This is to certify that the
thesis entitled

NETWORK CHANNEL CODING
IMPORTANCE OF IN-NETWORK PROCESSING
AND SIDE-INFORMATION

presented by

Shirish S Karande

has been accepted towards fulfillment
of the requirements for the

Doctoral degree in Electrical Engineering

Major Professor's Signature

12/13/07

Date

NETWORK CHANNEL CODING

IMPORTANCE OF IN-NETWORK PROCESSING AND SIDE-INFORMATION

By

Shirish S. Karande

A DISSERTATION

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

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering

2007

ABSTRACT

NETWORK CHANNEL CODING
IMPORTANCE OF IN-NETWORK PROCESSING AND SIDE-INFORMATION

By

Shirish S. Karande

Network Channel Coding (NCC) generically refers to a framework, under which we embed channel coding functionalities within a network, to improve the throughput in a multi-receiver communication scenario. In this work, we consider two types of NCC problems.

The first part is focused on NCC problems related to rateless codes. Rateless codes can be used for asynchronous broadcast applications. We focus our work on Luby-Transform (LT) codes, which are an example of low-complexity rateless codes. Prior designs of LT codes cater to the uniform demand case. In this work we design LT codes for non-uniform demands where the sinks are interested in recovering distinct fractions of the data symbols. Subsequently, we consider the design of Distributed LT (DLT) codes. DLT codes can be used in scenarios where multiple sources communicated to a sink via a common relay. In such a scenario, LT encoded data from various sources can be selectively mixed at the relay to improve the communication efficiency. Prior designs of DLT codes have focused on specific degree distributions, we generalize this work. We also comment upon the analysis of Generalized LT codes using *generalized ripples*.

The second part is focused on problems that study the impact of side-information on NCC. In particular we focus on design of NCC related cross-layer protocols. Typical wireless networks employ a link-abstraction under which packets with corruptions are discarded. While such an abstraction certainly leads to a simpler architecture, the capacity

loss on account of the packet drops can be significant. Hence, in this work we argue in favor of developing protocols which relay corrupted packet to higher layers, and across multiple hops. On the basis of theoretical deductions, model and trace based simulations; we analyze the utility of proposed protocols, particularly for multimedia applications. We investigate in detail the impact of receiver-side Channel State Information (CSI) on the performance of these protocols. We identify mechanisms that can provide CSI in commercially available wireless devices. Finally, we extend these protocols to exciting new area of network coding. Network coding saves bandwidth by mixing packets within the network. However, mixture of corrupt packets can lead to error amplification. Capacity loss due to error amplification can nullify the gains of network coding. To circumvent this phenomenon, we design an Adaptive Network Coding protocol that selectively mixes corrupt packets. The decision to mix is determined by the level of bit-corruption in each packet. All of the above proposed protocols exhibit a progressive improvement over comparable conventional protocols.

This work is dedicated to

my mother Sumeeta S. Karande and brother Shriniwas S. Karande

ACKNOWLEDGMENTS

I would like to thank my academic advisor, Professor Hayder Radha, for his guidance and help in refining my research ideas. His encouragement and insight have played a crucial role in my growth as a researcher. I am grateful to Professor Jonathan Hall, for his time and the mathematical training. I would also like to thank Professor Subir Biswas and Professor Tongtong Li for overseeing the completion of this dissertation. Feedback from a number of other ECE faculty members has been helpful and I thank all of them. I sincerely thank my colleagues from WAVES, without their collaboration and work spirit; my doctoral experience would have been incomplete. A number of friends have helped directly or indirectly in the completion of this work. A special mention should be made of Utpal, Kiran, Keyur, Akshay, Kantha, Yash-Ash, Shrini, Sunder, Roland, Keshav and Parmeshwar. I would like to thank my family (mom, Manish, Kavita, Asha maushi, Kiran mama and Rita maushi) for their love and support throughout my graduate study. Finally, I acknowledge the role of luck, or God almighty, in the completion of this work.

LIST

LIST

Part

A

A

A

A

A

Part

B.

B.

B.3

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figur

Figur
CON
param
perfor

Figur
show
the bl
maint

Figur
of Te
and ε

Figur
contir
distor
$\delta = 0$.

Figur
cross-

Figur

Figure

Figur

Figure

Figure

Figure

Figure

Figure

Figure
snapsh
with S

Figur
snaps
SSR l

Figur
snapsl
with S

Figure
snapsl
with S

Figure

Figure
mutual
in evol
employ

Figurel
parame
eventua
(N2) X
a better

FigureI

FigureI
and sta

FigureI
over F
m=SNP
plots w

PART A

ON DESIGN OF LT CODES

A.

1.1.

The

Heter

comm

above

distin

$\gamma \approx 1$

async

qualit

In t

thems

Luby

have

work

1.1.

The

k sou

a frac

LT d

A. 1. INTRODUCTION

1.1. Motivation and Overview

The two important issues that play a role in the design of broadcast protocols are (a) Heteregonity in the channel quality and (b) Asynchronous start and end to client-server communication. Rateless codes [1]-[4] provide an elegant and efficient solution for the above issues. In principle, these codes are able to generate a practically infinite number of distinct parity symbols from k message symbols, such that any γk parity symbols with $\gamma \approx 1$ being sufficient to recover all the message symbols. Thus a client can join asynchronously and still achieve near-optimal performance, irrespective of the channel quality vis-à-vis the other sinks in the network.

In this work, we focus on LT codes, which are an example of rateless codes that lend themselves to low-complexity encoding and decoding. LT codes, were first introduced by Luby in the seminal work [1]. Since, various generalizations and applications of the work have been considered (e.g. [2], [9], [12], [13]). The specific problems we consider in this work are as follows:

1.1.1. LT Codes for Non-uniform Demands

The original designs of LT codes were optimized for complete [1]-[2] recovery of the k source symbols. More recently, the intermediate performance of LT codes, where only a fraction z of the total k symbols are recovered, has also been investigated [9]. These LT distributions are optimized for a uniform demand case where all the receivers are

int

LT

dis

mu

1.1

a ra

help

to ty

appl

assoc

whic

corre

decoc

of the

case v

resear

(e.g. [

(DLT)

reduce

a comn

and the

Modifie

interested in downloading an identical fraction of the source data. In this work we design LT codes for the non-uniform demand case where the sinks are interested in recovering distinct fractions of the data symbols. These designs are motivated, for example, by multimedia applications, where the quality can be traded for bandwidth usage.

1.1.2. Generalized Distributed LT Codes

Random Linear Coding (RLC) can be utilized to code across multiple packet flows in a rateless ("elastic") fashion. In presence of packet losses, such inter-flow coding can help minimize the communication delay (e.g. [14]). However, under RLC, a receiver has to typically employ Gaussian Elimination (GE) to recover the message symbols. For applications, where the eventual receivers have limited computational capability, the cost associated with GE can be prohibitively high. Consequently, alternative approaches which lend themselves to low-complexity decoding are desirable. Rateless erasure correcting codes, such as the LT codes (e.g [1] lend themselves to low complexity decoding. However, in general, as packets get routed through the network, the "structure" of the LT codes cannot be preserved in a straightforward fashion. This is especially the case when intermediate nodes do not employ any decoding. Consequently, many recent research efforts have focused on realizing LT (or LT like) codes in a distributed fashion (e.g. [11]-[12]). In this work, we consider the problem of designing Distributed LT codes (DLT) as defined by Puducheri et. al. in [10]. Distributed LT (DLT) codes can be used to reduce the aggregate delay in networks where multiple sources communicate to a sink via a common relay. The DLT codes are used to independently encode packets at the sources, and these encoded packets are selectively XOR-ed at the relay to realize a composite Modified LT (MLT) code. The DLT codes and the XOR-ing rules are designed such that

the

feas

targ

our

optir

close

the p

provi

1.2.

This

prelim

Chapte

the MLT code closely approximates a target mother LT code. Prior work has shown the feasibility of realizing DLT codes with the Robust Soliton Distribution (RSD) [1] as the target. In this chapter we generalize this work for other target distributions. The key to our designs is a generalized multivariate representation of the target LT distribution. We optimize the design, such that the generalized representation of the MLT distribution closely approximates the Generalized Target LT (GTLT) distribution. A novel feature of the proposed method is the ability to facilitate the design of DLT codes capable of providing Unequal Recovery Time (URT).

1.2. Organization of this Part

This part consists of 4 chapters apart from the Introduction. Chapter 2 establishes the preliminaries. Chapter 3 is focused on design of LT codes for non-uniform demands. In Chapter 4 we generalize the notion of DLT codes.

A.

2.1.

We

LT F

proba

(a

(b

(

W

as th

I_y s

we

F

pol

ran

LT

fre

als

en

A. 2. BACKGROUND AND RELATED WORK

2.1. LT Codes: Encoding and Decoding

We provide a brief description of LT encoding/decoding. For details refer to [1]-[2].

LT Encoding: Given k input (or source/message) symbols $\{c_1,\cdots,c_k\} \in F(2)$ and a probability distribution β. An encoder generates an output (or parity) symbol y by :

(a) Selecting a degree d from $[1,k]$ according to β

(b) Selecting (at random) d distinct input symbols

(c) Adding these symbols in $F(2)$ to obtain an output symbol y.

With a slight abuse of notation we shall use y and c_i to represent the symbol as well as the value of the symbol. Moreover, with each symbol y, we associate an index set $I_y \subseteq [1,k]$, which is a collection of the indices of the input symbols used to form y. Thus, we can write $y = \sum_{c_i \in I_y} c_i$.

Finally, note that it is customary to represent the degree distribution β by a generator polynomial $\beta_x = \sum_d \beta_{x,d} x^d$ such that $\beta_{x,d}$ represents the probability of $|I_y| = d$, for a randomly chosen y.

LT Decoding: We assume that all the symbols recieved by the decoder are corruption free, that the decoder is aware of the index set associated with each recieved symbol and also that the decoding begins after the reciever has downloaded a desired number of γk encoding symbols.

Each encoding symbol is a linear combination of the input symbols. Thus the γk encoding symbols form a system of equations which can be solved to recover the original message. There exist a number of algorithms, with varying degree of complexity (e.g. [5]-[6]), to solve such a system of equations; however we focus our attention on the LT decoding algorithm proposed by Luby.

The LT decoding algorithm suggested by Luby is a Message Passing Algorithm that iteratively solves a sytem of equations. The decoding proceeds by progressively considering equations that get reduced to a single unknown. Thus, the decoding can be described as follows: The decoder maintains

(a) a list D of the indices of the decoded symbols

(b) a *ripple,* which is a list of output symbols satisfying $\left|I_y \cap D\right| = 1$. This is equivalent to identifying equations that have a single unknown in the reduced form. We refer to $\left|I_y \cap D\right|$ as the *reduced degree* of symbol y.

(c) a list U of output symbols satisfying $\left|I_y \cap D\right| > 1$.

At each step, the decoder removes one element y^* from the ripple to recover an input symbol c^*, by using the reduction $c^* = y^* - \sum_{i \in I_{y^*} \cap D} c_i$. The lists D, U and *ripple* are updated accordingly. Thus,

(a) the index of c^* is added to D.

(b) All encoding symbols satisfying $I_y - D = I_{y^*} - D$ are removed from the ripple.

Typically, multiple encoding symbols could be associated with the same recovered

(c)

The

recov

2.2

Th

the

sect

info

W

us

of

w

ar

symbol c^*. These symbols are said to *defect* from the ripple.

(c) After D has been updated, some encoding symbols from U may satisfy the condition $|I_y \cap D| = 1$, these symbols are removed from U and are added to the ripple. We say that these symbols *arrive* into the ripple.

The decoding stops when the ripple becomes empty or when all input symbols are recovered.

2.2. Analysis of the Ripple-size

The LT decoding stops when the ripple-size becomes zero. Thus the characterization of the ripple-size plays an important role in the design and analysis of LT codes. In this section we provide a brief sketch of recent analysis by Hajek [8]. For additional information and a more rigorous analysis please refer to [7]-[9].

We say that an LT decoder is in state v, when $|D| = v$ symbols have been recovered. Let us denote by X_v the size of the ripple when the decoder is in state v. Thus, the evolution of the ripple-size can be represented by the following recurrence relationship:

$$X_{v+1} = X_v - 1 - L_v + A_v \tag{1}$$

where, the arrival rate A_v represents the number of encoding symbols joining the ripple and $L_v + 1$ represents the number of symbols removed from the ripple. Note that, both A_v and L_v are random variables.

Defect Rate:

Let us denote by $v+1$ the input symbol recovered, when the decoder transits from state v to state $v+1$. Let us say that this symbol has been recovered by removing the symbol

y_v from the ripple. Now, let's consider an encoding symbol y from the remainder of the ripple. This symbol is connected to exactly one and any one of the $k-v$ input symbols, unrecovered untill state v. Thus, the probability of y being connected to input symbol $v+1$ is equal to $\frac{1}{k-v}$. Consequently, we can conclude that L_v is governed by the binomial distribution $B\left(X_v-1,\frac{1}{k-v}\right)$.

Arrival Rate:

To calculate the arrival rate, consider the event that causes an encoding symbol to join the ripple. Figure 1 represents such an event.

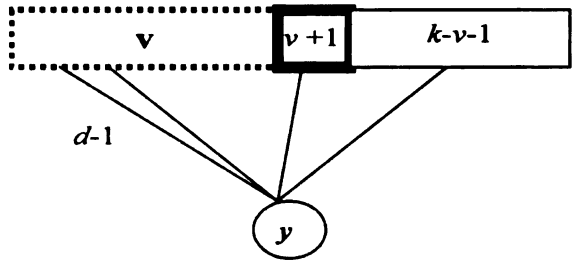


Figure A.1 The event that causes an encoding symbol y to join the ripple.

As shown in Figure 1, a randomly chosen encoding symbol y of degree d (> 2) joins the ripple, if and only if, it is connected to (d-1) recovered input symbols, the symbol $v+1$ and a symbol from the $k-v-1$ input symbols that remain unrecovered at state $v+1$. Thus,

$$p\left(y\ joins\ ripple \middle| \deg(y)=d, state=v\right) \approx d(d-1)\left(\frac{v}{k}\right)^{d-2}\cdot\left(\frac{1}{k}\right)\cdot\frac{1}{k-v-1} \qquad (2)$$

Thus, if total number of encoding symbols received by a decoder are γk, the expected arrival rate is given by:

$$\overline{A_v} = \gamma k\left(\sum_d\left(\beta_d \cdot d(d-1)\left(\frac{v}{k}\right)^{d-2}\cdot\left(\frac{1}{k}\right)\cdot\frac{k-v-1}{k}\right)\right) = \gamma\left(1-\frac{v+1}{k}\right)\beta^{"}(v/k) \quad (3)$$

where, $\beta^{"}$ is the second derivative of the degree polynomial.

Now, if we represent $(v/k)=t$ then (1) can be re-written as follows:

$$\lim_{k\to\infty}\frac{(X_{v+1}-X_v)/k}{1/k} = x'(t) = \lim_{k\to\infty}A_v - 1 - \lim_{k\to\infty}L_v \quad (4)$$

where, the ripple size is given by $k\cdot x(t)$. Now, note that

$$\lim_{k\to\infty}A_v = (1-t)\beta^{"}(t) \text{ and } \lim_{k\to\infty}L_v = \lim_{k\to\infty}\frac{(X_v-1)/k}{(k-v-1)/k} = \frac{x(t)}{(1-t)}$$

Thus (4) can be re-written as the following differential equation.

$$x^{'}(t) = \gamma(1-t)\beta^{"}(t) - 1 - \frac{x(t)}{(1-t)} \quad (5)$$

Finally, note that at the start of the decoding, the ripple-size is completetely determined by the degree one symbols. Thus, it can be verfied that the following expression satisfies the above differential equations and the necessary initial conditions:

$$x(t) = (1-t)\left(\gamma\beta^{'}(t) + \ln(1-t)\right) \quad (6)$$

For decoding to continue, the condition $x(t)>0$ has to be satisfied. Thus, the above analysis implies the following theorem:

Theorem 1 [7]: As $k\to\infty$, if the number of LT encoding symbols received by the decoder are $Poisson(\gamma k)$, then the an LT decoder is able to recover a fraction z of data symbols if and only if $\forall t\in[0,z)\ \ \gamma\beta^{'}(t)+\ln(1-t)>0$.

A.

3.1.

LT C

of LT

interm

are re

unifo

fracti

In

moti

band

z_i ,

by

requ

of

sin

η_i

qu

qu

A. 3. LT CODES FOR NON-UNIFORM DEMANDS

3.1. Motivation

LT Codes ([4]) can be used for asynchronous content distribution. The original designs of LT codes were optimized for complete [1]-[2] recovery of the k source symbols. The intermediate performance of LT codes, where only a fraction z of the total k symbols are recovered, has also been investigated [9]. These LT distributions are optimized for a uniform demand case where all the receivers are interested in downloading an identical fraction of the source data.

In this letter, we design LT codes for the non-uniform demand case. These designs are motivated, for example, by multimedia applications, where the quality can be traded for bandwidth usage. The quality that a sink i demands is proportional to the fraction of data, z_i, that it wishes to download. A sink's normalized bandwidth usage can be represented by *stretch* γ_i, such that $\gamma_i \cdot k$ represents the average number of LT encoded symbols required to recover the fraction z_i. It is important to note that, in general, different values of (desired) z require different values of (needed) γ; where $z \leq \gamma$. Thus a particular sink's download efficiency can be measured in terms of the *overhead* $\gamma_i - z_i$ or the *rate* $\eta_i = z_i / \gamma_i$.

In practice, the amount of time a sink wishes to remain connected, and hence the quality he demands can vary significantly. Thus we seek to address the following questions:

a. When

sink's ba

b. How c

fair fash

We ad

asympt

analys

averag

The

In se

analy

3.2

3.2

eac

de

fo

th

a

a. When can a single LT distribution satisfy all non-uniform demands (z_i) given each sink's bandwidth constraint (γ_i) ?

b. How can an LT distribution be designed to distribute the inefficiency in download in a fair fashion ?

We address (a) by identifying the conditions for a set of *rate-demands* to be asymptotically feasible under LT coding. These conditions directly follow from the analysis in [7]-[9]. To address (b) we determine cost-optimized LT codes, based on an *average-rate*, a *min-rate* and a *max-overhead* criterion.

The remainder of this chapter is organized as: Section 3.2 establishes the preliminaries. In section 3.3 we formulate the problems. Section 3.4 provides numerical results and analysis.

3.2. Preliminaries

3.2.1. Network Model

We consider a network with a single broadcast source S and m sinks $T_1 \cdots T_m$. In each time-slot the source broadcasts an LT encoded symbol obtained by employing a pre-determined degree distribution β. A sink T_i has a demand z_i and remains connected for an average of $\gamma_i k = (z_i/\eta_i)k$ time slots. The sinks connect asynchronously [2] and thus the exact slots for which a sink is connected are not known to the transmitter. When a sink is connected, it receives the transmissions without any corruption.

