.11 Hybrid ARQ schemes [38, 39] and more recent schemes like ZIPTX [37], ACE [34] and RASE [40], contain some sort of forward error correction and request additional parity data in case of reception of a corrupted packet. Hybrid ARQ (HARQ) protocols are proposed as an alternative to CS. In HARQ protocols, forward error correction bits are also added to the existing Error Detection (ED) bits, such as Reed-Solomon code, low-parity density-check code or Turbo code. In this way, these schemes reduce the number of re-transmissions as required in case of ARQ by making use of incremental channel codes. However, both ARQ and HARQ based approaches do not address the throughput stability issues in varying channel conditions, and thus lead to a great deal of throughput inefficiency. Other protocols like ZIPTX and ACE address the issues of reliability and/or stability in varying channel conditions but these fall short on many fronts. ZIPTX,

though provides a working system, ignores any aspect of stability. ACE takes into account both the issues of reliability and stability by embedding channel codes using well-defined code rates but ACE, i) lacks a practical demonstration system, ii) do not consider rate-adaptability to address throughput efficiency in changing network conditions.

3) SEFEC: SEFEC is an alternative to above schemes. Similar to FEC based schemes, an SEFEC scheme requests additional parity using feedback flags, but the requested parity data is localized and request depends upon side-information received from physical (MAC) and link layers. So SEFEC is a cross-layer design which extracts beliefs associated with received data either at MAC/physical layer, or at link layer and decides to requests additional parity data for a particular subset of code depending on values of beliefs for a particular subset of code. Some previous works [35, 41, 42] have proposed cross-layer designs for multimedia applications to overcome throughput degradations and performance limitations imposed by traditional protocols. An example of such a work is UDP Lite [41], which relies on the error-resilient nature of multimedia content and makes adjustments to the protocol stack at the transport and the link layers in order to improve the bandwidth utilization. A major drawback of the existing cross-layer protocols is that their implementations require key modifications in transport and application layers. However the proposed cross-layer SEFEC design requires no modification in the subsequent layers since a single decoder is required for 'mother' and 'daughter' codes and side-information used to request

parity data is received from physical (MAC) and link layers. Therefore, and unlike receivers of FEC- based schemes, SEFEC receiver has RHCs, is rate-adaptive depending on varying channel conditions, distinguishes between erasures and errors in a received packet, and does not require any modification in transport and application layers.

As we explained in Chapter 5 that current wireless network devices such as USB network adaptors, Peripheral Component Interconnect (PCI) wireless adaptor cards for computer desktops and notebooks, wireless Ethernet bridges, and Wireless Compact Flash Card Adapters for PDAs etc. are utilized to establish communication channels among different wireless users in various wireless environments. By design, these devices use IEEE802.11 standard to attain channel access, contention, and error control at the Physical (PHY) and MAC layers. IEEE 802.11 standard has some sort of basic error correction at PHY layer, like Reed Solomon codes etc. and this design strategy naturally leads to preventing PHY layer to pass distorted packets to the higher layers. The distorted packets are dropped right away at PHY layer, wasting a lot of useful data. So we faced the major challenge of utilization and alternation of link-layer corrupted packets, containing residual errors, at the receiver to:

1- Capture and measure the behavior of an error process of a wireless channel,

2- Analyze the residual errors, at link-layer, in a wireless channel to design suitable puncturing fractions in localized manner for rate-adaptability of error control protocols,

3- Implement error control protocols at the link-layer that employ and manipulate received corrupted packets, and
4- Extract side information from MAC/PHY layers to request localized parity for subsets with potentially highest number of errors.

To cope with this issue, we need to modify the configuration of these devices to attain the distorted packets at the PHY/MAC layer. But, these off-the-shelf wireless devices are not open source products, thus modification of the source code functionality is not possible.

To overcome this problem, we use Software-Defined Radio (SDR) technology. Specifically, we use GNU-USRP platform as the communication system front-end on each sender and receiver node in our experiments. Since, GNU Radio is an open-source software development toolkit, we are able to easily reconfigure and modify its building blocks. In this way, we have complete control over the communication system configurations down to the FPGA/antenna level, and down to PHY/MAC layers and we are able to capture distorted packets received at the PHY/MAC layer and pass packets with residual errors to link layer. Furthermore, we are able to extract side information associated with each packet which we used to request additional parity data from encoder in case of decoding failure.

Unless otherwise specified, following are the default parameters for conducting experiments and collecting channel traces:

Frequency:	5.1G
Packet Size:	1500 Bytes (Range = 0-4096 bytes)
No. of Packets in a session:	100,000
Modulation Scheme:	GMSK
Bursty Mode:	Disabled

To gauge the performance of the CS, FEC and SEFEC protocols over the wireless channel, we make use of two protocol-dependent parameters namely, 1) throughput; 2) goodput. We define throughput over a transmission channel as:

$$throughput = \#\ of\ bits\ received\ correct\ /total\ \#\ of\ bits \quad \ldots\ldots (7.1)$$

And we define *goodput* over a transmission channel as:

$$goodput = \#\ of\ correct\ bits\ containing\ actual\ data/total\ \#\ of\ bits \quad \ldots\ldots (7.2)$$

So for a session, goodput becomes ratio of total number of bits that contained actual data and are received correctly to the total number of bits transmitted for that session. In this way, the goodput parameter excludes parity bits from calculations of throughput values.

We compare the values of the above-mentioned protocol-dependent parameters against two wireless channel metrics, 1) Packet Error Rate (PER); 2) Bit Error Rate (BER). So our results fall in two categories depending upon wireless channel metrics and we measure each of the two protocol-dependent parameters against each of the metric.

7.1- Wireless Channel Metric BER

Bit Error Rate (BER) is defined as the rate at which errors occur in a transmission system. BER is a unit-less performance quantity and is expressed as a percentage number. The definition of bit error rate can be expressed as:

$$BER = number\ of\ bits\ in\ errors/total\ number\ of\ bits \quad \ldots\ldots (7.3)$$

BER is a parameter which gives an excellent indication of the performance of a wireless channel. As one of the main parameters of interest in any channel is the number of errors that occur, the bit error rate is a key parameter. So we chose it as one of the channel metrics to gauge the performance of protocol-dependent parameters.

Figure VII-I shows the plot for Throughput vs. Channel BER for the five protocols falling in three categories. From plot below, we see that the Conventional Standard (CS) ARQ suffers the most with increasing channel BER values. The average throughput of the CS-ARQ scheme falls to 10% as channel's average BER increases to 0.1. RASE makes use of incremental parity data to retrieve packets, resulting in increased channel bandwidth and throughput, and hence it performs better than HARQ-I and HARQ-II among FEC schemes. It is evident from the plot that SEFEC-PRAISE outperforms all CS and FEC schemes in terms of throughput in all channel conditions. PRAISE's performance is comparable to CS and FEC schemes in channel with very low BER values. But as channel BER increases, PRAISE's superiority over CS and FEC schemes becomes evident. At the channel's average BER of 0.15, PRAISE outperforms the state-of-art FEC scheme, RASE by about 12% high throughput value.

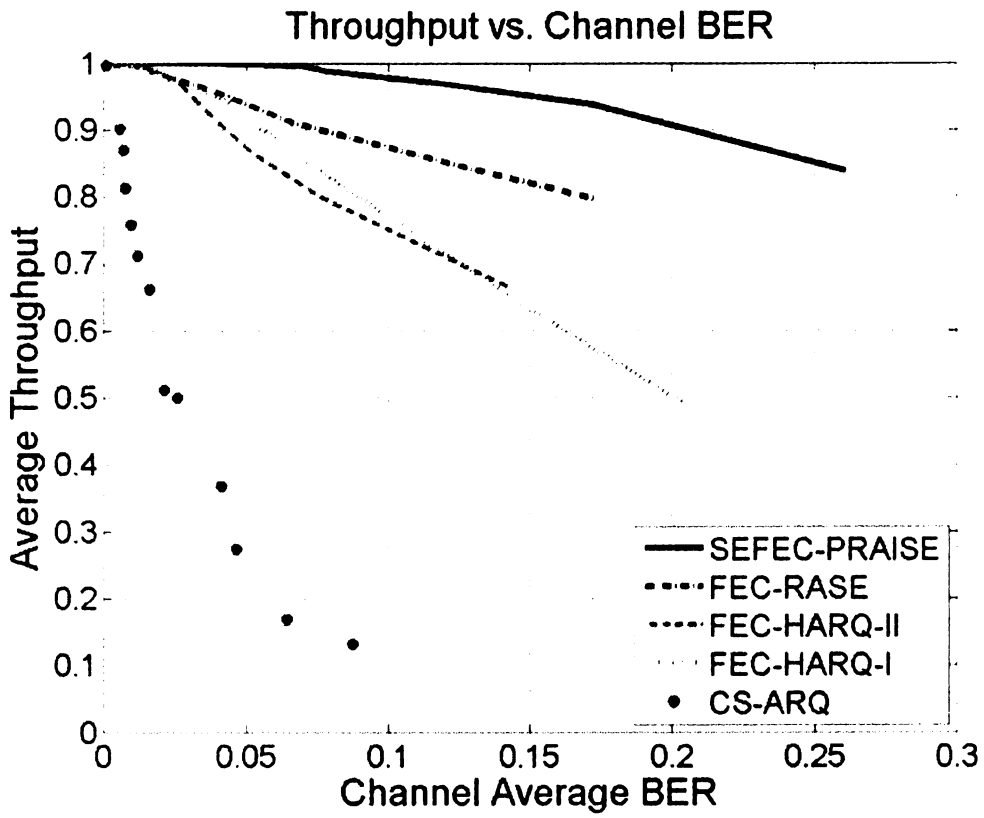

Figure VII-I: Throughput Vs. Channel BER

Figure VII-II shows the plot for Goodput vs. Channel BER for the five protocols falling in three categories. From plot below, we again see that the Conventional Standard (CS) ARQ suffers the most with increasing channel BER values. When the channel is in good condition (BER<0.02), CS-ARQ's performance is the best among all five protocols as in these scenario, goodput and throughput values are equal and all the data received is the '*useful*' data in case of CS-ARQ scheme. But as BER values increase, CS-ARQ's goodput drastically falls. Since the values of goodput and throughput are same in case of CS-ARQ, we see that the average goodput of the CS-ARQ scheme falls to 10% as channel's average BER increases to 0.1.