3.2.2. Asymptotically feasible rate-demands

Definition 1: A *rate-demand* tuple $(\bar{\eta}, \bar{z})$, where vector $\bar{z} = [z_i]_{i=1}^{m}$ represents the sink demands and $\bar{\eta} = [\eta_i]_{i=1}^{m}$ represents the rate constraints, is said to be *asymptotically feasible* if there exists an LT degree distribution β such that: As $k \to \infty$, if the number of encoding symbols received by sink T_i are *Poisson* $(z_i/\eta_i)k$, then the fraction of input symbols recovered by the sink tend to v_i, where $v_i \geq z_i$.

Theorem 2: A *rate-demand* tuple $(\bar{\eta}, \bar{z})$ is asymptotically feasible iff there exists a degree distribution $\beta(x)$ such that:

$$(z_i/\eta_i)\beta'(t) + \ln(1-t) > 0 \quad \forall t \in [0, z_i), \ \forall i \in [1.m] \tag{9}$$

where $\beta'(t)$ is the derivative of the distribution $\beta(\cdot)$ at t.

Proof: The proof for the above theorem follows directly from Theorem 1.

3.2.3. Feasible rate-demands for finite k

The constraints specified by Theorem 2 can be used to deduce bounds on the performance of LT codes [9] as $k \to \infty$. However, for finite values of k, the random fluctuations in the ripple size need to be accommodated. To accommodate such fluctuations, the desired ripple size is often bounded by a value greater than zero. In this work we utilize Shokrollahi's heuristic in [2], which leads to the constraint:

$$x(t) > \mu_k(t) = c\sqrt{(1-t)/k} \tag{10}$$

The above discussion motivates the following definition:

Defin

degre

where

$\bar{\eta}$ is f

3.3.

In

proble

Prob

In thi

The

z_i. Thus

Definition 2: A *rate-demand* tuple $\left(\bar{\eta},\bar{z}\right)$, is said to be *(k-)feasible* if there exists an LT degree distribution β s.t. $\forall i \in [1,m]$, $\forall t \in [0,z_i)$ we have $\beta'(t)+(\eta_i/z_i)\cdot\tilde{\mu}_k(t)\geq 0$, where $\tilde{\mu}_k(t)=\left(\ln(1-t)-(\mu_k(t)/1-t)\right)$. For a demand $\bar{z}$, we say that a rate allocation $\bar{\eta}$ is feasible iff $\left(\bar{\eta},\bar{z}\right)$ is feasible.

3.3. Cost-Optimized LT Codes

In this section we design LT codes by associating a cost to the rate-allocation. This problem of designing cost-optimized LT codes can be generically stated as follows:

Problem 1 (P1): $$\left(\beta^*,\bar{\eta}^*\right)=\arg\min_{\beta,\bar{\gamma}}\left(Q\left(\bar{\eta}\right)\right)$$

$$\textit{subject to}\quad \sum\beta_j=1,\ \forall j\ \beta_j\in[0,1],\ \forall i\ \eta_i\in[0,1]\quad\text{and}$$

$$\forall i\in[1.m],\ \forall t\in[0,z_i)\quad \beta'(t)+(\eta_i/z_i)\cdot\tilde{\mu}_k(t)\geq 0 \tag{11}$$

In this work, we focus on the following costs (or criteria):

$$\textbf{Optimal Single Rate:}\quad Q\left(\bar{\eta}\right)=-\eta_i \tag{12}$$

$$\textbf{Average-Rate:}\quad Q\left(\bar{\gamma}\right)=-\sum_i\eta_i \tag{13}$$

$$\textbf{Minimum-Rate:}\quad Q\left(\bar{\eta}\right)=\max_i\{-\eta_i\} \tag{14}$$

$$\textbf{Maximum-Overhead:}\quad Q\left(\bar{\eta}\right)=\max_{\eta}\left\{\left((\eta_i/z_i)-z_i\right)\right\} \tag{15}$$

The cost (12) leads to an LT distribution that maximizes rate η_i for a single demand z_i. Thus under cost (12), for $k\to\infty$, P1 is equivalent to the Linear Program (LP) in [9].

The rate η can be interpreted to represent the information recovered per received symbol. Thus cost (13) can be utilized to maximize the average information received per symbol, over the entire network. It can be easily verified that under cost (13) P1 reduces to a LP.

The average-rate can be traded for achieving a more uniform rate distribution. Cost (14) can be utilized to maximize the minimum rate (R) experienced by a sink. The following LP solves P1 under cost (14):

Problem 2 (P2): $\left(\beta^*, R^*\right) = \arg\min_{\beta,R}(-R)$

subject to $\sum \beta_j = 1,\ \forall j\ \beta_j \in [0,1],\ R \in [0,1]$ and

$$\forall i \in [1.m],\ \forall t \in [0, z_i)\quad \beta^{'}(t) + (R/z_i)\cdot \mu_k(t) \geq 0 \tag{16}$$

Thus one optimal rate allocation is $\eta_i^* = R^*$. In general, the β^* can achieve higher rates for some sinks. In section 3.5, we comment further on this topic.

The excess bandwidth used for demand z_i or the excess delay experienced by sink i is given by $\gamma_i - z_i$. Thus cost (15) can be utilized to minimize the maximum excess delay experienced by a receiver. We obtain the LT distribution that minimizes cost (15) by solving the following problem:

Problem 3 (P3): $\left(\beta^*, \Delta\right) = \arg\min_{\beta,\Delta}(\Delta)$

subject to LP-A being a feasible linear program, where

LP-A: $\left(\beta^*\right) = \arg\min_{\beta}\left(\sum \beta_j\right)$

subject to $\sum \beta_j = 0,\ \forall j\ \beta_j \in [0,1]$ and

$$\forall i \in [1,m],\ \forall t \in [0,z_i)\quad (z_i + \Delta)\beta'(t) + \mu_k(t) \geq 0 \tag{17}$$

P3 can be solved by a simple bi-section search for the smallest value of Δ such that LP-A is feasible.

The solutions obtained by minimizing costs (13)-(15) are respectively termed as Minimum Average Rate (MAR), Maximum Min-Rate (MMR) and Minimum Max-Overhead (MMO) criterion rate allocation.

3.4. Numerical Results

In this section we present results obtained by numerical optimization for $k = 1000$. Note that, since t is a continuous variable the optimization problems consist of an infinite number of constraints. Hence, to obtain a numerical solution we reduce the constraints to a finite number by sampling t [2].

We evaluate the performance of a distribution β in terms of the function $\gamma_\beta(z)$, which represents the average stretch required to recover a fraction z of the symbols. The analysis in the previous sections can be used to deduce $\gamma_\beta(z)$ as:

$$\gamma_\beta(z) = \min_a\ s.t.\ \left\{ \forall t \in [0,z),\ a\beta'(t) + \mu_k(t) \geq 0 \right\} \tag{18}$$

We also evaluate $\gamma_\beta(z)$ on the basis of actual experiments. Figures 2 and 3 present our evaluations. Also note that we have heuristically set the parameter c to 0.6. This value was empirically observed to provide reasonably accurate prediction of the performance of

LT codes. However, improved results could be obtained by using evolutionary optimization techniques [17].

The single demand case has been extensively studied in [9]. However the results in [9] are applicable for the asymptotic case where $k \to \infty$. Therefore, for a finite k, we re-evaluate the optimal performance under cost (5). Under cost (5), the optimal distribution varies with the value of the single demand. Thus, in Figures 2 and 3 the "Optimal Single Demand" curve is obtained from the convex-hull of the performance of individual LT distributions. This curve serves as a bench-mark for the remainder of our designs.

For costs (13)-(15) we consider a non-uniform demand $\bar{z} = [0.24, 0.39, 0.54, 0.69, 0.84, 0.99]$ for six sinks. The optimal LT distributions for each cost are listed below:

$$\beta_{MAR}(x) = 0.028x^1 + 0.932x^2 + 0.002x^{46} + 0.038x^{47} \tag{19}$$

$$\beta_{MMR}(x) = 0.633x^1 + 0.188x^2 + 0.121x^6 + 0.025x^7 + 0.024x^{46} + 0.01x^{47} \tag{20}$$

$$\beta_{MMO}(x) = 0.182x^1 + 0.658x^2 + 0.038x^8 + 0.087x^9 + 0.025x^{52} + 0.01x^{53} \tag{21}$$

Given a set of non-uniform demands, a single LT distribution cannot simultaneously provide the best possible performance to each sink. However, Figures 2 and 3 show that the intermediate performance of a single LT distribution can be modulated significantly to optimize a particular performance objective. From Figure 3, we observe that the β_{MAR} distribution is able to achieve an average rate greater than 0.64, the β_{MMR} distribution is able to provide a minimum rate of 0.58 for each demand and the β_{MMO} distribution is able to maintain the worst case overhead to be under 0.37.

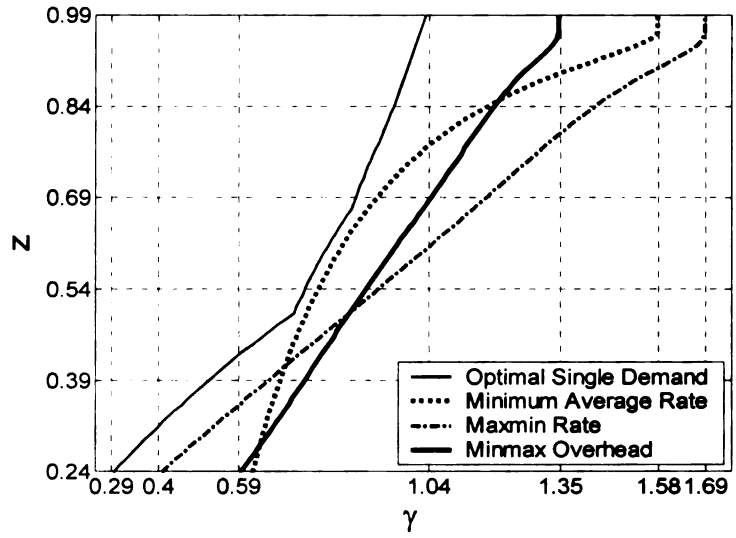


Figure A.2 Theoretically predicted performance in terms of the expected stretch γ required to recover a fraction z.

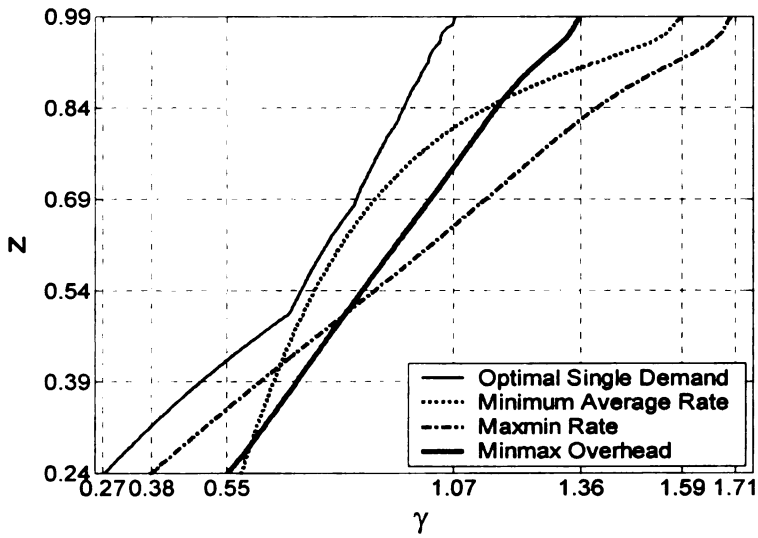


Figure A.3 Experimentally evaluated value of the mean stretch γ.

3.5. Discussion

The designs of broadcast codes often seek to minimize the *regret* experienced by each sink (see [13]). If we represent by a_i^* the least stretch required to satisfy a demand z_i (without regard to other demands), then the regret can be defined as the difference $\gamma_\beta(z_i) - a_i^*$, which describes the excess inefficiency that has to be endured on account of sharing the resources. Figure 3 demonstrates that, under β_{MMO}, the maximum regret is less than 0.29. It is natural to inquire, whether significant improvements could be obtained by explicitly optimizing the degree distribution to minimize the maximum regret. P3 can be modified to identify such a *Min-Regret* (MR) distribution, β_{MR}. However, the performance difference between β_{MMO} and β_{MR} was observed to be negligible.

Finally it is noteworthy, that under costs (14)-(15), P1 does not have a unique optimal solution. For example, with reference to cost (14), there exist multiple LT distributions that support minimum rate of R^*. In general, we can impose an ordering on the set of optimal distributions by choosing to further increase the rate for a subset of the sinks while ensuring that the minimum rate remains greater than R^*. Such an approach is similar to the iterative water filling algorithm used for minmax rate allocation [15]. We were successful in adopting the water-filling algorithm [15]. The modified water-filling algorithm leads to a sequence of LPs, which can be solved to identify a unique optimal LT distribution. However, we observed that performance of such an LT distribution was only marginally better than the solution to P2.

A. 4. GENERALIZED DISTRIBUTED LT CODES

4.1. Motivation

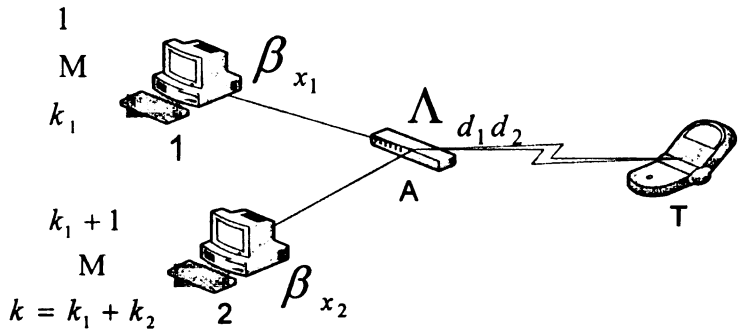


Figure A.4 Network consisting of two sources [1,2], a relay A and a sink T. Network operation is governed by β_{x_1}, β_{x_2} and Λ_{d_1,d_2}.

In this work, we consider the problem of designing Distributed LT codes (DLT) as defined by Puducheri et. al. in [5]. DLT codes can be used in network topologies where multiple sources communicate distinct data segments to a single sink via a common relay, as shown in Figure 4. In Figure 1 interflow coding at the relay can be used to increase the time-diversity. If the link between the relay and the sink is assumed to be lossy, then the increased diversity can provide performance benefits. DLT codes can be used to achieve exactly this time-diversity gain, albeit in a scenario where each source transmits LT encoded data and the sink employs LT decoding. Thus, the DLT encoded packets[1] are selectively mixed at the relay to present a Modified LT code (MLT) to the sink. The DLT

[1] For simplicity, we assume that all symbols are binary and packets consist of as single bit. The terms bit, symbol and packet are used interchangeably.

codes and the mixing rules at the relay are designed such that the MLT distribution represents an efficient LT code over the concatenation of all sources

DLT codes can be designed by using well known LT distributions as targets for the MLT codes. Consequently, the DLT codes employed at the source can be thought of as a decomposition of the Target LT (TLT) code. In [10] this intuition has been used to motivate a deconvolution based design of DLT codes. Such an approach has been shown to provide an elegant analytical decomposition when the TLT code is described by a Robust Solition Distribution (RSD) [1]. In practice it may often be desirable to realize LT distributions other than RSD, for example: the LT distributions suggested by Shokrollahi [2] could be used to design DLT codes capable of facilitating linear-time encoding/decoding. Hence, in this chapter, we seek to generalize the concept of DLT codes for target distributions other than RSD.

The core contribution in our design approach emerges from the manner in which we choose the DLT target degree distribution. In [10], a single variable target distribution is utilized. A single variable distribution cannot suitably represent the inter-segment mixing that a TLT code naturally achieves. We argue that the structure of a TLT code over multiple data segments can be captured more appropriately by a generalized multi-variable degree distribution $\beta(x_1,\cdots,x_m) = \sum \beta_{\{d_1,\cdots,d_m\}} x_1^{d_1} \cdots x_m^{d_m}$, where each variable corresponds to a source/segment. We optimize the design of DLT codes and the "mixing rules" so that the Generalized MLT (GMLT) distribution provides an accurate polynomial approximation of the multivariate target distribution. We utilize the above approach to obtain DLT codes from the LT distributions presented in [2]. The above described approach provides flexibilities which were not feasible with the methods

proposed in [10]. For example, the designs in [10] are applicable to the special case where all sources have equal priority and size. The multivariate degree distribution we employ can be used to represent rateless codes capable of providing Unequal Recovery Time (URT). We exploit this flexibility to design DLT codes over segments with unequal sizes and priorities.

The remainder of this chapter is organized as follows: In Section 4.2 we describe the system model. We formulate the problem in Section 4.3. The numerical results are presented in Section 4.4. We discuss in Section 4.5.

4.2. System Model

We employ a network model similar to the one used in [10]. We primarily concentrate on the network described by Figure 4. The network consists of two sources [1, 2], a common relay A and sink T. We assume that the communication takes place in terms of epochs. Each epoch consists of an LT encoded transmission from each source to the relay A, followed by a single transmission from the relay to the sink T. We assume that the source-to-relay links are lossless while the relay to sink link is assumed to be lossy. The losses on link A-T are assumed to be independent of the data being transmitted.

The input symbols [1, k] are segregated into two disjoint segments. Thus source 1 communicates symbols [1, k_1], while source 2 communicates symbols [k_1+1, k_1+k_2]. The coding at the sources is determined by the DLT distributions $\beta_{x_1} = \sum_{d_1=1}^{k_1} \beta_{x_1,d_1} x_1^{d_1}$,

$\beta_{x_2} = \sum_{d_2=1}^{k_2} \beta_{x_2,d_2} x_2^{d_2}$.