In FEC schemes, forward error correction bits are added to the existing ED bits to correct a subset of all errors while relying on ARQ to detect uncorrectable errors. As a result FEC-HARQ I/II perform better than CS-ARQ in poor signal conditions, but in its simplest form this comes at the expense of significantly lower goodput in good signal conditions. Here the signal quality cross-over point below which FEC-HARQ is better, and above which CS-ARQ is better is at BER = 0.02 in case of HARQ-II and at BER = 0.03 in case of HARQ-I. RASE makes use of incremental parity data to retrieve corrupted packets, resulting in increased channel bandwidth and goodput, and hence it performs better than HARQ-I and HARQ-II among FEC schemes. Here the signal quality cross-over point below which FEC-RASE is better, and above which CS-ARQ is better is at BER = 0.01.

It is evident from the plot that SEFEC-PRAISE outperforms FEC-HARQ I/II schemes in terms of goodput in all channel conditions. The signal quality cross-over point below which SEFEC-PRAISE is better, and above which CS-ARQ is better is at BER = 0.01. Since PRAISE needs additional $f = \left\lceil log_2 \binom{c}{l} \right\rceil$ feedback bits to request parity for l out of c chunks of lowest reliability, its performance in terms of goodput suffers insignificantly (due to small number of f bits) than that of FEC-RASE at lower BER values (BER<0.02). But as channel BER increases to 0.15, it outperforms state-of-art FEC-RASE by 0.9% higher goodput values.

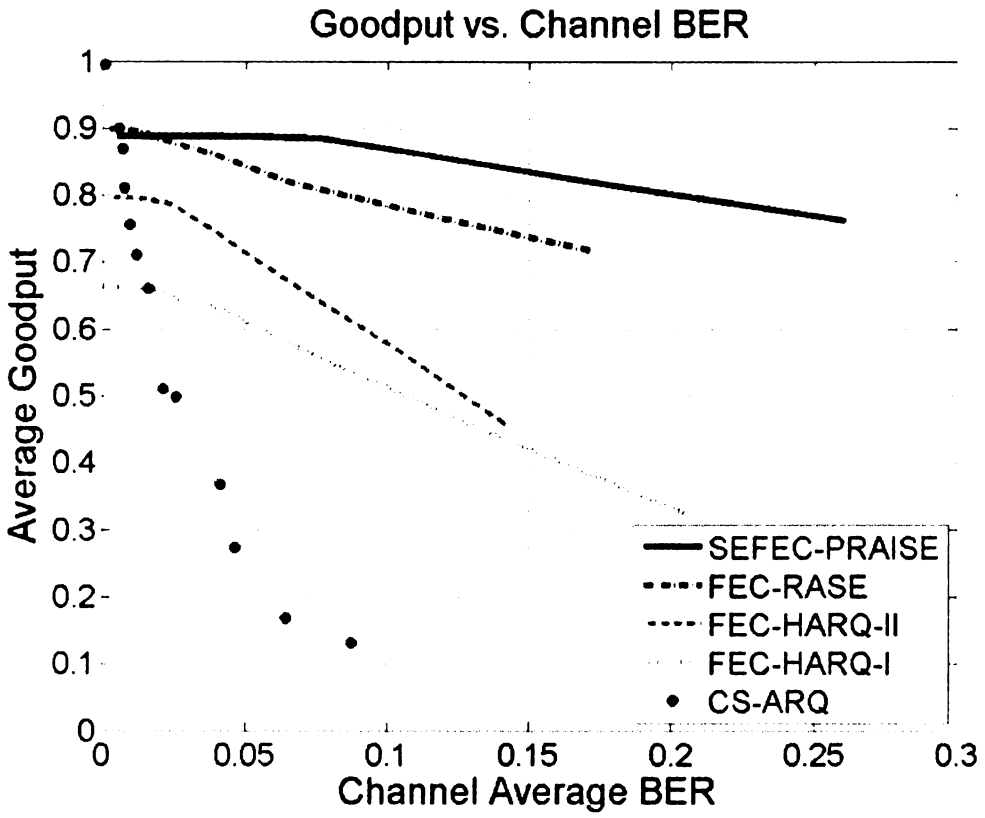

Figure VII-II: Goodput Vs. Channel BER

7.2- Wireless Channel Metric PER

The Packet Error Rate (PER) is the number of incorrectly transferred data packets divided by the number of transferred packets. A packet is assumed to be incorrect if at least one bit is incorrect. PER is a unit-less performance quantity and is expressed as a percentage number. Mathematically, PER can be expressed as:

$$PER = number\ of\ packets\ in\ errors/total\ number\ of\ packets \quad \ldots\ldots (7.4)$$

PER is a parameter which gives an excellent big-picture of the overall performance of a wireless channel. As one of the main parameters of interest in any channel is the frequency of occurrence of errors, the packet error rate is a key parameter. That's why

we chose it as one of the channel metrics to gauge the performance of protocol-dependent parameters.

Figure VII-III shows the plot for Throughput vs. Channel PER for the five protocols (ARQ, HARQ-I/II, RASE and PRAISE) falling in three categories of CS, FEC and SEFEC as described above.