The relay is assumed to have storage capacity of one symbol (or packet) per source and processing capability that is limited to simple XORs. At each epoch the relay receives a packet from the individual sources. The relay probabilistically chooses to either forward one of the received packets or to transmit an XOR-ed combination of the received packets. Upon transmission the relay discards the received packets. The probabilistic decisions taken by the relay can be described by "mixing rules" $\Lambda_{d_1 d_2} = \left\{ \Lambda_{d_1 d_2,\{1\}}, \Lambda_{d_1 d_2,\{1,2\}}, \Lambda_{d_1 d_2,\{2\}} \right\}$, where each rule represents a conditional probability distribution:

Mixing Rule $\Lambda_{d_1 d_2}$: Given degrees d_1, d_2 of output symbols transmitted by source 1 and 2 resp.,

$\Lambda_{d_1 d_2,\{1\}}$ = *probablity of forwarding packet from source* 1

$\Lambda_{d_1 d_2,\{1,2\}}$ = *probablity of transmitting an X-OR* where

$\Lambda_{d_1 d_2,\{2\}}$ = *probablity of forwarding packet from source* 2

$$1 \geq \Lambda_{d_1 d_2,\{1\}}, \Lambda_{d_1 d_2,\{1,2\}}, \Lambda_{d_1 d_2,\{2\}} \geq 0, \quad \Lambda_{d_1 d_2,\{1\}} + \Lambda_{d_1 d_2,\{1,2\}} + \Lambda_{d_1 d_2,\{2\}} = 1 \qquad (22)$$

The relay is said to employ a switching policy if the mixing rules satisfy the condition that $\Lambda_{d_1 d_2,\{1\}} = p$, $\Lambda_{d_1 d_2,\{1,2\}} = 0$, $\Lambda_{d_1 d_2,\{2\}} = 1-p$, in all other cases we say that the relay employs a mixing policy.

The above described network model can be extended to more than two sources. The operation at the sources and the sink generalize to multiple sources in a straight-forward fashion, however, the mixing rules at the relay do not. As in [10], we define the mixing

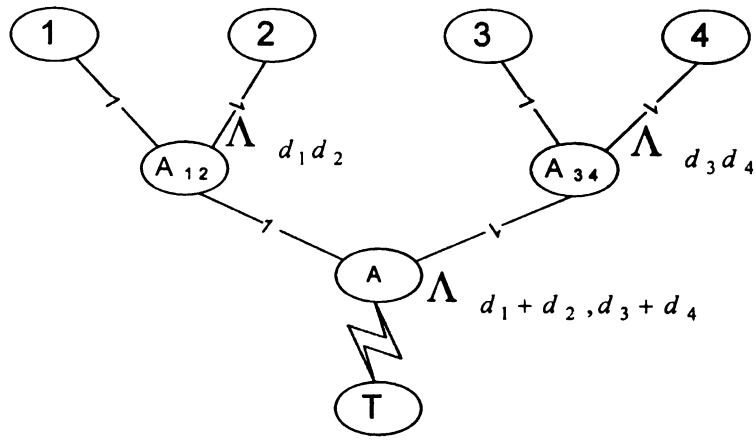


Figure A.5 Hierarchical mixing for a network consisting of 4 sources.

for multiple sources in a hierarchical fashion. Thus in the four source case, as illustrated in Figure 5, we create virtual intermediate relay nodes A_{12} and A_{34}. We employ "two-source" mixing rules $\Lambda_{d_1 d_2}$, $\Lambda_{d_3 d_4}$ at these virtual nodes. The output of these virtual nodes is subsequently exchanged by a higher level rule $\Lambda_{d_1+d_2, d_3+d_4}$. Note that the operation of a higher level rule is controlled by the sum-degree of the symbols emerging from lower exchanges at nodes A_{12} and A_{34}.

4.3. Problem Formulation

4.3.1. Evaluating the MLT distribution

The performance of the system described in the previous section is determined by the composite MLT distribution which describes the output symbols transmitted by the relay. This composite degree distribution can be described in two different ways: We could define the composite distribution in terms of the sum degree $(d_1 + d_2)$ or, in terms of the tuple (d_1, d_2), where (d_1, d_2) is referred as the generalized degree of an encoding symbol. To be consistent with [10] we refer to the sum-degree distribution as the MLT

distribution and to differentiate we refer to the tuple-degree distribution as the Generalized MLT (GMLT) distribution. For a given set of source distributions and mixing rules: $\{\beta_{x_1}, \beta_{x_2}, \Lambda_{d_1 d_2}\}$; the MLT and GMLT distributions can be deduced as:

$$\text{MLT:} \quad \beta_x = \sum_{d=1}^{k_1+k_2} \beta_{x,d} x^d \quad \text{where,} \tag{23}$$

$$\beta_{x,d} = \sum_{\substack{d_1,d_2 \\ s.t.\ d_1+d_2=d}} \left(\beta_{x_1,d_1} \cdot \beta_{x_2,d_2} \cdot \Lambda_{d_1,d_2,\{1,2\}}\right) + \sum_{d_1=d,d_2} \left(\beta_{x_1,d_1} \cdot \beta_{x_2,d_2} \cdot \Lambda_{d_1,d_2,\{1\}}\right) + \sum_{d_1,d_2=d} \left(\beta_{x_1,d_1} \cdot \beta_{x_2,d_2} \cdot \Lambda_{d_1,d_2,\{2\}}\right)$$

$$\text{GMLT:} \quad \beta_{(x_1,x_2)} = \sum_{\substack{k_1 \geq d_1 \geq 0, k_2 \geq d_2 \geq 0 \\ d_1+d_2 \geq 1}} \beta_{(x_1,x_2)(d_1,d_2)} x_1^{d_1} x_2^{d_2} \quad \text{where}$$

$$\begin{aligned} &\textit{If } d_2 = 0, \quad \beta_{(d_1,0)} = \sum_d \left(\beta_{x_1,d_1} \cdot \beta_{x_2,d} \cdot \Lambda_{d_1,d,\{1\}}\right) \\ &\textit{If } d_1, d_2 > 0, \quad \beta_{(d_1,d_2)} = \beta_{x_1,d_1} \cdot \beta_{x_2,d_2} \cdot \Lambda_{d_1,d_2,\{1,2\}} \\ &\textit{If } d_1 = 0, \quad \beta_{(0,d_2)} = \sum_d \left(\beta_{x_1,d} \cdot \beta_{x_2,d_2} \cdot \Lambda_{d,d_2,\{2\}}\right) \end{aligned} \tag{24}$$

4.3.2. Design of DLT codes using norm-approximation

DLT codes are designed by minimizing the difference in the MLT distribution and an appropriately chosen Target LT distribution (TLT) Ω_x. In this work, we measure the difference between the two degree distributions in terms of the sum of the square of the difference between the LT distribution coefficients. Thus the design method adopted in [10] can be considered to be equivalent to the following:

Design1 (D1): $\left(\beta_{x_1}^*, \beta_{x_2}^*, \left\{\Lambda_{d_1,d_2}^*\right\}\right)^* = \underset{\left(\beta_{x_1}, \beta_{x_2}, \left\{\Lambda_{d_1,d_2}\right\}\right)}{\arg\min} \left(\sum_{d=1}^{k_1+k_2} \left(\beta_{x,d} - \Omega_{x,d}\right)^2 \right)$ (25)

The drawback of above approach can be described by considering a simple example:

Example 1: Let, $\Omega_{x,d} = 0.9x + 0.1x^2$, then the following describe two non-unique optimal solutions for D1.

(a) $\beta_{x_1} = x_1$, $\Lambda_{d_1d_2,\{1\}} = 0.9$, $\Lambda_{d_1d_2,\{1,2\}} = 0.1$, $\Lambda_{d_1d_2,\{2\}} = 0$

(b) $\beta_{x_2} = x_2$, $\Lambda_{d_1d_2,\{1\}} = 0$, $\Lambda_{d_1d_2,\{1,2\}} = 0.1$, $\Lambda_{d_1d_2,\{2\}} = 0.9$

In both the above solutions, the sources employ a trivial *all-one* LT code. The solutions differ in the mixing rules employed at the relay. As per solution (a), the relay forwards the data received from source 1 with probability 0.9, while an XOR of the data received from source 1 and 2 is forwarded with probability 0.1. Solution (b) provides the exactly opposite mixing. It can be verified that even though both the above solutions reduce the square-error to zero, there is a significant difference in their performance. Solution (a) is biased towards source 1, while (b) is biased towards source 2.

4.3.3. Generalized LT distributions

Example 1 has demonstrated that when mixing two DLT codes, it is not sufficient to describe the modified distribution purely in terms of the sum-degree. This phenomenon can be attributed to the fact that a single variable distribution cannot represent the mixing that a TLT code naturally achieves when employed over a multiple segments. Hence, to appropriately represent the mixing, we deduce Generalized Target LT (GTLT) distributions from a given TLT distribution.

Consider a degree d encoding symbol formed by randomly choosing d symbols from [1, k], where the symbols can be segregated into two segments [1, k_1] and [k_1+1, k_1+k_2]. The probability of a degree d symbol being formed by choosing d_1 symbols from segment 1 and d_2 from segment 2 is given by:

$$p(d_1, d_2 | d) \approx \binom{d}{d_1} \left(\frac{k_1}{k}\right)^{d_1} \left(\frac{k_1}{k}\right)^{d_2} \tag{26}$$

Thus when an LT code Ω_x is employed on [1,k], the inter-segment mixing achieved by the code can be appropriately represented by a Generalized LT distribution $\Omega_{(x_1,x_2)}$ where the coefficients $\Omega_{(x_1,x_2),(d_1,d_2)}$ can be expressed in terms of the coefficients $\Omega_{x,d}$ as follows:

$$\Omega_{(x_1,x_2),(d_1,d_2)} = \Omega_{x,d} \binom{d}{d_1} \left(\frac{k_1}{k}\right)^{d_1} \left(\frac{k_2}{k}\right)^{d_2} \tag{27}$$

In the above equation the fractions: k_1/k and k_2/k, represent the probability of an arbitrary edge of the encoding symbol being connected to an input symbol from segment 1 and 2 respectively. Thus we can represent GTLT codes with priority by generalizing (27) in the following manner:

$$\Omega_{(x_1,x_2),(d_1,d_2)} = \Omega_{x,d} \binom{d}{d_1} (p)^{d_1} (1-p)^{d_2} \tag{28}$$

where, p represents the bias or priority of source 1. In [13] Nazanin et. al. use such a bias to realize LT codes capable of facilitating URT for different data segments. Similarly, we can use the above described distributions as targets to realize DLT codes capable of providing URT.

4.3.4. Design of DLT codes from GTLT distributions

The problem of designing DLT codes can now be stated as:

$$\left(\beta_{x_1},\beta_{x_2},\left\{\Lambda_{d_1,d_2}\right\}\right)^* =$$

Design2(D2):

$$\underset{\left(\beta_{x_1},\beta_{x_2},\left\{\Lambda_{d_1,d_2}\right\}\right)}{\arg\min}\left(\sum_{\substack{k_1\geq d_1\geq 0,k_2\geq d_2\geq 0\\ d_1+d_2\geq 1}}\left(\beta_{(d_1,d_2)}-\Omega_{(d_1,d_2)}\right)^2\right) \tag{29}$$

In our experiments we observed that for many interesting target distribution, the square-error in the above problem cannot be reduced to zero. Our simulations results will show, that it is not essential to reduce the error to zero to obtain efficient DLT codes. However, the above problem often led us to distributions which perform significantly differently, not necessarily worse, than the TLT code Ω_x. Hence, we modify the problem D2 to the following problem:

Design 3 (D3): $\left(\beta_{x_1},\beta_{x_2},\left\{\Lambda_{d_1,d_2}\right\}\right)^* =$ (30)

$$\underset{\substack{\left(\beta_{x_1},\beta_{x_2},\left\{\Lambda_{d_1,d_2}\right\}\right)\\ \Omega_{x,d}=0\Rightarrow\beta_{x_1,d}=\beta_{x_2,d}=0}}{\arg\min}\left\{\begin{array}{l}\displaystyle\sum_{\substack{k_1\geq d_1\geq 0,k_2\geq d_2\geq 0\\ d_1+d_2\geq 1}}\left(\beta_{(x_1,x_2),(d_1,d_2)}-\Omega_{(x_1,x_2),(d_1,d_2)}\right)^2\\ +\lambda\left(\displaystyle\sum_{d=1}^{k_1+k_2}\left(\beta_{x,d}-\Omega_{x,d}\right)^2\right)+\rho\left(\beta_{x,1}-\Omega_{x,1}\right)^2\end{array}\right\}$$

The objective function of the above problem is obtained by employing a typical scalarization trick used to convert a multi-objective optimization problems into a single-objective. Thus, in D3 we seek to simultaneously minimize (a) the difference between the MLT and the TLT distribution, (b) the difference between the GMLT and the GTLT distribution. Additionally, since the performance of an LT code is extremely sensitive to

the percentage of degree-one encoding symbols, we choose to regularize our solution by adding an extra cost for the differences in the degree one coefficient.

Finally, it should be highlighted that in Design D3 we employ trust region constraints. Thus, if the coefficient of a particular degree is zero in the TLT distribution Ω_x, then we force the coefficient of the same degree to be zero in the DLT codes β_{x_1}, β_{x_2}. Trust region constraints help in significantly reducing the complexity of the optimization. Nevertheless these constraints make certain combinations of (d_1, d_2) unachievable, i.e. it impossible to realize a degree (d_1, d_2) by mixing the DLT codes. In equations (6) and (7), we obtained the generalized distribution $\Omega_{(x_1,x_2)}$ from Ω_x, by distributing the probability mass $\Omega_{x,d}$ over all possible (d_1, d_2)-tuples such that $d_1 + d_2 = d$. When certain (d_1, d_2)-tuple are unachievable, we modify the CTLT distribution prior to its use in D3. We modify the CTLT by removing the probability mass associated with unachievable (d_1, d_2) and by proportionately distributing this mass over the remaining achievable (d_1, d_2).

4.4. Results & Analysis

For our experimental analysis, we considered LT distributions from [2] as targets. Due to brevity, we present the results for the LT distribution listed as Target-1 in Table I. However, the results presented here are representative of the observations made for other target distributions. In all the simulations here, we maintain the total number of input symbols k = 1000. We consider two and four source networks. In these scenarios the DLT codes are accordingly termed as DLT-2 or DLT-4 and similarly the MLT codes are

termed as MLT-2, MLT-4. We compare the performance of the MLT codes with (a) performance of networks that employ the Target-1 code directly at the source along with a switching policy at the relay and (b) performance of a Target-1 code directly applied to a concatenation of the source segment.

In all the experiments in this section, the DLT codes are derived by solving problem D3. We solve D3 using a sequential quadratic programming algorithm [13]. We heuristically choose the values $\lambda = 10$ and $\rho = 50$. However these values seem to work well for a large number of target distributions and additionally the quality of our solutions did not seem to be very sensitive to minor changes (10-25 %) in the value of λ and ρ. The choice of λ and ρ determine the tradeoff between replicating the efficiency of the target code and replicating the bias (towards a particular source) of the target code. A more rigorous approach in selecting λ and ρ shall be considered in future.

In the first scenario we consider, each data segment is assumed to be of equal size and priority. Some of the degree distributions obtained for this scenario are listed in Table I. It can be clearly seen that the MLT distribution, listed in column two and four, closely match the Target-1 LT code. For furthur perusal, column three of Table I lists the DLT-2 distribution, while Table-II lists $\Lambda_{d_1 d_2,\{1,2\}}$, the conditional probability of XOR-ing the packets at the relay. An important fact to note is that in the designs employed in [5], the value of $\Lambda_{d_1,d_2,\{1,2\}}$ is equal to 1 for all degrees that are greater than 1 and not equal to the "*spike*" of the RSD. As against this, it can be observed in Table II that $\Lambda_{d_1,d_2,\{1,2\}}$ is significantly far from either zero or one for a number of degree pairs. Figure 6 shows that this comparatively "weak" mixing has very little impact on the performance of the MLT

codes. In Figure 6 we compare the performance of the MLT codes to a switching policy with equal priority. We observed that the performance of MLT-2 and MLT-4 codes were almost identical to that of the Target-1 code. However, it can be clearly seen that the performance of a switching policy is significantly worse. The difference in performance between switching and mixing increases with the number of sources.

We also consider two additional scenarios which require us to incorporate flow priorities. In first of these scenarios, we partition the input symbols into two un-equal segments of size 300 and 700. We realize proportional priority for each of these segments by setting p=0.3 while deriving the GTLT distribution. The MLT code achieved by our design has been listed in column six of Table I. It can be observed that the MLT code again matches the target distribution very closely. Thus, the *bias* towards a particular source is purely realized on account of the GTLT distribution.

Under proportional fairness, the MLT codes should recover an equal proportion of symbols from all the segments. Figure 7 shows that the MLT codes are able to provide reasonably fair recovery. For comparison we simulated a comparative switching scheme with priorities 0.3 and 0.7. Figure 7(a) clearly exhibits the benefits of mixing in a scenario where the segment sizes are not equal.

In the second scenario, we employ the above designed DLT distributions and rules on segments of equal size. Figure 7(b) shows the performance under such a design. It can be observed that recovery time of the more important source is comparable under switching and mixing. However, the performance the delay experienced by the low priority source is significantly worse under switching.