Here we again see that the Conventional Standard (CS) ARQ suffers the most with increasing channel PER values. The average throughput of the CS-ARQ scheme falls to 10% as channel's average PER increases to 0.85. RASE makes use of incremental parity data to retrieve packets, resulting in increased channel bandwidth and throughput, and hence it performs better than HARQ-I and HARQ-II among FEC schemes. It is evident from the plot that SEFEC-PRAISE outperforms all CS and FEC schemes in terms of throughput in all channel conditions. PRAISE's performance is comparable to FEC schemes in channel with low PER values. But as channel PER increases to over 20%, PRAISE's superiority over FEC schemes becomes evident. At the channel's average PER of 0.80, PRAISE outperforms HARQ-I by about 27%, HARQ-II by 23%, and the state-of-art FEC scheme, RASE by about 4% higher throughput value.

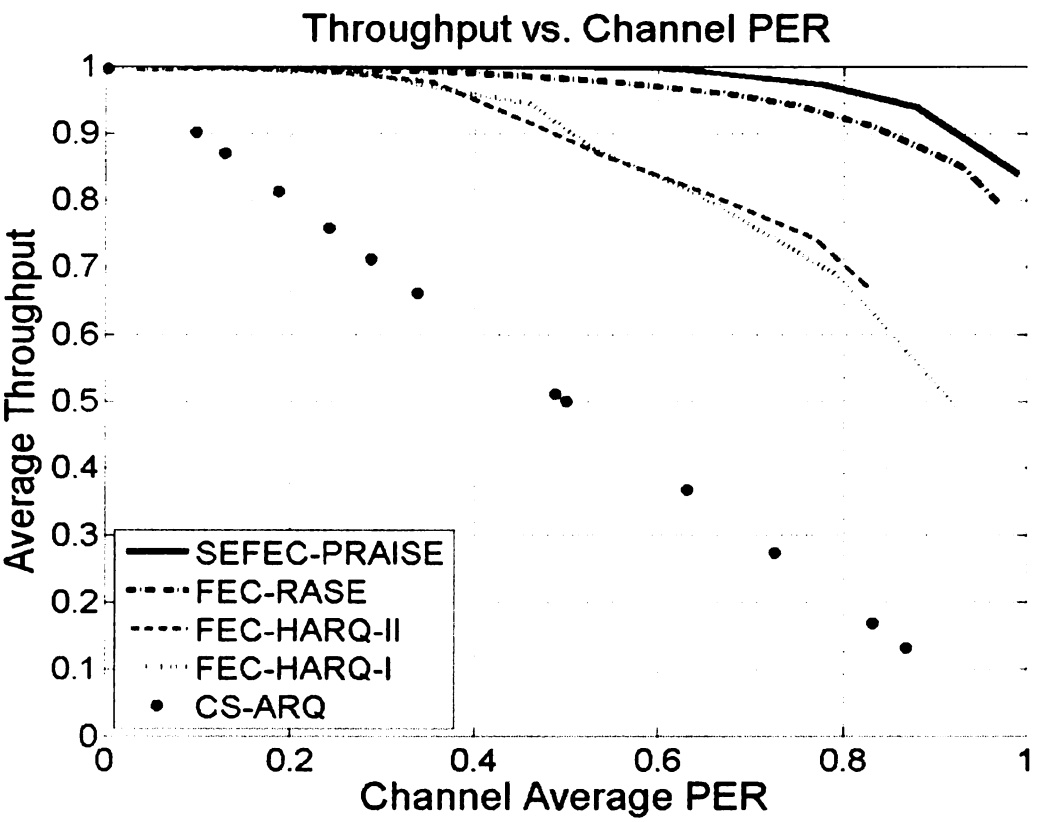

Figure VII-III: Throughput Vs. Channel PER

Figure VII-IV shows the plot for Goodput vs. Channel PER for all the five protocols. From plot below, we again see that the Conventional Standard (CS) ARQ suffers the most with increasing channel PER values. When the channel is in good condition (PER<0.1), CS-ARQ's performance is the best among all five protocols because in these scenario, goodput and throughput values are equal and all the data received is the '*useful*' data in case of CS-ARQ scheme. But as PER values increase, CS-ARQ's goodput drastically falls. Since the values of goodput and throughput are same in case of CS-ARQ, we see that the average goodput of the CS-ARQ scheme falls to 10% as channel's average PER increases to 0.85.

In FEC schemes, forward error correction bits are added to the existing ED bits to correct a subset of all errors while relying on ARQ to detect uncorrectable errors. As a result FEC-HARQ I/II perform better than CS-ARQ in poor signal conditions, but in its simplest form this comes at the expense of significantly lower goodput in good signal conditions. Here the signal quality cross-over point below which FEC-HARQ is better, and above which CS-ARQ is better is at PER = 0.2 in case of HARQ-II and at PER = 0.36 in case of HARQ-I. RASE makes use of incremental parity data to retrieve corrupted packets, resulting in increased channel bandwidth and goodput, and hence it performs better than HARQ-I and HARQ-II among FEC schemes. Here the signal quality cross-over point below which FEC-RASE is better, and above which CS-ARQ is better is at PER = 0.1.

It is evident from the plot that SEFEC-PRAISE outperforms FEC-HARQ I/II schemes in terms of goodput in all channel conditions. The signal quality cross-over point below which SEFEC-PRAISE is better, and above which CS-ARQ is better is at PER = 0.01. Since PRAISE needs additional $f = \left\lceil log_2 \binom{c}{l} \right\rceil$ feedback bits to request parity for l out of c chunks of lowest reliability, its performance in terms of goodput suffers insignificantly (due to small number of f bits) than that of FEC-RASE at lower PER values (PER<0.3). But as channel PER increases to 0.9, it outperforms state-of-art FEC-RASE by 0.2% higher goodput values.