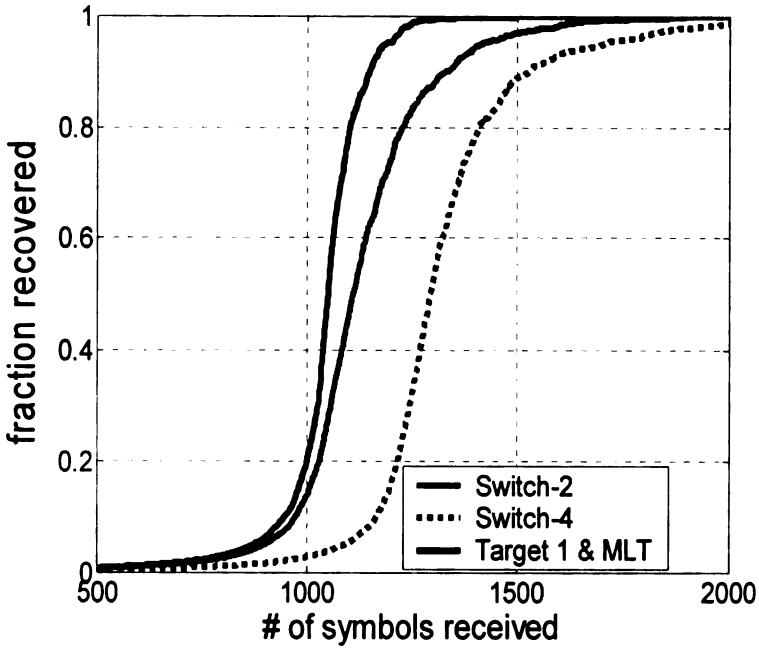
Table A.I Single variable LT distributions

	Target 1	MLT–2 ($p = 0.5$)	DLT-2 ($p = 0.5$)	MLT-4 ($p = 0.5$)	MLT-2 ($p = 0.3$)
1	0.008	0.008	0.495	0.009	0.009
2	0.494	0.494	0.282	0.493	0.494
3	0.166	0.167	0.074	0.167	0.167
4	0.073	0.074	0.031	0.074	0.073
5	0.083	0.083	0.009	0.083	0.082
8	0.056	0.054	0.046	0.052	0.054
9	0.037	0.037	0.010	0.036	0.037
19	0.056	0.056	0.035	0.057	0.056
65	0.025	0.025	0.019	0.026	0.025
66	0.003	0.003	0.000	0.004	0.003

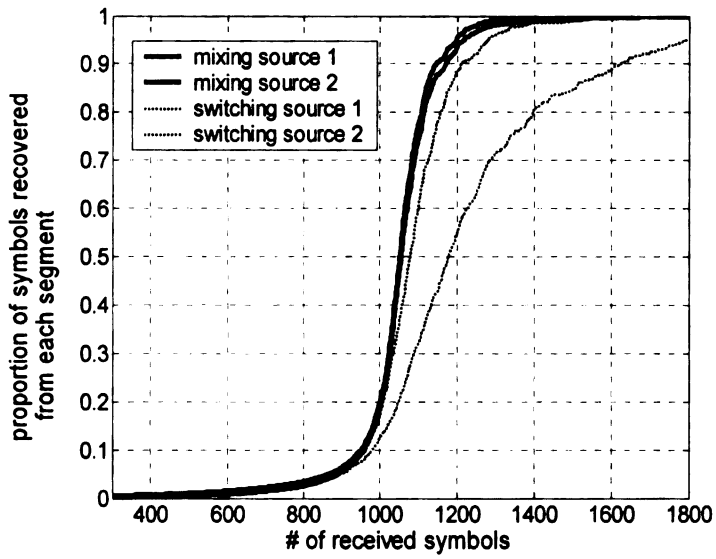
Table A.II Mixing Rule {1,2}

	1	**2**	**3**	**4**	**5**	**8**	**65**
1	1	0.42	0.50	0.95		0.48	0.18
2	0.44	0.31	1				
3	0.49	1			0.34		
4	0.94			0.12	0.68		
5			0.34	0.68			
8	0.48						
65	0.18						

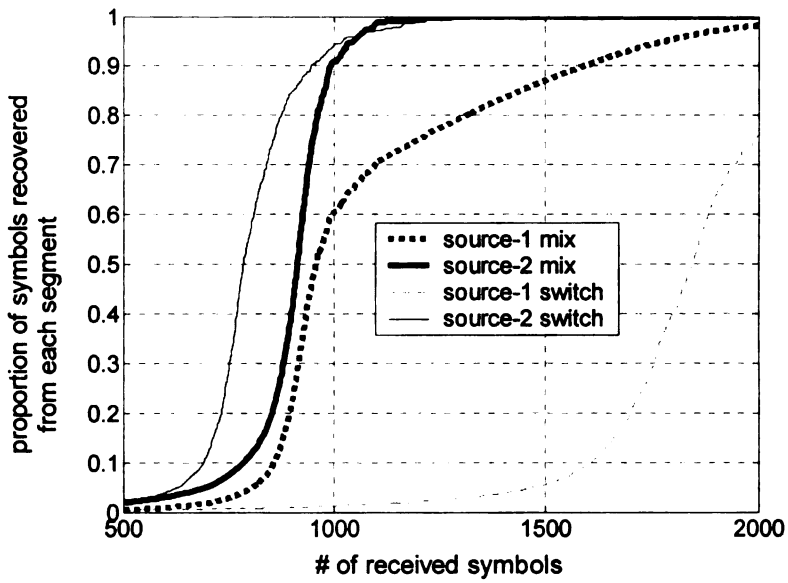
*Each row correpsonds to a value of d_1 while each column corresponds to a value of d_2.



FigureA.6 Comparison between Target LT code, MLT codes and Switching. The performance of the MLT-2 and MLT-4 codes was observed to be almost identical to that of the Target LT code. Hence in the above figure, the individual plots corresponding to each scheme are not discernable.



(a)



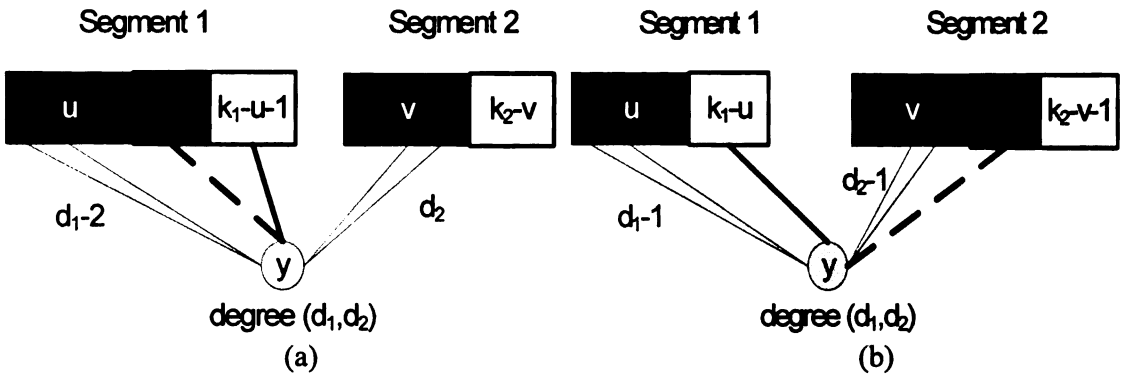
(b)

FigureA.7 Comparison between prioritized mixing and prioritized switching. (a) Segment 1 contains 300 symbols and segment 2 contains 700 symbols (b) Both segments contain 500 symbols. For both (a), (b) data from source 1 is transmitted with priority 0.3.

4.5. Discussion: Analysis of Generalized Ripples

In the main-body of the chapter, we have focused on deducing DLT from a target distribution. However, in general target distributions cannot be used to determine the mixing of two LT encoded flows. In practice we may often encounter scenarios where the source LT distributions are fixed. With such additional constraints it may be difficult to achieve specific target distributions. Consequently, it is essential to develop analytical tools that can evaluate the efficiency of GMLT distribution without reference to a target distribution. In this section we provide a brief sketch of the approach we shall be adopting in future for generalizing the presented work.

The analysis in [7]-[9] is specific to the single variable LT distribution. Here, we adopt an approach similar to [8] to extend the analysis described in Chapter 2 to GLT distributions. The decoding of GLT codes can be described in terms of colored ripples. As discussed previously, an output symbol in the ripple contains only one input symbol that has not been already recovered. We associate a color with each segment and thus utilize the un-recovered input symbol to associate a color with each output symbol in the ripple. Thus the ripple can be segregated into multiple colored ripples. To build upon the discussion in [8] we can now look upon the colored ripples as multiple-classes in a single buffer and hence the overall ripple can now be modeled as a single buffer multi-class queue.



FigureA.8 Scenarios that may lead to new arrivals in the ripple associated with segment 1. (a) Input symbol currently being recovered belongs to segment 1 (b) Input symbol belongs to segment 2

Let us focus on the two segment scenario and the ripple associated with segment 1, ripple 1. Let where k_1 be the size of segment 1 and k_2 be the size of segment 2 .We say that a decoder is in state (u,v) when u symbols from segment 1 and v symbols from segment 2 have already been recovered. Figure 8 shows the two scenarios which may cause an encoding symbol y with degree (d_1, d_2) to join the ripple-1 when a decoder is in state (u,v). Figure 8 (a) shows a scenario where the decoder goes from state (u,v) to state $(u+1,v)$ by recovering symbol $u+1$ from segment 1. At the end of this transition, ripple-1 is transformed as follows:

(a) All encoding symbols in the ripple, connected to $u+1$, are removed. Note all the input symbols connected to such encoding symbols are already recovered.

(b) All encoding symbols, previously not in the ripple, which can be represented by y are added to the ripple.

Thus, if we let $X_{i,u,v}$ represent the size of ripple-i, then the above described transition from state (u,v) to state $(u+1,v)$ modifies the size of ripple-1 as follows:

$$X_{u+1,v} = X_{u,v} - 1 - L_{1,1}(u,v) + a_{1,1}(u/k_1, v/k_2) \qquad (31)$$

In the above recursive expression, $1+L_{1,1}(u,v)$ represents the number of encoding symbols removed from ripple-1 when symbol u+1 is recovered. It can be easily shown that $L_{1,1}(u,v)$ is governed by the binomial distribution $B\left(X_{u,v}-1,\frac{1}{k_1-u}\right)$. Thus the expected number of symbols that *defect* from the ripple are

$$\overline{L_{1,1}(u,v)}=(X_{u.v}-1)/(k_1-u) \tag{32}$$

Furthermore, in (31), *arrival rate* $a_{1,1}(u/k_1, v/k_2)$ represents the number of symbols added to the ripple-1 when transiting from state (u,v) to (u+1,v). If we let, $t_1 = u/k_1$, $t_2 = v/k_2$ then in the limit $k_1 \to \infty$, $k_2 \to \infty$, (31) can be re-written as

$$\frac{\left(X_{u+1,v}-X_{u,v}\right)/k_1}{1/k_1} = a_{1,1}(u/k_1, v/k_2) - 1 - \frac{X_{u,v}/k_1}{(k_1-u)/k_1}$$

$$\Rightarrow x_1^{(1,0)}(t_1,t_2) = a_{1,1}(t_1,t_2) - 1 - \frac{x_1(t_1,t_2)}{1-t_1} \tag{33}$$

where,

(a) the state of the decoder can be described by (t_1, t_2)

(b) t_1, t_2 can now be considered continuous variables

(c) the size of the ripple-1 in state (t_1, t_2) is given by $k_1 \cdot x_1(t_1,t_2)$

(d) $f^{(i,j)}(t_1,t_2)$ represents the $(i,j)^{th}$ mixed derivate of $f(t_1,t_2)$ w.r.t t_1 and t_2.

The above analysis can be replicated for Figure 8 (b). Figure 8 (b) shows a scenario where the transition is from state (u,v) to state (u,v+1) upon recovery of symbol v+1 from segment 2. Thus it can be shown that

$$\frac{\alpha_1}{\alpha_2} x_1^{(0,1)}(t_1,t_2) = a_{1,2}(t_1,t_2) \tag{34}$$

where, $\alpha_1 = {}^{k_1}\!/_{k}$, $\alpha_2 = {}^{k_2}\!/_{k}$ and $a_{1,2}(t_1,t_2)$ is the limit of $a_{1,2}(u/k_1, v/k_2)$, such that $a_{1,2}(u/k_1, v/k_2)$ represents the number of symbols added to ripple-1 when the decoder transits from state (u,v) to state $(u,v+1)$.

The arrival rates $a_{1,1}(t_1,t_2)$ and $a_{1,2}(t_1,t_2)$ can be calculated by evaluating the probability $p_{u,v,(i,j)}$ where,

$$p_{u,v,(i,j),(d_1,d_2)} = \text{prob}\left(\begin{array}{l} \textit{An encoding symbol } y \textit{ with deg. } (d_1,d_2) \textit{ joins ripple-}i \textit{ when the} \\ \textit{decoder transits from state } (u,v) \textit{ by recovering a symbol from seg. } j \end{array} \right) \tag{35}$$

and as $k_1 \to \infty$, $k_2 \to \infty$ we have $p_{u,v,(i,j),(d_1,d_2)} \to p_{t_1,t_2,(i,j),(d_1,d_2)}$

It can be shown that

$$p_{t_1,t_2,(1,1),(d_1,d_2)} = \frac{1}{\alpha_1 \cdot k} d_1 (d_1 - 1)(1-t_1)(t_1)^{d_1-2} \cdot (t_2)^{d_2} \tag{36}$$

$$p_{t_1,t_2,(1,2),(d_1,d_2)} = \frac{1}{\alpha_2 \cdot k} d_1 (d_2)(1-t_1)(t_1)^{d_1-1} \cdot (t_2)^{d_2-1} \tag{37}$$

Thus we have

$$a_{1,1}(t_1,t_2) = \gamma k \sum_{d_1,d_2} p_{t_1,t_2,(1,1),(d_1,d_2)} = \frac{\gamma}{\alpha_1}(1-t_1)\beta^{(2,0)}(t_1,t_2) \tag{38}$$

$$a_{1,2}(t_1,t_2) = \gamma k \sum_{d_1,d_2} p_{t_1,t_2,(1,2),(d_1,d_2)} = \frac{\gamma}{\alpha_2}(1-t_1)\beta^{(1,1)}(t_1,t_2) \tag{39}$$

Thus the variation in the size of ripple-1 is described by the following system of differential equations:

$$x_1^{(1,0)}(t_1,t_2)=\frac{\gamma}{\alpha_1}(1-t_1)\beta^{(2,0)}(t_1,t_2)-1-\frac{x_1(t_1,t_2)}{1-t_1} \tag{40}$$

$$x_1^{(0,1)}(t_1,t_2)=\frac{\gamma}{\alpha_1}(1-t_1)\beta^{(1,1)}(t_1,t_2) \tag{41}$$

It can be easily verified that the following equation provides a solution for the above system is given by :

$$x_1(t_1,t_2)=(1-t_1)\left(\frac{\gamma}{\alpha_1}\beta^{(1,0)}(t_1,t_2)+\log(1-t_1)\right) \tag{42}$$

Similarly, it can be shown that

$$x_2(t_1,t_2)=(1-t_2)\left(\frac{\gamma}{\alpha_2}\beta^{(0,1)}(t_1,t_2)+\log(1-t_2)\right) \tag{43}$$

where $k_2 \cdot x_2(t_1,t_2)$ gives the size of ripple -2.

Equations (43) and (44) can be used to identify good GLT distributions in the following manner: Consider a two dimensional co-ordinate space as shown in Figure 9. For any ε less than a sufficiently small δ we say that:

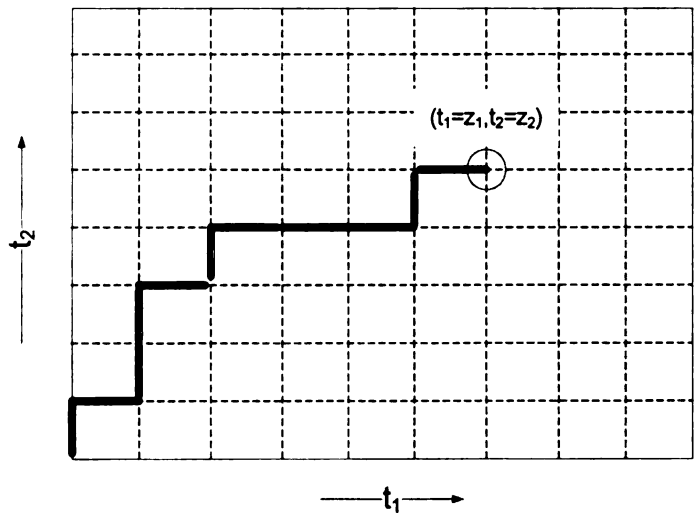


Figure A.9 Feasible decoding path to $(t_1 = z_1,\ t_2 = z_2)$

(a) A horizontal movement from (t_1,t_2) to $(t_1,t_2+\varepsilon)$ is feasible iff $x_1(t_1,t_2)>0$.

(b) A vertical movement from (t_1,t_2) to $(t_1,t_2+\varepsilon)$ is feasible iff $x_2(t_1,t_2)>0$.

(c) There exists a feasible path from $(t_1=o_1,t_2=o_2)$ to $(t_1=z_1,t_2=z_2)$ iff we can go from (o_1,o_2) to (z_1,z_2) in a sequence of feasible movements.

The success of LT decoding can be equated to the existence of a feasible path. The existence of a feasible path is determined by γ and the GLT distribution β. For any distribution β, the choice of a large enough γ ensures the existence of a feasible path. However, good GLT distributions can be identified by minimizing the value of γ, essential for the existence of a feasible path.

In order to examine the accuracy of our analysis, we consider the following GLT distribution:

$$\begin{aligned}\beta_{GLT-example} &= 0.1667x_1 + 0.0556x_2 \\ &+ 0.2222{x_1}^2 + 0.0556x_1x_2 + 0.0556{x_1}^2 \\ &+ 0.1667{x_1}^3 + 0.1667{x_1}^2x_2 + 0.0556x_1{x_2}^2 + 0.0556{x_2}^3\end{aligned} \tag{44}$$

We consider two segments of size 500 each. In the following Figure we plot the performance of the above distribution as observed on the basis of actual simulations and as predicted by the feasible path analysis. It can be clearly seen that our analysis predicts the performance of GLT codes reasonably accurately. Currently, we are in the process of designing DLT codes based on the above analysis.

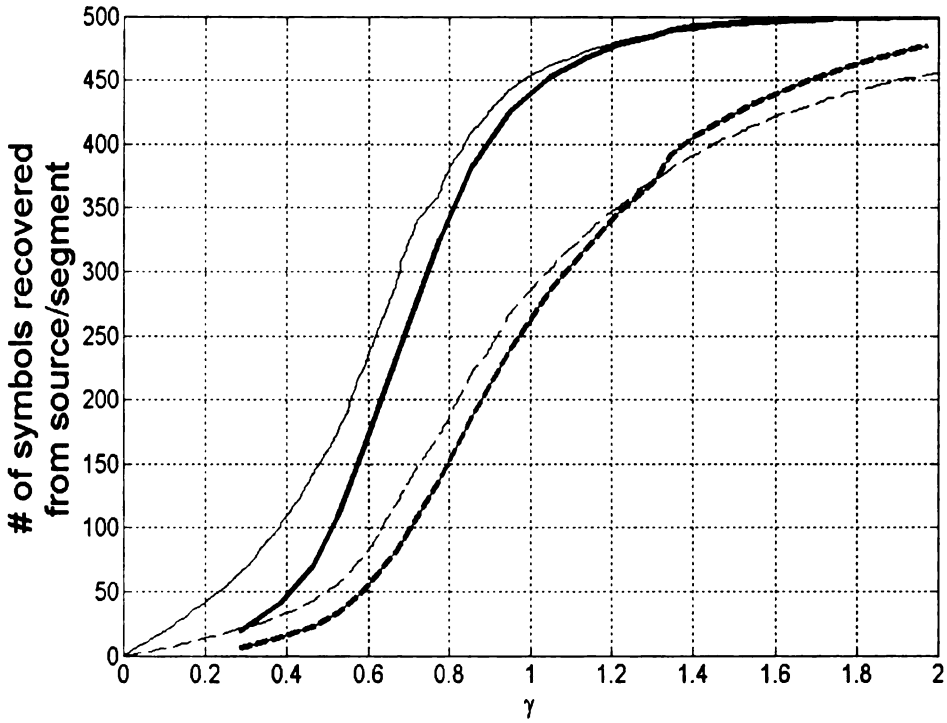


Figure A.10 Performance $\beta_{GLT-example}$

5. PART-A REFERENCES

[1] M. Luby, "LT codes", FOCS, 2002.

[2] A. Shokrollahi, "Raptor Codes," IEEE Trans. on Information Theory, vol. 52, no. 6., pp. 2551-2567, June 2006.