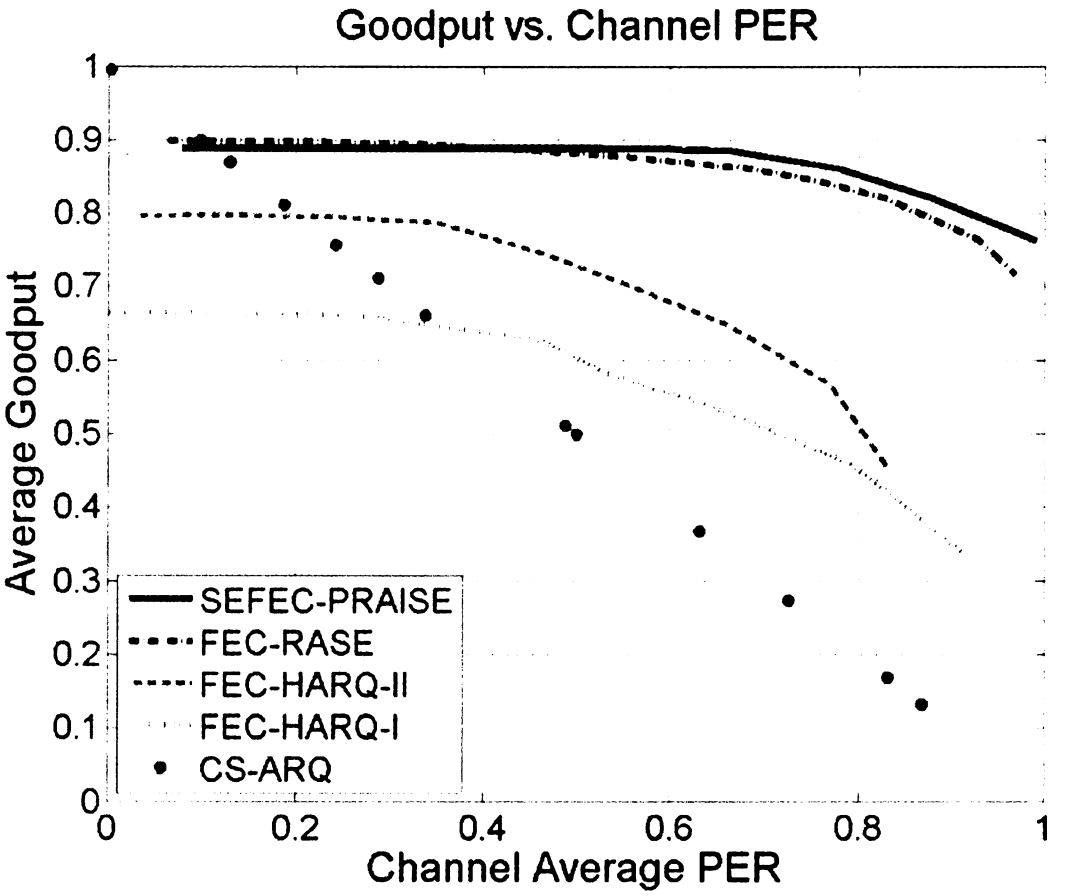

Figure VII-IV: Goodput Vs. Channel PER

7.3- Analysis of Results for Different Channel Metrics:

We explained our experimental setup in Chapter-5 of the thesis. Our experimental test-bed consists of two USRP devices: one connected to a desktop PC which we refer to as the *Data-Host (DH)* PC and the other connected to the *Data-Ghost (DG)* laptop computer. DH is hosting the data to be transmitted and DG is streaming data from DH. In our experiments, the application layer in DH and DG generates packet data and the acknowledgment packets respectively. We set up the wireless communication system between DH and DG using the parameters detailed in Chapter-5.

Using our GNU-USRP based wireless platform, we conducted several communication scenarios to capture the behavior of wireless error in the presence of fading, interference and mobility. We deployed a Markovian channel model to collect data, and details of the deployed channel model can be found in Chapter-4 of the thesis. Our analysis consists of measuring 100 wireless channel conditions which are categorized in four groups:

1- *Environment factor*: To cater for the environments effects on wireless channel conditions, we carried out extensive set of experiments in both, open-spaced out-door environment, i.e. outside the building, and in closed-space in-door environment, i.e. inside the building. For in-door setting, several factors contributed in wireless channel conditions as explained in coming lines. However, for out-door setting, the wireless communication was mostly line-of-sight (LOS) communication and we noticed that distance was the only metric for wireless connection failure and packet dropping.

2- *Fading factor*: In this set of scenarios, the DG laptop is located within certain distances of DH. The distance between DG and DH governs the impact of fading on the wireless communication. In particular, the error conditions for channels where DH and DG are relatively close to each other (e.g., less than 3m) are remarkably better than those with DG and DH are farther. It is also important to note that placing DG and DH very close to each other (e.g., less than ½ m) causes a lot of missing packet at DG due to interference caused by DH, so we did not consider such close distances.

3- *Interference factor*: To capture the impact of interference on wireless channel condition, we establish wireless communication in the presence of other wireless networks. These networks include Wi-Fi in our lab (i.e. WAVES lab), Wi-Fi in the building (i.e. College of Engineering), and in the presence of other hot-spots established over some laptop computers. It is evident that as more active wireless networks are presented in the environment, the likelihood of packet collisions increases leading to frequent packet errors due to interference.