[3] P. Maymounkov, "Online Code" at http://www.scs.cs.nyu.edu/~petar/oncodes.pdf

[4] J. W. Byers, M. Luby and M. Mitzenmacher, "A Digital Fountain Approach to Asynchronous Reliable Multicast," IEEE J-SAC, 20 (8), pp. 1528-1540, October 2002.

[5] N. Rahnavard and F. Fekri, "Bounds on maximum-likelihood decoding of finite-length rateless codes," in Proc. of the 39th annual Conference on Information Sciences and Systems (CISS'05), Baltimore, MD, March 2005.

[6] Ki-Moon Lee and Hayder Radha, "A Maximum-Likelihood Decoding Algorithm of LT Code with a Small Fraction of Dense Rows," Proceedings on ISIT 2007.

[7] R. Darling and J. Norris, "Structure of large random hypergraphs," Annals of Applied Probability, vol. 15, no. 1A, pp. 125-152, 2005.

[8] B. Hajek, "Connections between network coding and stochastic network theory," Stochastic Networks Conference, June 19-24, 2006, Urbana.

[9] S. Sanghavi, "Intermediate Performance of Rateless Codes," arXiv:cs/0612075v1 [cs.IT] submitted 15 Dec 2006.

[10] S. Puducheri, J. Kliewer, T. E. Fuja, "The design and performance of distributed LT codes" accepted for publication in IEEE Transactions on Information Theory, 2007. available at (www.nd.edu/~jkliewer/chapter/PKF_TransIT07.pdf)

[11] A. Kamra, J. Feldman, V. Misra, D. Rubenstein, "Growth Codes: Maximizing Sensor Network Data Persistence," ACM Sigcomm, 2006.

[12] A. G. Dimakis, V. Prabhkaran, K. Ramchandran, "Distributed Fountain Codes for Networked Storage," ICASSP 2006.

[13] N. Rahnavard, B. N. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," IEEE Trans. on Information Theory, vol. 53, no. 4., pp. 1521-1532, April 2007

[14] Eryilmaz A., Ozdaglar A., Medard M., "On Delay Performance Gains from Network Coding", Proceedings of Conference on Information Sciences and Systems, Princeton, 2006.

[15] A. Majumdar, D. G. Sachs, I. Kozintsev, K. Ramchandran, and M. Yeung "Multicast and Unicast Real-Time Video Streaming over Wireless LANs," IEEE Trans. on CSVT, Vol.12, June 2002/

[16] B. Radunovic and J.-Y. Le Boudec, "A Unified Framework for Max-Min and Min-Max Fairness with Applications," *to appear* ACM/IEEE Transactions on Networking.

[17] K. Price and R. Storn, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, pp. 341–359, 1997.

[18] P.E Gill, W. Murray, and M.H. Wright, "Practical Optimization", London, Academic Press, 1981.

PART B

CROSS LAYER DESIGN

Importance of Side-information

B. 1. INTRODUCTION

1.1. Motivation

The challenges faced by high bitrate transmissions (such as video) over wireless networks have necessitated the design of more flexible protocols. Lack of information transmission across layer boundaries is considered to be a key shortcoming of the conventional protocol stacks for the wireless media. In order to provide support for high bitrate transmission, better power management, improved QOS, more efficient routing, packet scheduling and improving other network functionalities over wireless networks, there is an increased willingness to allow inter-layer communication. Network strategies, which allow cross-layer information transmissions, can be broadly classified as cross-layer protocols. Examples of these protocols and related wireless schemes can be found in [1]-[21].

Of particular interest to the work presented in this section are the cross-layer designs presented in [7]-[22]. Traditionally, the link/Medium Access (MAC) layer detects the presence of bit corruptions in a received packet with the help of checksums and drops all the packets that are detected to be corrupted. Such an operating principle can often lead to an excessive number of packet drops, thus severely affecting the performance of a dependent application. The protocols presented in works such as [7]-[22] propose to overcome an excessive loss in throughput by relaying corrupted packets to the application layer. Figure 1 (a) and Figure 1 (b) provide a schematic of a conventional and a cross-layer protocol design, highlighting the key difference in their operation.

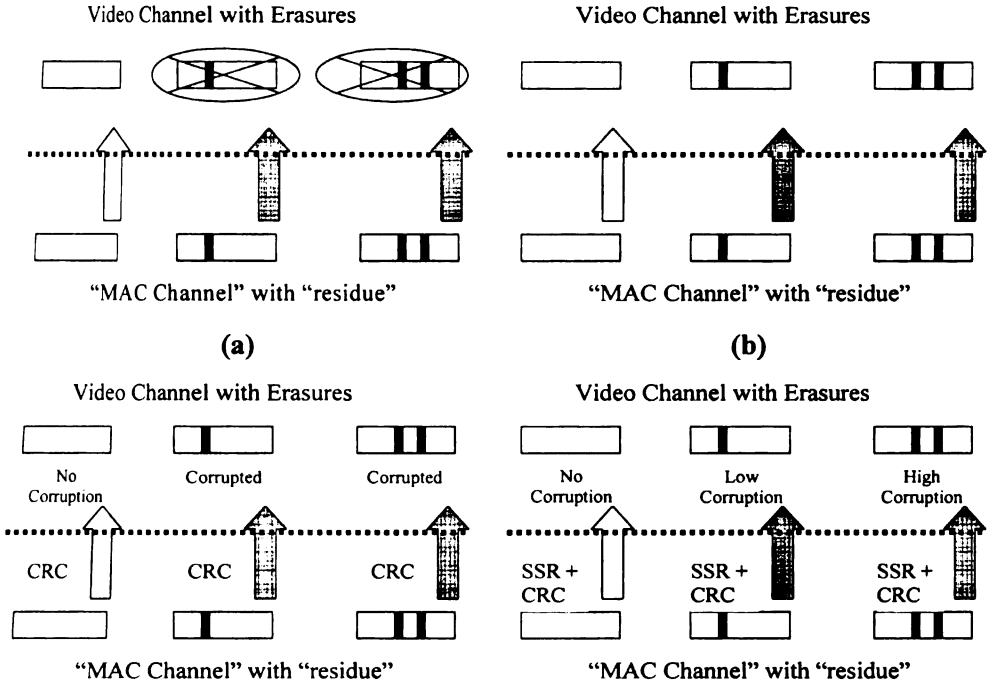


Figure B.1 Schematic of the different protocols considered in this work.
(a) A conventional protocol that drops all corrupted packets.
(b) A cross-layer design under which corrupted packets are relayed to the application layer.
(c) A cross-layer design (protocol) with side-information under which corrupted packets are relayed to the application layer along with binary side-information that allows differentiation between corrupted and un-corrupted packets.
(d) An SSR_aware cross-layer design (protocol) with side-information under which corrupted packets are relayed to the application layer along with:
    (i) binary side-information that allows differentiation between corrupted and un-corrupted packets.
    (ii) side-information based on an observable variable, Signal-to-Silence Ratio (SSR), which facilitates a finer differentiation between the corruption levels in the packet.

As shown in Figure 1 cross-layer designs lead to an application layer channel that consists of both erasure and errors. Hence we shall broadly refer to the cross-layer designs considered in this part as hybrid erasure-error protocols (HEEPs). Furthermore, it

should be highlighted that from here on the discussion on "cross-layer" protocol designs is strictly within the paradigm of HEEPs.

The guiding principle in the design HEEPs has been the assumption that applications stand to benefit from receiving an increased amount of data, even if the received data is noisy. The common intuition has been that, if the number of errors in the corrupted packets relayed to the application layer is not large then the total number of "channel failures" is reduced. (Here, a channel failure could be an error or an erasure.) This reduction in the total number of failures can lead to an increase in channel capacity. However, there had been no formal attempt prior to the work presented in this part, to explicitly quantify the gain in channel capacity on account of employing a HEEP. Studies such as [7]-[22] have implicitly assumed that the probability of bit-error in the corrupted packets is always low enough to dramatically reduce the total number of bit/symbol impairments and thus render the performance of a cross-layer approach to be better than a conventional one. We motivate the need for a formal investigation by noting that the erasure correcting capability of any channel code is much better than its ability to correct symmetric errors[1]. Therefore, if the corrupted packets have a high percentage of errors, allowing the existence of corrupted packets at the application layer can in fact lead to a reduction in throughput and channel capacity. Thus there is an inherent tradeoff in relaying corrupted packets to the application layer and the utility of a HEEP is strongly coupled with the corruption level in the relayed packets.

---

[1] For example, it is known that a linear code with minimum distance d_{min} can correct upto $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ errors, as against the ability to correct upto $d_{min}-1$ erasures [31], [32].

The exact modifications in a conventional protocol that make the relay of corrupted packets feasible are implementation specific and in practice can vary significantly. However, a standard approach has been to employ partial-checksums which are calculated only on a certain important parts of the packet (e.g. just the headers), instead of the entire packet, and thus have lesser likelihood of failure. Therefore, approaches explored in works such as [7]-[22] naturally lead to a reduction of packet drops but at the same time also deprive the higher layer of an ability to differentiate between corrupted and non-corrupted packets. This loss of channel state information (CSI) leads to poor error localization and thus can adversely impact the error recovery. This observation motivates the development of cross-layer design with side-information. Figure 1 (c) and (d) show examples of cross-layer designs with side-information. Figure 1 (c) shows a design which utilizes a simplistic binary side-information. Additional improvements can be obtained by differentiating in packet corruptions at a finer level and thus we shall explore the feasibility and utility of realizing more advanced cross-layer designs represented by Figure 1 (d).

1.2. Overview of Contributions

The contributions in this part can be further segregated into three important sub-parts named (I) Hybrid Erasure Error Protocols (II) Utility of Practically Observable Variables in HEEPs (III) Cross-layer Information Exchange. In sub-part I we formulate simplistic albeit important abstractions of the considered protocols to deduce important insights and protocol design guidelines. The sub-part II improvises upon these guidelines to identify mechanisms in actual 802.11b implementations that can facilitate improved cross-layer

protocols in practical settings. The sub-part III extends the above ideas to the exciting new domain of network coding. Important contributions in each of these sub-parts are elaborated upon in the following:

1.2.1. Hybrid Erasure Error Protocols:

In this sub-part, the primary goal of the presented work is to analyze and identify the channel conditions under which the existence of corrupted packets at the application layer can be preferred over dropping them entirely. In order to keep the discussion generic and not dependent on a particular implementation (or standard), in this part we consider three rather abstract communication schemes:

- transmission over erasure channels, which represents the conventional protocols, as illustrated by Figure 1 (a)
- transmission in presence of erasures and errors using a cross-layer design (CLD) as illustrated by Figure 1 (b)
- side-information enhanced transmission in presence of erasures and errors using a cross-layer design (CLDS) as illustrated by Figure 1 (c).

We evaluate and make a comparative analysis of the channel capacities of each of the above mentioned schemes over single and multi-hop wireless channels in order to identify the conditions under which the cross-layer protocols can provide improved performance. As a crucial contribution of this work, we show the existence of channel conditions when the performance of CLD is worse than CON. We further show that it is feasible to achieve performance improvements using HEEP in a guaranteed manner by employing CLDS. We highlight the importance of side-information from lower layers

and the utility of distinguishing between corrupted/non-corrupted packets by proving that the capacity of the CLDS scheme is always better than either CLD or CON.

Despite increase in capacity, the utility of a cross-layer approach can be undermined, if an application layer Forward Error Correction (FEC) is unable to take advantage of it or if the improvement in throughput does not translate into an improvement in video quality. Hence, we compare the performance of Reed-Solomon (RS) [31]-[35] and Low Density Parity Check (LDPC) [36]-[39] codes based FEC schemes on CON, CLD and CLDS. We compare the three schemes and their combinations with the above FEC schemes in terms of video quality. Simulation results validate the predictions made by the capacity deductions.

1.2.2. Utility of Practically Observable Variables in HEEPs

The performance of HEEPs can be improved with the help of Receiver-side Channel State Information (CSI). However, in practice, the bits observed at the link-level do not have an inherent soft-information associated with them. Hence, standard physical-layer link quality indications (e.g. the signal strength of each individual bit) cannot be used as CSI. Due to the lack of a practical predictive tool, many standard optimizations for capacity planning, soft-decoding, rate control etc., cannot be practically employed with the considered cross-layer protocols. In this part we address this important engineering problem. In practice, 802.11b radio devices are capable of measuring the signal power for the first few microseconds (μs) of each packet reception. This measurement can be used to associate a coarse Signal-to-Silence (SSR) ratio with each individual packet. This work, investigates the practical predictive utility of such coarse measurements. We show that SSRs can be used to provide meaningful CSI with a reasonably link/infrastructure

invariant model training. Depending upon the specific application, the empirically determined correlation between the SSR indications and the Bit Error Rate (BER) can be utilized in a real-time or non-realtime fashion.

1.2.3. Cross-layer Information Exchange

The capacity of wireless networks can be limited due to Medium Access Congestion [47]. Network Coding (NC) [48], [49], [50] in combination of physical layer broadcasting ([51] and references within, [52]) can be used to reduce the impact of congestion. In addition to Medium Access Congestion, the throughput of wireless networks can also be further reduced due to excessive packet drops due to bit corruptions. A successful strategy to counter such excessive packet drops is achieved by deploying HEEPs. Therefore, in this part we evaluate the feasibility of such a combination and in particular it's utility for wireless video. More specifically, we consider the Mutual Information Exchange (MIE) problem introduced by Wu et. al. in [52] and exhibit the utility of combining NC with HEEPs.

The remainder of this part is organized as: Chapter 2 provides a summary of the related work. Chapter 3 is focused on preliminary investigations into HEEPs, Chapter 4 provides a performance evaluation of practically observable CSI, Chapter 5 considers the application of Network Coding (NC) in presence of corrupted packets.

B. 2. RELATED WORK

2.1. Hybrid Erasure Error Protocols

Larzon et al. [7]-[9] were the first to explicitly advocate the relay of corrupted packets for multimedia transmission when they presented the UDP-lite protocol. The UDP-lite protocol was realized by employing partial checksums and the experimental work was primarily based on streaming over the Internet. The UDP-lite approach has subsequently been extended to wireless architectures [10]-[14]. Singh et. al. extended this work for cellular video [10]. Zheng et. al. (e.g. [11]) presented the Complete UDP Protocol (CUDP) which used the frame error rate from the Radio Link Physical Layer for improved performance. Khayam et. al. extended this work to 802.11b WLANS [14]. Servetti et. al. have investigated the performance of HEEP in conjunction with Unequal Error Protection [15]. Masala et.al investigated HEEP in conjunction with Hybrid ARQ and also the performance of HEEPs over 802.11b networks [16]-[18]. As shall be explicitly shown in our analysis, the performance of HEEPs can be severely diminished due to excessive packet drops on account of bit corruptions in the Header. The insight developed in this work was utilized in an associated collaborative work to diminish this vulnurability in [22]. A similar idea was proposed in [19] also. In [20] HEEPs are employed to improve the range of 802.11b networks. The work presented in [21] is a standardization contribution, where efficient scalable video coding mechanisms suitable for HEEPs are investigated.

All of the above works, with the exception of [22], do not explicitly utilize any side-information to determine the corruption status of the packet. Some of the design guidelines recommended in this part were incorporated in [22] and thus a coarse usage of side-information has been made. Additionally, none of the prior works without any exception employ soft channel decoding for error recovery and thus report performances that are significantly worse than the best possible.

2.2. Link Quality Differentiated Protocols

Protocols that adjust: the TCP bandwidth guarantee or video rate control (e.g. [1], [6]), scheduling/server allocation and bandwidth sharing (e.g. [1], [5], [6]), the packet retransmission limits (e.g. [2]), PHY bit-rate (e.g. [3], [5], [6]), the Unequal Error Protection (UEP) / Joint Source Channel Coding (JSCC) schemes (e.g. [2], [3]), routing mechanism (e.g. [4]) etc, on the basis of the wireless Link-Quality (LQ), can be broadly referred to as LQ based protocols, LQP. The CSI required for the design of an efficient LQP can be obtained non-realtime with the help of a site-survey [28]-[29] or at real-time on the basis of real-time LQ monitoring. The studies mentioned above develop LQD protocols over a framework where corrupted packets are not relayed to higher layers. The SSR aware CLDS is a first attempt at developing a LQ based CLDS protocol in presence of packet corruptions. In future, all of the above mentioned protocols could be extended to work in presence of packet corruptions.

2.3. 802.11b Measurement Studies

Performance evaluations on the basis of actual measurements of wireless network play an important role in determining the practical efficiency of communication protocols and, also helps identify important design goals and constraints. While, measurement based studies of 802.11b network have been in abundance (e.g. [23]-[24]) almost all of these studies limit their analysis to packet drops and do not measure the bit corruptions at all. In the context of HEEPs notable exceptions are [14], [20]-[22]. However even these studies do not measure the bit errors vis-à-vis any indications of the radio-link quality e.g. SSR. The work presented in Chapter 4 of this part is the only study we are aware of to do such measurements. The measurements presented in section 4 were done in conjunction with the work presented in [25]. Thus the 802.11b measurement methodology and the first order dependence of BER on SSR presented in Section 4 (i.e. Figure 14 and Figure 15) also appear in [25]. In this work we utilize SSR as side-information to improve the performance of cross-layer protocols, while the focus of [25] was to provide improved channel models for simulation purposes.

2.4. Network Coding

Traditional data routing is based on a store and forward policy. Network Coding generalizes traditional routing by forming combinations of packets within the network. The seminal work by Ahlswede et. al. [48] shows that such combinations can lead to a throughput increase. In [49], Li et.al. have shown the sufficiency of Linear Network Coding in achieving the multicast capacity of single source network on a directed graph. In [50], Koetter et. al. develop an algebraic framework for employing network coding.

The area of network coding has seen tremendous research interest and in a short time, immense literature has been generated on this topic. We refer the reader to [51] for an introduction to this area.

Network Coding can be efficiently combined with the broadcast nature of wireless networks. We build upon the Information Exchange problem introduced by Wu. et. al. [52]. We extend this problem to the noisy case, where packets may contain bit corruptions and the processing within the network is limited to simple XORs. Such a communication paradigm has also been considered by Tuninetti et. al. [53].

B. 3. HYBRID ERASURE ERROR PROTOCOLS

3.1. Quasi-static Memoryless Channel Abstractions

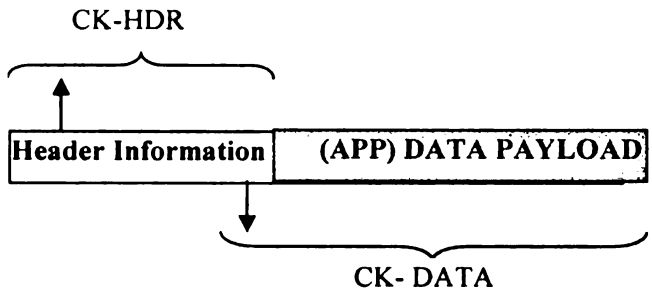


Figure B.2 A single Logic Transmission Unit (LTU).