4- *Mobility factor*: The DH transmits data packets to the DG laptop which is a mobile node and changes its location frequently. Please figure III-III to get an idea of the room locations. In this set of experiments, the mobility is conducted within a room (i.e. WAVES lab), within the hallway (next to the room where DH resides), and within the room opposite to hallway. However the wireless connection never fails; all the transmitted packets are received by DG.

We characterize wireless channel condition with average Bit Error Rate (BER) and average Packet Error Rate (PER). In figure VII-V, we compare these parameters (i.e. channel BER and channel PER) for the collected traces after deployment of a specific protocol, to get an insight on the results.

From Figure VII-V, we see that the range of corresponding channel average BER and average PER values lie in the range of 10% for all traces collected for the deployed protocols. The collected data reveals that even when the channel average PER approaches 0.8, the channel average BER value is 0.13, 0.07, 0.13, 0.14 and 0.07 in case

of SEFEC-PRAISE, FEC-RASE, FEC-HARQ-II, FEC-HARQ-I, and CS-ARQ respectively. These values present quite lower channel average BER values for the corresponding higher channel average PER values. This trend is platform and implementation specific and explains lower advantage of SEFEC-PRAISE (in terms of throughput and goodput) over FEC-RASE in terms of channel PER values as compared to channel BER values. We can also safely state that if we get higher BER values for corresponding PER values for some other platform/implementation, SEFEC-PRAISE's performance (in terms of throughput, goodput, and end-to-end bandwidth utilization) will get more boost as compared to the state-of-art FEC-RASE for both channel metrics, i.e. BER and PER values.

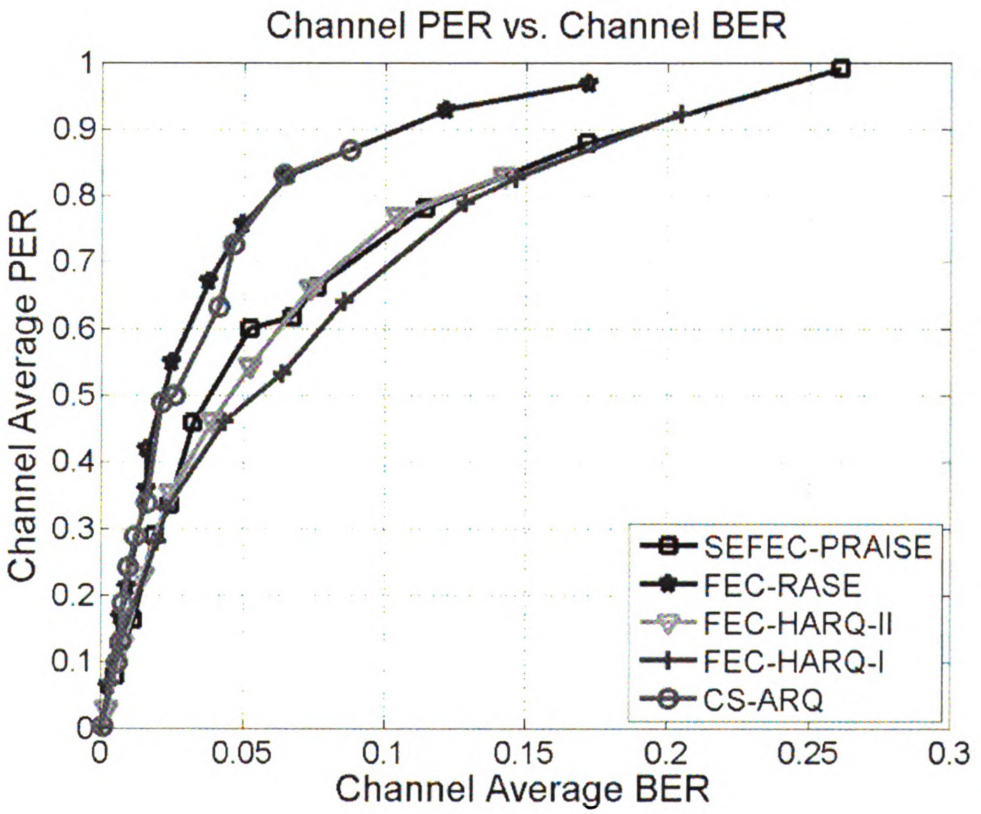

Figure VII-V: Channel BER vs. Channel PER

7.4 Conclusions and Future Directions:

In this work, we provided a framework for the rate adaptability of wireless protocols. Wireless channel is more prone to errors than wired media and we proposed localized puncturing patterns to adapt rates of wireless protocols with time-varying wireless channel conditions. First we developed the channel model that we used to develop the proposed scheme. We deployed a Markovian channel model to model the errors introduced in the channel. The Markovian channel is characterized by its Hybrid Erasure-Error Rate ($HEER$) in each state, and within each state, the channel is modeled by a cascade of a Binary Erasure Channel (BEC), to model the puncturing operation at encoder end, and a Binary Symmetric Channel (BSC) to model the wireless channel.