In this section we characterize the three abstract protocols under consideration. The three communication schemes considered in this sub-part can be explained by considering a generic Logic Transmission Unit (LTU) as shown in Figure 2. The general packet structure can be segregated into two parts 1) the header information[2] and 2) the data payload. In addition to traditional header information (e.g., node addresses etc), the LTU header contains two sets of checksums CK-HDR and CK-DATA. The CK-HDR checksum is applied to and dependent on the header information only; while the CK-

---

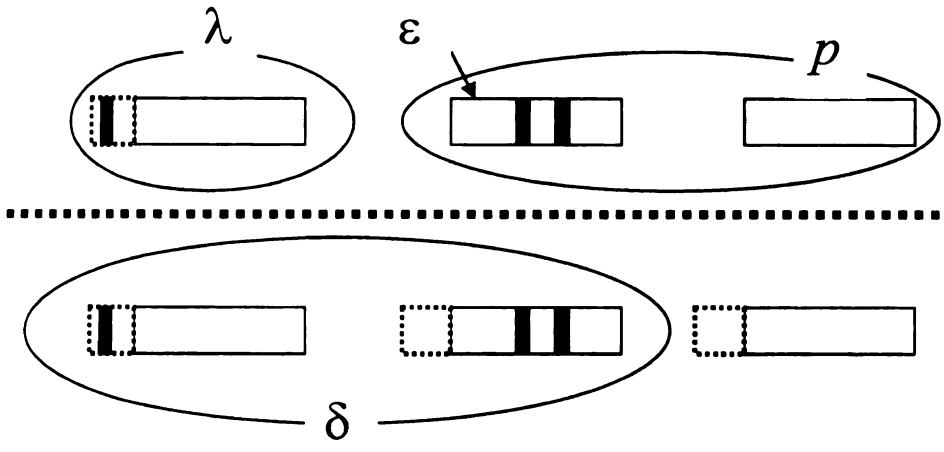
[2] In many practical implementations the header and CK-HDR might be further partitioned into multiple headers and checksums. In addition a direct correlation of a specific standard/architecture/implementation with the above-considered abstract LTU may not always be possible. Reallocation (or even addition) of some header fields might be required. However, we intentionally do not address such implementation details to maintain the generic nature of arguments presented in this section.

DATA check sum is applied to and dependent on the data payload only. Hence, under this generic LTU model:

- CON, the conventional (non-cross layer) protocol drops a packet if either of the checksums CK-HDR, or CK-DATA is not satisfied and hence has been represented by schematic Figure 1 (a)
- CLD, which represents protocols like UDP-lite [7]-[21], turns the CK-DATA checksum off and drops the packet only if CK-HDR is not satisfied. Therefore, a CLD channel exhibits both erasures (due to CK-HDR violations) and possible errors in some of the delivered packets. As explained before this protocol can also be represented by Figure 1 (b). It is important to note that (without further information or additional parity bits) the CLD channel receiver does not know which delivered packets are error-free and which packets are corrupted. It only distinguishes between erasures and delivered packets.
- CLDS is an alternative to the above schemes. Similar to CLD, a CLDS channel drops a packet only if CK-HDR is not satisfied. However, in CLDS the CK-DATA is not turned off but neither is the decision to drop a packet dependent on this checksum. Moreover CK-DATA and information about the success or failure of this check-sum is made available to the application layer as side information. Therefore, and unlike a CLD receiver, the CLDS receiver can distinguish corrupted packets from error-free packets. Thus the schematic of CLDS was represented by Figure 1 (c)

The channels under consideration can thus be parameterized by:

- δ : The probability that at least a single bit is in error in the header and/or the data payload. Thus δ is the probability of a packet being dropped in a conventional (non-



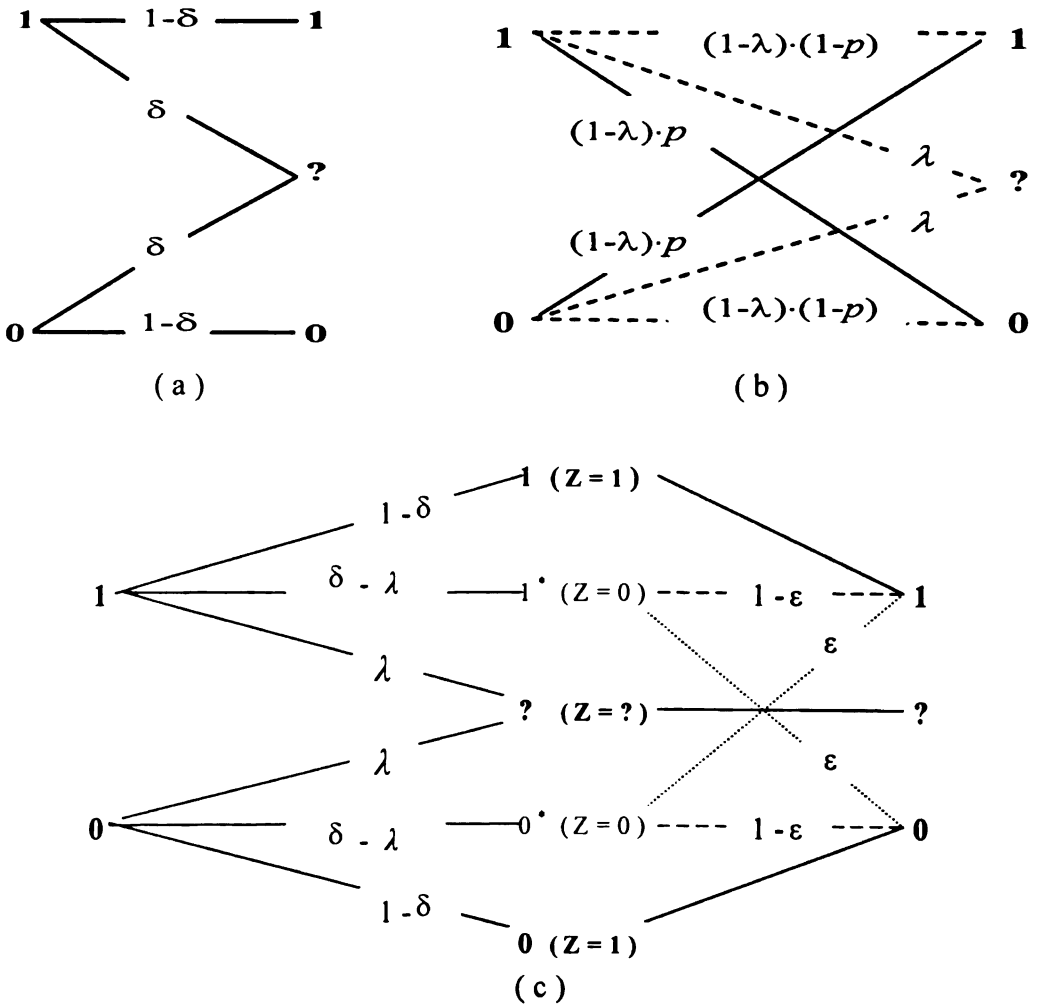
FigureB.3 Important parameters utilized to formulate the channel abstractions of protocols

cross layer) protocol because at least one of the checksums, CK-HDR and/or CK-DATA, was not satisfied.

- λ : The probability that the packet header contains at least a single bit in error. Thus λ is the probability of packet being dropped in the cross-layer schemes because the check CK-HDR was not satisfied. (Note that this event could occur regardless if there is an error within the packet data or not.)
- ε : The conditional probability of a bit in the data payload being in error given that the checksum CK-HDR is satisfied and checksum CK-DATA has failed. Given a corrupted packet at a CLD/CLDS receiver, ε represents the probability of having a random bit selected from that packet to be in error, i.e. ε specifically represents the probability of error in corrupted packets relayed to the application layer.

In Figure 3 the important parameters δ, λ and ε, that determine the channel abstractions of the considered cross-layer protocols, have been identified with reference to Figure 1, in order to provide a more graphic illustration of their definition.

In order to formulate the channel abstractions we also need to define:



FigureB.4 (a) Binary Erasure Channel (BEC) representing the UDP channel (b) Hybrid Binary Symmetric/Erasure Channel (BSEC) (c) BSEC with Side information Z

- Z: A discrete random variable that takes on three possible outcomes: $S_Z = \{0,1,?\}$.

Where,

(i) $Z = ?$ if the header contains at least single bit error and CLDS drops the packet.

Thus $p(Z = ?) = \lambda$.

(ii) $Z = 0$ if a packet contains no errors in the header but contains at least a single bit error in the data payload. Thus $p(Z = 0) = (\delta - \lambda)$, i.e. $\delta - \lambda$ represents the

probability of a corrupted packet being delivered to a CLD/CLDS channel receiver.

(iii) $Z = 1$ if neither the header nor the data payload contain even a single erroneous bit. Thus $p(Z = 1) = (1 - \delta)$.[3]

- p: The conditional probability of a bit in the data payload being in error given that the checksum CK-HDR is satisfied. In other words p is the overall probability of bit error in packets that are received at the application layer (i.e. all packets that don't get dropped due to corruption in the header). Also note that $p = \frac{(\delta - \lambda) \cdot \varepsilon}{(1 - \lambda)}$.

Thus the CON, CLD and CLDS schemes can be represented by Figure 2 (a), (b), (c) respectively.

3.2. Capacity Evaluation

Popular information channels [40] and some extensions of these channels (as outlined below) can provide invaluable representation and insight regarding the performance of the conventional and cross-layer protocols. Our focus here is on providing a unifying framework (albeit with simplifying assumptions) for a comparative analysis among the different protocols in terms of channel capacity. We begin by briefly outlining the channel capacities of the conventional and cross-layer protocols using the generic error and erasure parameters described in the previous section:

---

[3] In this work we assume that probability of false alarm and the probability of missed detection of the checksums is negligibly small.

The conventional protocol (CON), which is the simplest among the three (CON, CLD, and CLDS), can be represented by a Binary Erasure Channel (BEC). It is well known [40] that the channel capacity of a BEC is given by $1-\delta$; hence, the capacity of a conventional protocol is given by

$$C_{CON} = 1-\delta \quad (1)$$

The cross-layer channel (CLD) can be represented as a cascade of a BEC channel with probability of erasure equal to λ followed by a Binary Symmetric Channel (BSC) with probability of bit error equal to p. Such a cascade can hence be termed as Binary Symmetric/Erasure Channel (BSEC). It can be easily shown that the channel capacity of such a cascade is given by the product of the channel capacities of the individual channels:

$$C_{CLD} = (1-\lambda)\cdot(1-h_b(p)) \quad (2)$$

where $h_b(p)$ is the entropy of a binary random variable with parameter p.

The channel capacity of the cross-layer channel in presence of side information Z is as follows: when $Z=1$ all the bits are transmitted reliably; and in this case, which occurs with probability $(1-\delta)$, the (conditional) capacity is 1. When $Z=?$ all the bits get erased and the conditional capacity is 0 while when $Z=0$ the channel reduces to a BSC with a cross-over probability ε and the conditional capacity is $(1-h_b(\varepsilon))$. Thus the channel capacity of CLDS is given by:

$$C_{CLDS} = (1-\delta)+(\delta-\lambda)\cdot(1-h_b(\varepsilon)) \quad (3)$$

Note that in the above discussion and the remainder of this part, even though the side-information is available only at the receiver we are able to deduce the overall capacity in terms of conditional capacity only because $I(X;Y,Z) = I(X;Y|Z)$.

3.2.1. Transmission over multiple-hops

A multi-hop wireless channel can be represented as a cascade of channels. For example, a cascade of BEC channels can represent the conventional scheme, where a distinct BEC channel in the cascade represents each link. Similar cascades of BSEC channels can be made for the cross-layer schemes too. However, in case of CLDS it should be noted that the side information Z shall depend on the cumulative errors encountered over all the hops. Thus in general we can express the end-to-end channel capacities over multiple hops as:

$$C_{CON(n-hop)} = \prod_{i=1}^{n}(1-\delta_i) \tag{4}$$

$$C_{CLD(n-hop)} = \left(\prod_{i=1}^{n}(1-\lambda_i)\right)\cdot\left(1-h_b\left(\overset{n}{\underset{i=1}{*}} p_i\right)\right)^{4} \tag{5}$$

$$C_{CLDS(n-hop)} = \left(\prod_{i=1}^{n}(1-\delta_i)\right) + B\cdot A \quad , where \tag{6}$$

---

[4] **Note:** Operator "*" is s.t. $p_1 * p_2 = p_1\cdot(1-p_2)+p_2\cdot(1-p_1)$ and $\overset{n}{\underset{i=1}{*}} p_i = (p_1 * p_2) * p_3)\cdots) * p_i)\cdots) * p_n)$.

$$B=\left(\prod_{i=1}^{n}(1-\lambda_i)\right)-\left(\prod_{i=1}^{n}(1-\delta_i)\right),\quad A=1-h_b\left(\left(\frac{\left(\prod_{i=1}^{n}(1-\lambda_i)\right)}{\left(\prod_{i=1}^{n}(1-\lambda_i)\right)-\left(\prod_{i=1}^{n}(1-\delta_i)\right)}\right)\cdot\left(\overset{n}{\underset{i=1}{*}}\,p_i\right)\right)$$

Equations (4), (5), (6) can be converted into equations (1), (2), (3) respectively by employing the following substitutions: $\delta=1-\prod_{i=1}^{n}(1-\delta_i)$, $\lambda=1-\prod_{i=1}^{n}(1-\lambda_i)$, $p=\left(\overset{n}{\underset{i=1}{*}}\,p_i\right)$ and $\varepsilon=\left(\frac{1-\lambda}{\delta-\lambda}\right)\cdot p$. Thus for the remainder of this part, unless explicitly stated, we find it sufficient to maintain the discussion in terms of δ, λ, p, ε.

3.2.2. Comparative Analysis

Figure 5 shows a comparison of the application layer capacities offered by the three protocols considered here, under various channel conditions. It can be clearly seen that the cross-layer protocols can provide dramatic improvements in capacity under a variety of channel conditions. Furthermore it can be observed, (i) as shown in Figure 5 (a), the relative performance of the cross-layer schemes improves when the packet drops due to data payload corruption increase (i.e. $\delta-\lambda$ increases) , while (ii) as shown in Figure 5 (b) and (c), the relative performance deteriorates as the packet drops due header corruptions increases (i.e. λ increases) or when the corruption level in the un-erased but corrupted packets increases (i.e. ε increases). Two additional important observations that can be derived from Figure 5 (c) are:

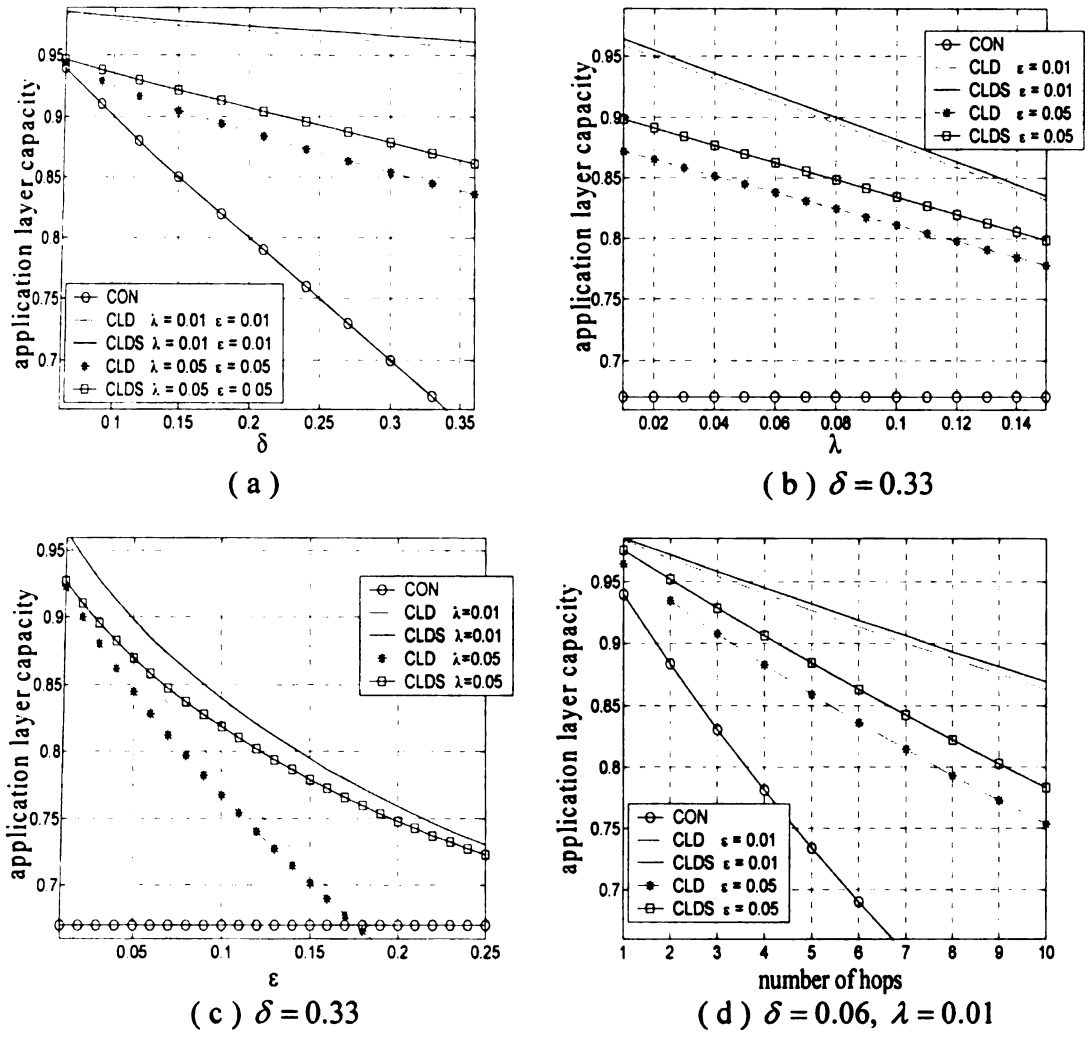


Figure B.5 Comparison of channel capacity for CON, CLD and CLDS

(i) In the absence of CK-DATA side-information, when the packets are highly corrupted, the capacity of conventional protocol can actually be better than that of the cross-layer. Thus cross-layer schemes that advocate a simple disabling of the payload checksum, should ensure that the bit-error rate in the corrupted packets is within acceptable limits, else the performance can infact worsen.