We used GNU-USRP based implementation of a *Software Defined Radio (SDR)* to conduct experiments. We divided our experimentation phase in two stages. In the first stage, we collected traces to train our Markovian channel model and to analyze the element of burstiness in a packet. We analyzed the element of burstiness using three standard techniques; 1) Error Distributions within a Packet; 2) Bit Error Rate (BER) vs. Cumulative Density Function (CDF); 3) End-to-End (E2E) Burst Length within Packets.

In the second phase of experiments, using our GNU-USRP based wireless platform, we conducted several communication scenarios to capture the behavior of wireless error in the presence of fading, interference and mobility. Our analysis in this phase consists of measuring 100 wireless channel conditions which are categorized in four groups; 1) Environment Factor; 2) Fading Factor; 3) Interference Factor; 4) Mobility Factor.

We then focus on code-designing aspects and present limits on designing localized puncturing fractions. We used Density Evolution techniques to find noise thresholds and extract degree distributions out of LDPC ensembles for given code rates and allowed degrees. Based on these results, we designed puncturing patterns for the proposed scheme.

We analyzed the performance of five network protocols; 1) Parity Localization for Rate Adaptability of Reliable and Stable Wireless Protocols (PRAISE), 2) Reliable and Stable (RASE), 3) Hybrid ARQ-II, 4) Hybrid ARQ-I, and 5) Automatic Repeat Request (ARQ) over GNU-USRP platform. To keep discussion generic, we divided these protocols in three categories; 1) Conventional Standard (CS, like ARQ), 2) Forward Error Correction enabled Protocols (FEC, like HARQ-I, HARQ-II, and RASE), 3) Side-Information Enhanced Forward Error Correction enabled Protocols (SEFEC, like PRAISE).

To gauge the performance of these protocols over the wireless channel, we make use of two protocol-dependent parameters namely, 1) throughput; 2) goodput. We compared the values of the above-mentioned protocol-dependent parameters against two wireless channel metrics, 1) Packet Error Rate (PER); 2) Bit Error Rate (BER). So our results fall in two categories depending upon wireless channel metrics and we measure each of the two protocol-dependent parameters against each of the metric. The results show that SEFEC-PRAISE outperformed state-of-art FEC-RASE by 0.9% and 0.2% higher goodput values as channel BER and channel PER increase to 0.15 and 0.9 respectively.

Similarly, SEFEC-PRAISE showed 12% and 0.4% higher throughput values than FEC-RASE as channel BER and channel PER increase to 0.15 and 0.8 respectively.

During the course of this work we identified some future directions. We will work on the development of a more comprehensive set of error traces. This set will comprise of the error traces collected for IEEE standard networking protocols (like IEEE 802.11, IEEE 802.15 etc.) at all bitrates for varying client positions and network configurations. The effects of environment, interference, fading and mobility will be further distinguishable with this complete data set. This is an active area of research and our preliminary analysis depicts that the position of the client plays an instrumental role in determining the overall throughput of the network particularly in line-of-sight out-door environments.

Moreover, cross- layer strategies that can enhance the side-information provided by the PHY/MAC layer in our case need further investigation. We would like to supplement these strategies with modeling and information extraction from link and transport layers without doing any modification at these layers to boost the overall performance of the system. We would also like to investigate information extraction from intermediate stages of Belief Propagation Algorithm while decoding LDPC codes.

In our previous work, we designed puncturing patterns based on online Density Evolution tools. We would like to further investigate other techniques to design puncturing patterns that can provide a suitable alternative for both, regular and irregular LDPC ensembles, and can improve end-to-end system performance.

References:

[1] C. E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, 1948.

[2] R. G. Gallager, "Low Density Parity-Check Codes", MIT Press, Cambridge, MA, 1963.

[3] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Analysis of Low Density Codes and Improved Designs using Irregular Graphs", in IEEE Transactions on Information Theory, 2001.

[4] M. Luby, M. Mitzenmacher, and A. Shokrollahi, "Analysis of Random Processes via And-Or Tree Evaluation", in Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, 1998.

[5] M. Yang, W. Ryan, and Y. Li, "Design of Efficiently-Encodable Moderate-Length High-Rate Irregular LDPC Codes", in IEEE Transactions on Communications, 2004.

[6] M. Yang, W. Ryan, "Lowering the Error Rate Floors of Moderate-Length High-Rate LDPC Codes", in IEEE International Symposium on Information Theory, 2003

[7] M. Yang, W. Ryan, Y. Li, "Design of Efficiently-Encodable Moderate-Length High-Rate LDPC Codes", in IEEE Transactions on Communications, 2003.

[8] T. Richardson and R. Urbanke, "The Capacity of Low-Density Parity Check Codes under Message-Passing Decoding", in IEEE Transactions on Information Theory, 2001.

[9] E. Eleftheriou, S. Olcer, "Low-Density Parity-Check Codes for Digital Subscriber Lines" in Proceedings of International Conference on Communications, 2002.

[10] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufmann Publishers, Inc., 1988.

[11] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes", in IEEE Transactions on Information Theory, 2001.

[12] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Efficient Erasure Correcting Codes", in IEEE Transactions on Information Theory, 2001.

[13] D. Mackay and M. Davey, "Evaluation of Gallager codes for Short Block Length and High Rate Applications", in Codes, Systems, and Graphical Models: Vol. 123 of IMA Volumes in Mathematics and its Applications, Spring-Verlag, 2000.