(ii) If the CK-DATA checksum is not turned off, and this information is provided to higher layers, then the resilience of HEEPs to high corruption levels, significantly

improves. In fact it can be observed that the capacity of CLDS is always better than that of CON and CLD. This improvement in capacity is substantially magnified as the corruption level increases.

Since an appropriate choice of parameters can represent a multi-hop scenario, performance trends exhibited in Figure 5 (a) (b) (c) are valid for multi-hop scenario also. It can be observed in Figure 5 (d) that even in the multi-hop case the performance of CLDS is the best. Unlike CLDS, the performance of CLD is not always better than CON. If the corruption level increases, the performance of CLD can in fact drop below CON, however as shown in Figure 5 (d), there can exist many operating conditions when the capacity of CLD is better than CON even over multiple hops.

The above observations can be formalized by considering the following two propositions. Proposition 1 states that the performance of CLDS is always better than CON and CLD under all channel conditions. While Proposition 2 will allow us to identify the channel conditions under which CLD can perform better than CON.

Proposition 1: (a) $C_{CLDS} \geq C_{CON}$ with equality occurring iff $\varepsilon = 0.5$ or $\delta = \lambda$.

(b) $C_{CLDS} \geq C_{CLD}$ with equality occurring iff $\delta = \lambda$.

Proof: a) The proof follows from observing that $(\delta - \lambda)\cdot(1 - h_b(\varepsilon)) \geq 0$. Additionally, equality occurs iff $(\delta - \lambda)\cdot(1 - h_b(\varepsilon)) = 0$, i.e. iff $\delta = \lambda$ or $h_b(\varepsilon) = 1$ i.e. $\varepsilon = 0.5$.

b) This result can be proved by using the following fact:

If $f'(x)$ is strictly monotonically decreasing over $[a,b]$ then $\forall \alpha, \beta \in [a,b]$, $\alpha, \beta \neq 0$, $\alpha < \beta \Rightarrow \frac{f(\alpha)}{\alpha} > \frac{f(\beta)}{\beta}$. $h_b'(x)$ is strictly monotonically decreasing for

$\forall x \in [0,1]$. As $1-\lambda \geq 1-\delta$ it can be shown that $p \leq \varepsilon$, which implies that $\frac{h_b(p)}{p} \geq \frac{h_b(\varepsilon)}{\varepsilon}$. Substituting for p we can conclude that $(1-\delta)+(\delta-\lambda)\cdot(1-h_b(\varepsilon)) \geq (1-\lambda)\cdot(1-h_b(p))$. Equality can occur only when $p=\varepsilon$, which is possible iff $\delta = 1$.

Proposition 2: $C_{CON} > C_{CLD} \Leftrightarrow h_b(p) > (p/\varepsilon)$.

Proof: Can be directly derived from expressions (2) and (3).

Half-Space decomposition:

For a given value of δ, λ and for $\varepsilon \leq 0.5$ the equation $h_b(p) = (p/\varepsilon)$ i.e. $h_b\left(\frac{(\delta-\lambda)\cdot\varepsilon}{(1-\lambda)}\right) = \frac{(\delta-\lambda)}{(1-\lambda)}$ has single unique solution. Combined with the above observation, Proposition 2 tells us that for a given δ, λ, there exists a threshold $\varepsilon_{\min}(\delta,\lambda)$ such that if the level of corruption in the corrupted (but not dropped) packets is greater than this threshold then the conventional schemes (CON) shall perform better than the cross-layer scheme (CLD). Hence, the surface $s = [\delta, \lambda, \varepsilon_{\min}(\delta,\lambda)]$ divides the parameter space into two distinct regions, one that contains all possible values of $\delta, \lambda, \varepsilon$ s.t. $C_{CON} > C_{CLD}$ and another that contains all possible values of $\delta, \lambda, \varepsilon$ s.t. $C_{CON} < C_{CLD}$.

For a fixed value of λ, we can identify a curve $[\delta, \varepsilon_{\min}(\delta)]$ that similarly divides the two dimensional parameter space into two regions. Figure 6 shows such curves for various values of λ. In Figure 5 the region "above" the curve corresponds to all possible

values of $[\delta, \varepsilon]$ that lead to $C_{CON} > C_{CLD}$, while the region "below" the curve corresponds to $C_{CON} < C_{CLD}$. All the points on the surface $[\delta, \lambda, \varepsilon_{\min}(\delta, \lambda)]$ correspond to the channel conditions when $C_{CON} = C_{CLD}$, similarly for a fixed λ , all the points on $[\delta, \varepsilon_{\min}(\delta)]$ correspond to the channel condition when $C_{CON} = C_{CLD}$.

The threshold defined by $\varepsilon_{\min}(\delta, \lambda)$ can be used to improve the efficiency of a communication scheme, in scenarios where side-information from the physical layer or some steady state channel statistics (or some other method) may make it possible to acquire an estimate of the corruption level in a packet. This estimate in conjunction with $\varepsilon_{\min}(\delta, \lambda)$ can allow us to identify the highly corrupted packets, which might be preferred to be dropped.

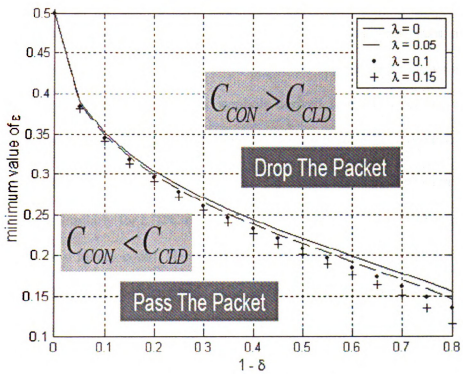


Figure B.6 $\varepsilon_{\min}$ as a function of $1-\delta$

3.3. Performance Evaluation of FEC With HEEPs

In order to exploit the increased capacity offered by the cross-layer approach, efficiency of FEC schemes suitable for the HEEPs needs to be evaluated. As many current deployments of FEC schemes for low-delay applications employ RS codes (e.g. [33]-[34]) it is important to establish the gains of using a cross-layer protocol with an RS based FEC scheme. On the other hand, the past few years have seen tremendous research interest in LDPC codes [36]-[39]. These codes are shown to be capacity achieving over a variety of channels. Thus the utility of LDPC codes in the design of efficient FEC schemes for the cross-layer channels also needs to be evaluated. In this section we shall make comparative analysis of all combinations of protocols and channel coding schemes.

For all the simulations in this part, unless stated otherwise, we use a packet block-length of 30 packets, a packet size of 500 bytes and coding rate of 0.66 (i.e. 20 message packets in each block). The packet based FEC is obtained on the basis of a simple interleaving scheme, where the entire block of packets can be broken down into c code words. Each codeword consists of u symbols derived from each packet, where each symbol can be represented by v bits. Thus for the RS based FEC, we choose $c=100$, $u=5$, $v=8$ (i.e. RS code is based on $GF(2^8)$). Hence the length of each RS code is 150 symbols. Similarly for LDPC based FEC, we choose $c=4$, $u=1000$ and $v=1$ (i.e. LDPC code is a binary code). Hence the length of each LDPC code is 30000 symbols. Though our choice of code lengths is quite arbitrary, care has been taken to ensure that the performance trends are representative and the time-complexity of LDPC based FEC is lesser than the RS based FEC. The LDPC code we use in this section is a regular LDPC

code with 3 checks per variable bit. The parity check matrix was constructed using the Progressive Edge Growth Technique (PEG) [38], [39].

For RS based FEC we emulate the the decoding algorithms used for simultaneous erasure and error decoding by keeping a count of the number of symbols that are erased or in error in each channel code. In actual deployments an algorithm such as the Berlekamp-Massey Algorithm (see e.g. [32]) may be used to realize the actual decoding. However the exact method of RS decoding does not influence our conclusions and hence is outside the scope of this work. As against this the decoding algorithm used for LDPC is based on the Log-Likelihood Ratio (LLR) domain [37]5 implementation of the sum-product decoder. LDPC decoding in presence of erasures is achieved by setting the initial LLR value of the erased bit to zero. For the LDPC decoding, we used a "stopping criteria" of checking for a valid codeword after each iteration and the maximum number of iterations has been limited to 25.

Side-Information for Improved decoding efficiency:

Performance over CLDS can be improved by taking advantage of the side-information provided by CK-DATA for FEC decoding. This can be done as described below

(a) FEC block decoding failure: On occurrence of a block decoding failure the channel decoder is unable to recover all the dropped/corrupted packets. However as the FEC block is systematic, message packets that can be identified as uncorrupted can still be forwarded to the eventual application. In a cross layer design like CLD as we do not

---

[5] The software implementation used for the simulations in this section is a modified version of that provided at [38] - [39].

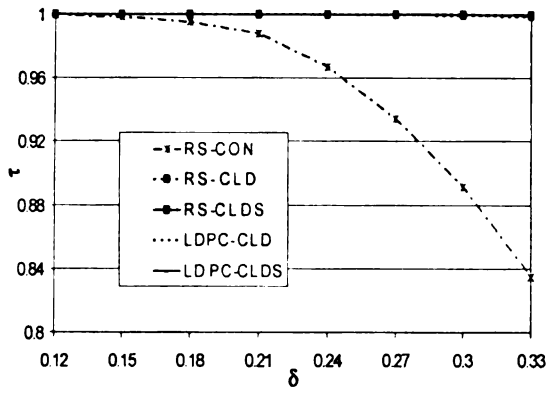
have information about CK-DATA, if a decoding failure is encountered, the entire packet block is dropped. However in a CLDS scheme the CK-DATA information is used to forward the correctly received message packets to the application layer.

(b) Apriori estimates for improved FEC decoding: CK-DATA side-information can be utilized to acquire improved apriori estimate of channel impairments and thus improve FEC decoding:

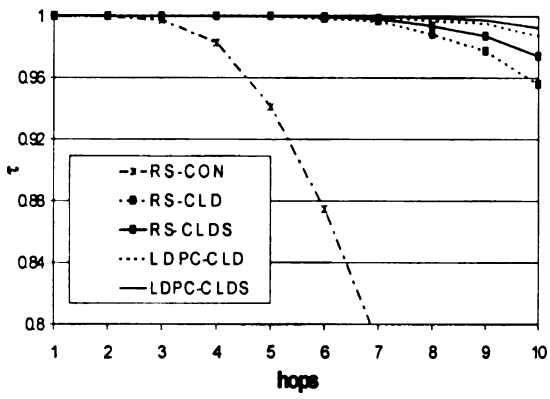
- In case of RS based FEC, we use this information rather simplistically, if the number of packets for which CK-DATA fails is less than the total number of redundant packets, then even if all the corrupted packets are treated as erasures the entire FEC packet block is recovered by pure erasure decoding. Thus the CLDS scheme can switch to the pure erasure decoding whenever possible to avoid decoding failure due to a high corruption level in the packets. Further improvements in performance can be obtained by employing soft-decoding of RS codes (e.g. see [35]). However RS based soft-decoding algorithms usually have higher time complexity and may not always be suitable for low-latency video applications.
- The sum-product LDPC decoding algorithm can also be modified to take advantage of the side information provided by CK-DATA. If the CK-DATA of a particular packet is satisfied than we set the apriori probability of the bits in this packet being in error to 0, this is achieved by setting the magnitude of the LLR to ∞.

Figure 7 shows the performance of various FEC and protocol combinations averaged over a transmission of 10,000 blocks of 30 packets. All the results in Figure 7 are expressed in terms of message throughput or goodput:

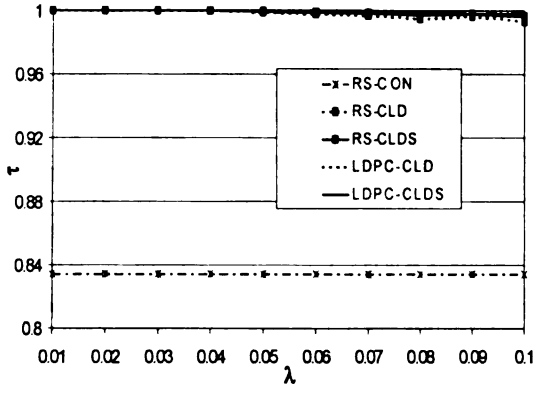
$$\tau = \frac{(message\ packets\ recieved\ after\ channel\ decoding)}{(message\ packets\ tranmitted)}$$



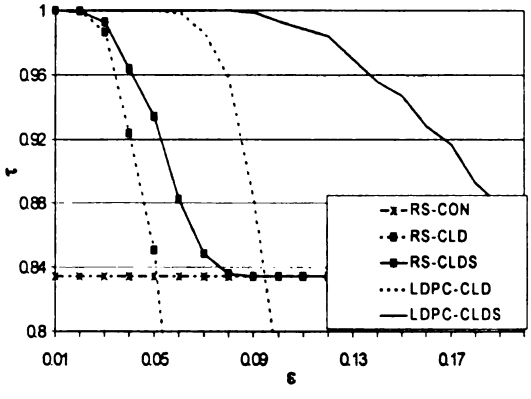
(a) $\lambda = 0.05$, $\varepsilon = 0.01$



(d) $\delta = 0.06$, $\lambda = 0.01$, ε=0.01



(b) $\delta = 0.33$, $\varepsilon = 0.01$



(c) $\delta = 0.33$, $\lambda = 0.01$

Figure B.7 Throughput performance of RS/LDPC based FEC schemes over CON/CLD/CLDS. In (a), (b), (c) the simulations are over a single hop and only one parameter is varied while in (d) all three parameters are maintained constant and performance scaling over multiple hops is recorded.

It can be observed that for a given FEC scheme, the performance of CLDS is better than both CON and CLD. In Figure 7 (a), it should be observed, that though the coding rate is well below capacity, the CON scheme is unable to provide 100% packet recovery, as against that it can be observed in Figure 7 (a), (b) and (d) that when the corruption

level is low, even the most straight forward of cross-layer combinations (i.e. RS-CLD), is sufficient to provide substantial improvement in throughput over conventional schemes and almost a 100% reliability. However, contrary to some of the previous works (e.g. [14]), Figure 7 (c) shows that, when the corruption level increases, the performance of RS-CLD (as well as LDPC-CLD) can actually drop well below the performance of the conventional scheme (CON). It can be observed that CK-DATA side-information is essential to guarantee that the performance of a HEEP is never worse than CON. A simplistic usage of the side-information as done by RS-CLDS is sufficient to provide improved performance (as can be observed for $\varepsilon > 0.01$ in Figure 7 (c)) and in addition to guarantee that a HEEP does not ever perform worse than a conventional scheme. (as can be observed for $\varepsilon > 0.09$ in Figure 7 (c))

Finally it should be noted that among all the considered combinations, LDPC-CLDS provides the best performance. The performance improvement on account of using LDPC-CLDS, as compared to other combinations, is especially high when the corruption level in the packets is high. Under poor channel conditions, the decoding algorithm fails to converge for the CLD scheme, while still successfully decoding in the case of the CLDS scheme. It was observed that, even under conditions where both LDPC-CLD and LDPC-CLDS are able to provide almost 100% reliability, the number of iterations used by the CLDS scheme are lesser. This can play a critical role, when the end-receiver is a low-power device. Detailed analysis of complexity versus throughput tradeoff is deemed outside the scope of this part.

3.4. Video Simulations

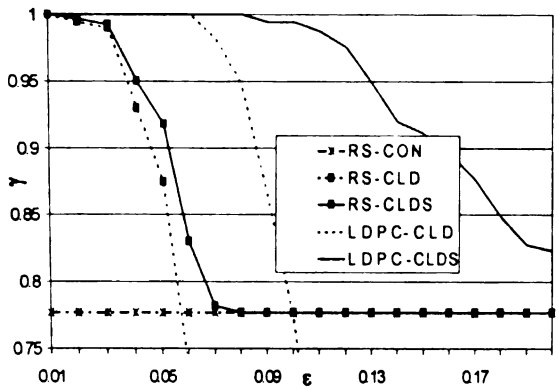
Discussions in previous sections have concentrated on exhibiting the capacity and packet throughput improvements that can be achieved by using cross-layer protocols. At this stage, it is necessary to clearly establish whether these improvements indeed translate into improvement in the quality of video available at the end receivers. Thus in this section we use the emerging H.264 [45] video standard to present some results based on video simulations.

The results presented in this section are a subset of the examples we considered and thus it should be stated that the choice of test sequence, quantization, frame frequency and frame size do not compromise on the generalness of the conclusions derived on the basis of the video analysis presented here. Thus here we present results based only on the "Stefan", "Carphone", "Paris", test sequences. A "CIF" (352x288) frame size and a frequency of 30 frames/sec have been chosen for all sequences. We used a constant quantization size of QP = 28 for all sequences. We used a GOP size of 15 frames and a GOP structure of IPPP.... It is also worth noting that all the encoded streams are made up of video packets of size 500 bytes each. The reference software version of H.264 used for our simulations was TML 9.0. The encoding of the sequence was RD-optimized for 30% losses to increase the error-resilience. The FEC encoding parameters used in this section are identical to those used in the previous section.

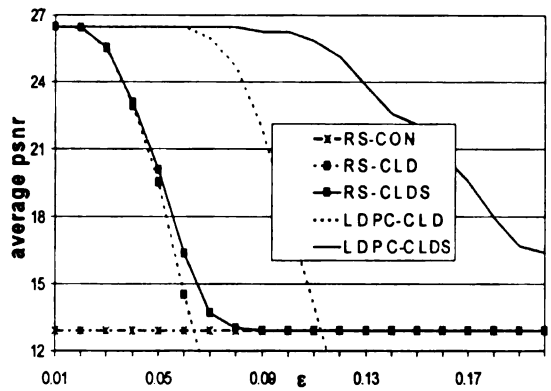
The video quality performance was studied under a variety of channel coding parameters and channel conditions. However in this section we present results based only on a sample of the various scenarios we considered. Thus we specifically focus on channel conditions, which were similar to the ones used to generate Figure 7 (c) and (d).

Results presented in Figure 8 evaluate the impact of corruption level on the video quality and correspond to channel conditions similar to those used in Figure 7 (c), while results presented in Figure 10 correspond to the channel conditions similar to those used in Figure 7 (d) and evaluate the degradation in video quality as the data is relayed over multiple hops. Each data point in Figure 8 and Figure 10 is obtained on the basis of averaging the performance over three repetitions of transmitting the first 300 frames of a test sequence. In each experiment and for each protocol-FEC combination we ensured that the first I frame is transmitted in a lossless manner.