[14] B. Vasic, "Structured Iteratively Decodable Codes Based on Steiner Systems and their Application to Magnetic Recording", in Proceedings of IEEE GlobeCom Conference, 2001.

[15] P. Oswald and A. Shokrollahi, "Capacity-Achieving Sequences for the Erasure Channel", in IEEE International Symposium on Information Theory, 2002.

[16] I. Sason and G. Wiechman, "On Achievable Rates and Complexity of LDPC Codes for Parallel Channels: Information-Theoretic Bounds and Applications", in IEEE International Symposium on Information Theory, 2006.

[17] I. Sason and G. Wiechman, "On Achievable Rates and Complexity of LDPC Codes Over Parallel Channels: Bounds and Applications", in IEEE International Symposium on Information Theory, 2007.

[18] B. Vasic, "Combinatorial Constructions of Low-Density Parity-Check Codes for Iterative Decoding from Kirkman Triple Systems", in Proceedings of IEEE International Symposium on Information Theory, 2002.

[19] S. Johnson and S. Weller, "Construction of Low-Density Parity-Check Codes from Kirkman Triple Systems", in Proceedings of IEEE GlobeCom Conference, 2001.

[20] Y. Kou, S. Lin, and M. Fossorier, "Low-Density Parity-Check Codes based on Finite Geometries: A Rediscovery and New Results", in IEEE Transactions on Information Theory, 2001.

[21] D. MacKay, and R. Neal, "Good Codes based on very Sparse Matrices", in Lecture Notes in Computer Science, Springer, 1995.

[22] D. MacKay, "Good Error-Correcting Codes based on very Sparse Matrices", in IEEE Transactions on Information Theory, 1999.

[23] R. Lucas, M. Fossorier, Y. Kou, S. Lin, "Iterative Decoding of One-Step Majority Logic Decodable Codes based on Belief Propagation", in IEEE Transactions on Communications, 2000.

[24] D. Divsalar, H. Jin, and R. McEliece, "Coding Theorems for Turbo like Codes", in Proceedings of the Annual Allerton Conference on Communication, Control and Communication, 1998.

[25] H. Jin, A. Khandekar, and R. McEliece, "Irregular Repeat-Accumulate Codes", in Proceedings of 2nd International Symposium on Turbo Codes and Related Topics, 2000.

[26] J. Fan, "Constrained Coding and Soft Iterative Decoding for Storage", PhD thesis, Stanford University, 1999.

[27] J. Fan, "Array Codes as Low-Density Parity-Check Codes", in Proc. 2nd Int. Symposium on Turbo Codes and Related Topics, 2000.

[28] T Okamura, "A Hybrid ARQ Scheme Based on Shortened Low-Density Parity-Check Codes", in IEEE Wireless Communication and Networking Conference, 2008.

[29] J. Hagenauer, "Rate-compatible Punctured Cconvolutional Codes (RCPC codes) and their Applications" in IEEE Transactions on Communications, 1988.

[30] J. Ha, J. Kim, and S. McLaughlin, "Puncturing for Finite Length Low-Density Parity-Check Codes", in IEEE International Symposium on Information Theory, ISIT, 2004

[31] H. Pishro-Nik, and F. Fekri, "Results on Punctured LDPC Codes", in IEEE Information Theory Workshop, 2004.

[32] H. Pishro-Nik, and F. Fekri, "Results on Punctured Low-Density Parity-Check Codes and Improved Iterative Decoding Techniques", in IEEE Transactions on Information Theory, 2007.

[33] C. Hsu and A. Anastasopoulos, "Capacity Achieving LDPC Codes through Puncturing", in IEEE International Conference on Wireless Networks, Communications and Mobile Computing, 2005.

[34] S. Soltani, K. Misra and H. Radha, "On Link-Layer Reliability and Stability for Wireless Communication", IEEE Conference on Mobile Computing and Networking (MOBICOM 08), 2008.

[35] S. Karande and H. Radha, "The Utility of Hybrid Error Erasure LDPC (HEEL) Codes for Wireless Multimedia", IEEE International Conference on Communications (ICC), May 2005.

[36] J. Ha, J. Kim, and S. McLaughlin, "Rate-Compatible Puncturing of Low-Density Parity-Check Codes", in IEEE Transactions on Information Theory, November 2004.

[37] K. Lin, N. Kushman, and D. Katabi, "ZipTx: Harnessing Partial Packets in 802.11 Networks", in IEEE Conference on Mobile Computing and Networking (MOBICOM 08), 2008.

[38] S. Lin and P. Yu, "A Hybrid ARQ Scheme with Parity Retransmission for Error Control of Satellite Channels", in IEEE Transactions on Communications, 1982.

[39] Y. Wang and S. Lin, "A Modified Selective-repeat Type-II Hybrid ARQ System and its Performance Analyses", in IEEE Transactions on Communications, 1983.

[40] S. Soltani, K. Misra, A. Khan, and H. Radha, "On the Design and Implementation of a Reliable Wireless Link Layer Protocol with Guaranteed Sustainable Flows", (To be *Submitted).*

[41] L. Larzon, M. Degermark, and S. Pink, "UDP Lite for Real Time Multimedia Applications", in IEEE International Conference on Communications (ICC 99), June 1999.

[42] S. Karande and H. Radha, "Hybrid Erasure-Error Protocols for Wireless Video", in IEEE Transactions on Multimedia, February 2007.