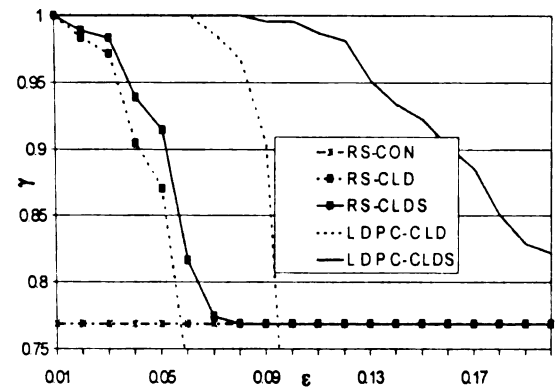
Under severe channel conditions, many of the picture frames are completely un-recovered. This leads to significant amount of motion discontinuity and also block-distortion due to lack of a good reference frame. Thus in Figure 8 (a), (b), (c) and Figure 10 (a), (b), (c) the motion continuity has been measured in terms of the proportion of successfully recovered picture frames γ; while the block-distortion in Figure 8 (d), (e), (f) and Figure 10 (d), (e), (f) has been calculated in terms of average PSNR, which has been obtained by assigning 0dB to the missing frames. Naturally, the presence of error concealment and improved error-resilience techniques can improve the video quality performance of each of the considered scheme. However, detailed analysis of all the currently available error concealment/resilience tools in H.264 is outside the scope of this part. Instead we focus on presenting the results as per the above described setup and expect that the trends in video quality degradation may be useful in design of future error-resilience and concealment features specifically catering to communication over HEEPs.



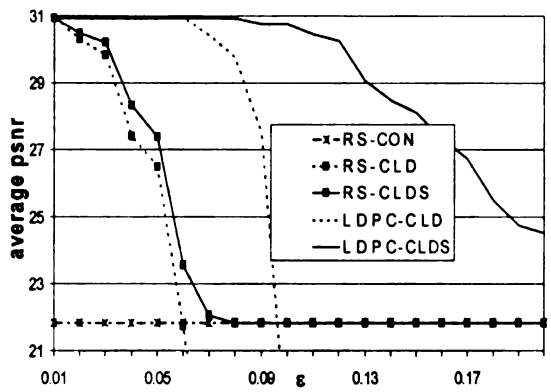
(a) Stefan



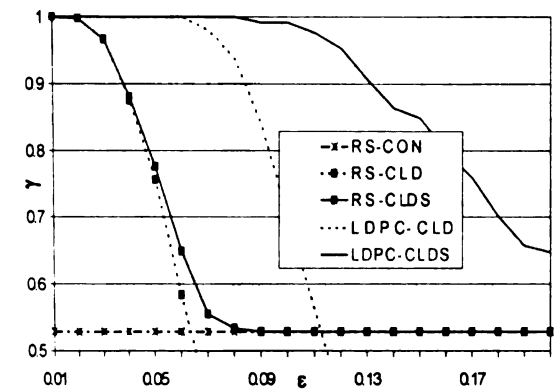
(d) Stefan



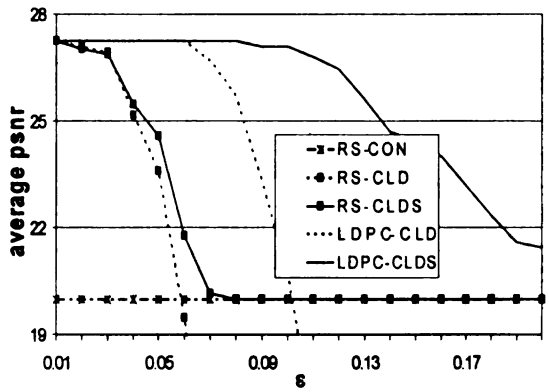
(b) Carphone



(e) Carphone



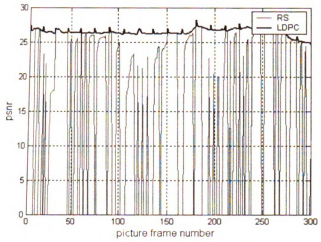
(c) Paris

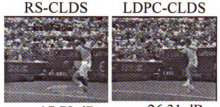


(f) Paris

Figure B.8 Impact of corruption level on the video quality performance. (a), (b), (c) show the motion continuity γ in terms of % picture frames recovered. (d), (e), (f) show the block-distortion. The simulations are over a single hop and the channel parameters are maintained constant at $\delta = 0.33$, λ=0.01.

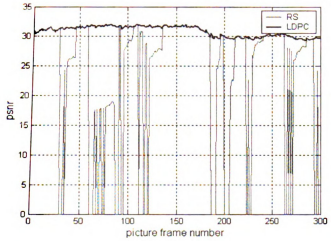
In Figure 8, the channel conditions ($\delta = 0.33$, $\lambda = 0.01$) render the operation of the CON scheme to be extremely close to capacity. Thus even when the corruption levels are low, the CON scheme provides a very poor video quality. As against that, at low corruption levels, all the FEC-HEEP combinations can provide a performance improvement in excess of 14 dB for Stefan, 9dB for Carphone and 7dB for Paris. The motion discontinuity for the CON scheme is also very high as more than 20% of the picture frames remain completely un-recovered. Since the decoding algorithm employed for RS is a hard-decoding algorithm, the impact of errors on RS based FEC is significant. With a slight increase in the corruption level, the performance of both RS-CLD and RS-CLDS drops drastically. It may be noted that the performance of RS-CLD can infact drop below RS-CON even at modestly high corruption levels. Thus we re-iterate that when CK-DATA checksum is completely turned off and the application layer FEC is an RS based hard-decoding algorithm, its essential to verify that the channel conditions do not render the corruption levels to be to high, else the performance of a HEEP may infact be worse than a conventional scheme. As against this, and as should have been anticipated, the performance of RS-CLDS never drops below RS-CON. The soft-decoding algorithm employed in conjunction with LDPC makes it more resilient to errors. However, after a certain corruption threshold, the performance of even LDPC-CLD drops below that of CON. As against this LDPC-CLDS can continue to provide performance improvement in video quality in excess of 5dB over all other schemes, even when the corruption level is as high as 15%.

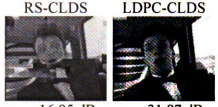




(d) Stefan: Subjective Comparison based on Frame 108

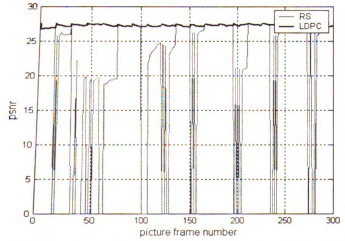
--(a) Stefan: Temporal Snapshot





(e) Carphone: Subjective Comparison based on Frame 64

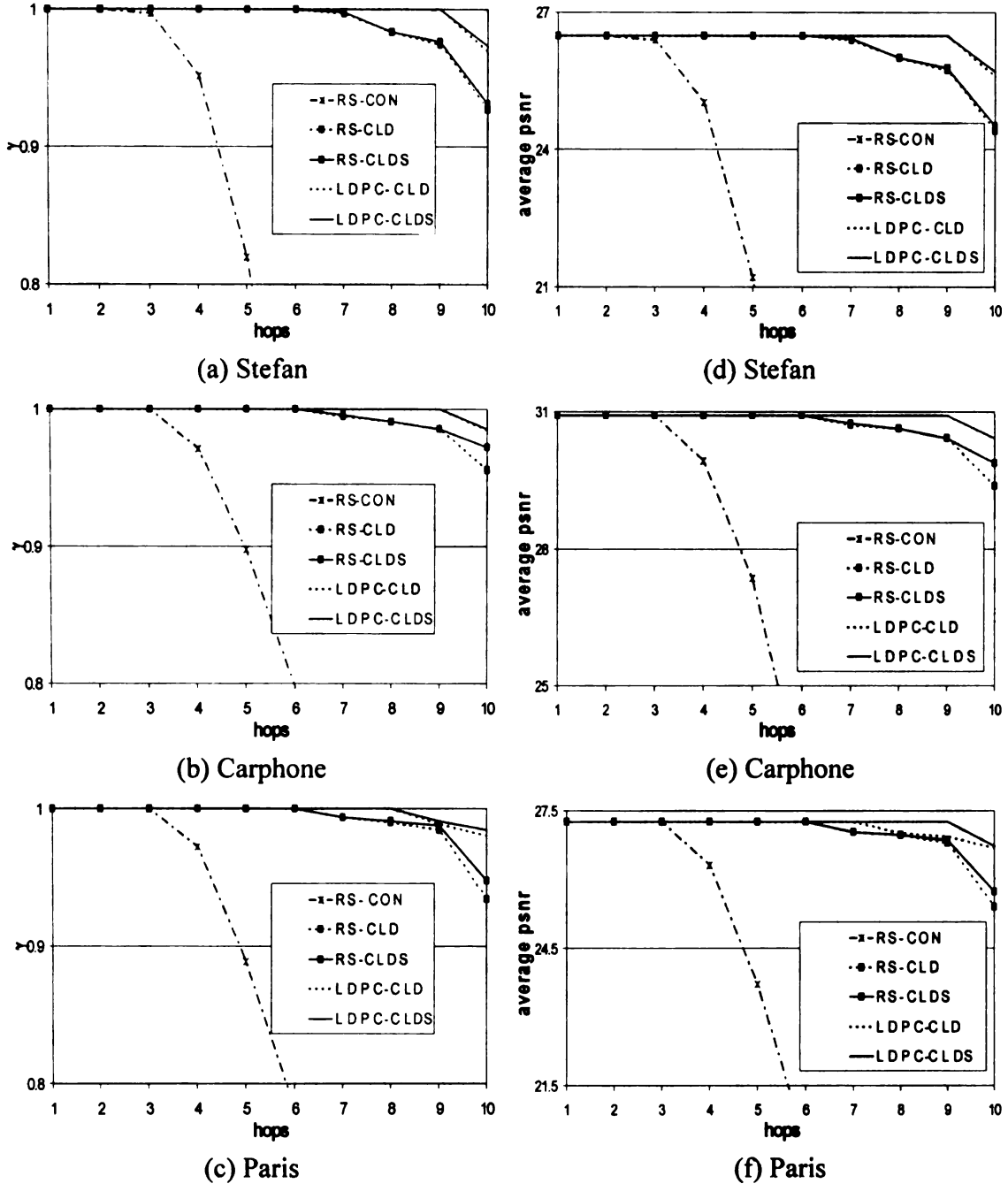
---(b) Carphone: Temporal Snapshot



(f) Paris: Subjective Comparison based on Frame 48

---(c) Paris: Temporal Snapshot

Figure B.9 Video quality Comparison between RS-CLDS and LDPC-CLDS on the basis of Temporal Snapshots (a)–(c) and picture frame samples (d)-(f).



(a) Stefan (d) Stefan

(b) Carphone (e) Carphone

(c) Paris (f) Paris

Figure B.10 Video quality scaling over multiple hops. (a), (b), (c) show the motion continuity γ in terms of % picture frames recovered. (d), (e), (f) show the block-distortion. The channel parameters for each hop are maintained at $\delta = 0.06$, λ=0.01, ε=0.01.

In Figure 9 we provide a detailed comparison between the video quality provided by RS-CLDS and LDPC-CLDS for channel conditions corresponding to $\delta = 0.33$, $\lambda = 0.01$, $\varepsilon = 0.07$, which represents a data point in Figure 8. From the temporal snapshots in Figure 9 (a), (b), (c) it can be observed that at high corruption levels the video quality for RS-CLDS can frequently drop below unacceptable levels. It should be noted that the video quality is able to recover, despite frequent deterioration, because source coding is RD-optimized for 30% losses. If the source coding is less robust/error resilient, the performance of RS-CLDS compared to LDPC-CLDS would be even worse. It should also be noted that in Figure 9 (a), (b), (c) the downward spikes that reach all the way upto 0 dB represent the loss of an entire picture frame. As has been elaborated before, loss of entire picture frames can lead to severe motion discontinuity. Thus the primary artifact, especially in a video sequence such as Stefan, was "motion discontinuity". However, even when a picture frame is not dropped entirely, severe block-distortion can be observed. Figure 9 (d), (e), (f) show subjective examples for comparing the block-distortion. For the particular images shown in Figure 9, the difference in video quality between RS-CLDS and LDPC-CLDS is greater than 8dB for Stefan, 14dB for Carphone and 8dB for Paris.

In Figure 10, the channel conditions on each hop are represented by $\delta = 0.06$, $\lambda = 0.01$, $\varepsilon = 0.01$. Thus the channel conditions on each hop are not very severe and it can be seen that, on the initial 2-3 hops all the schemes including CON can provide reasonable video quality. However, as the hops increase, the performance of the CON scheme drops drastically. It was observed that by hop 6, the performance of CON as compared to all the FEC-HEEP combinations is extremely poor. Thus, even if the

channel conditions on every single hop are not very bad, the cumulative effect of a multi-hop communication on a conventional scheme can be quite drastic. Hop 6 onwards all the FEC-HEEP combinations provide atleast 6dB improvement in video quality over the CON scheme. Though the corruption level in each packet is at $\varepsilon = 0.01$, the average bit error probability among all the un-erased packets is as low as $p = 0.0005$. A packet that is corrupted on one hop, does not necessarily get corrupted on another. Thus even as the number of hops increase, the average corruption level, even by the 10th hop does not increase drastically. Thus it can be observed that all the FEC-HEEP combinations continue to provide reasonable video quality even till the 10th hop. Never the less, by the 10th hop, the soft decoding algorithm employed in conjunction with LDPC allows it to provide 1-2dB performance improvement over all other FEC-HEEP combinations. The performance of LDPC-CLD and LDPC-CLDS is almost identical on all the hops, however if we keep increasing the number of hops over which the information is relayed, eventually LDPC-CLDS shall start performing better (and at times significantly better) than LDPC-CLD.

3.5. Subsidiary Discussions

3.5.1. Hop-by-Hop retransmissions

It should be noted that the deductions of channel capacities in Section 3.2 are not applicable to schemes that employ hop-hop retransmissions. Hop-by-Hop retransmissions represent a possible way of embedding error control mechanisms within the network. Though currently this is the most commonly deployed method, other methods to embed error control mechanisms within the network do exist. Thus a network architecture,

which allows embedded error control, needs separate attention and should be distinguished from end-to-end error control. To provide a fair comparison between the cross-layer approach and the conventional approach we should permit the existence of network embedded error control in a cross-layer architecture too. However detailed analysis of such schemes is outside the scope of this part. Meanwhile, it is worth noting the following:

a) The capacity of conventional channel employing hop-hop retransmissions is given by

$$C_{CON(retrans)} = 1 - \arg\max_{i\in[1,n]}(\delta_i) \tag{7}$$

b) Hop-Hop retransmissions can be employed in a cross-layer scenario too. One possible method to do so is to request retransmissions when the CK-HDR fails. The capacity of a cross-layer scheme employing hop-by-hop retransmissions as described above is

$$C_{CLD(retrans)} = \left(1 - \arg\max_{i\in[1,n]}(\lambda_i)\right)\cdot\left(1 - h_b\left(\overset{n}{\underset{i=1}{*}} p_i\right)\right) \tag{8}$$

$$C_{CLDS(retrans)} = \left(\prod_{i=1}^{n}(1-\delta_i)\right) + \left(\left(1 - \arg\max_{i\in[1,n]}(\lambda_i)\right) - \left(\prod_{i=1}^{n}(1-\delta_i)\right)\right)\cdot$$
$$\left(1 - h_b\left(\left(\frac{\left(1 - \arg\max_{i\in[1,n]}(\lambda_i)\right)}{\left(\left(1 - \arg\max_{i\in[1,n]}(\lambda_i)\right) - \left(\prod_{i=1}^{n}(1-\delta_i)\right)\right)}\right)\cdot\left(\overset{n}{\underset{i=1}{*}} p_i\right)\right)\right) \tag{9}$$

c) Wireless LAN's usually consist of a single wireless hop. Thus for most WLANs it should be expected that the reliability/throughput offered by the considered cross-layer

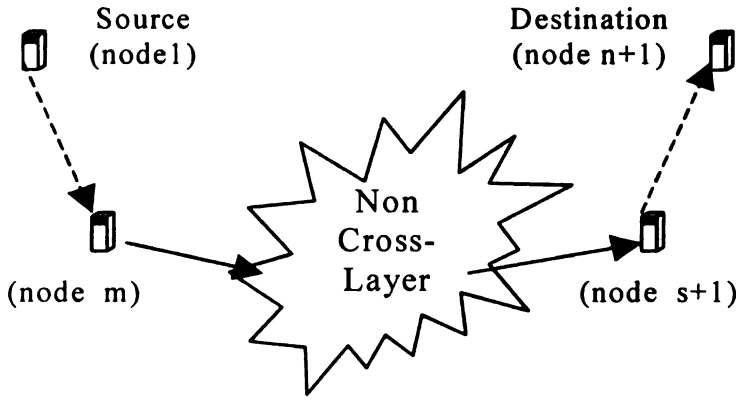


Figure B.11 A Hybrid Network. In the above figure the links outside the star-cloud are cross-layer enabled but the links inside are not.

schemes in conjunction with FEC should be even better than a retransmission based scheme on the conventional stack.

3.5.2. Hybrid Architecture

Throughout this part we assume that the entire network path from the source to the destination is capable of supporting a cross-layer approach. However, as shown in Figure 11 that need not be the case always. If we assume that the total path consists of n hops; and hops m to s lie inside the "non cross-layer" section then the expression for capacity deduction would get modified as described below:

$$C_{CLD(n-hop)} = \left(\prod_{i=1}^{s}(1-\delta_i)\right)\cdot\left(\prod_{i=s+1}^{n}(1-\lambda_i)\right)\cdot\left(1-h_b\left(\overset{n}{\underset{i=s+1}{*}} p_i\right)\right) \tag{10}$$

$$C_{CLDS(n\text{-}hop)} = \left(\prod_{i=1}^{n}(1-\delta_i)\right)+\left(\prod_{i=1}^{s}(1-\delta_i)\right)\left(\left(\prod_{i=s+1}^{n}(1-\lambda_i)\right)-\left(\prod_{i=s+1}^{n}(1-\delta_i)\right)\right)$$

$$\left(1-h_b\left(\left(\frac{\left(\prod_{i=s+1}^{n}(1-\lambda_i)\right)}{\left(\prod_{i=s+1}^{n}(1-\lambda_i)\right)-\left(\prod_{i=s+1}^{n}(1-\delta_i)\right)}\right)\cdot\left(\overset{n}{\underset{i=s+1}{*}} p_i\right)\right)\right) \tag{11}$$

Thus it should be noted that the capacity improvement on account of a cross-layer approach can be severely diminished over multiple hops. This can be avoided by updating all the CK-DATA's at the last node (in this case node m) before the non cross-layer section begins. In addition, a header field that keeps a count of the checksum updates should be included. At the end receiver the "number of updates" field in addition to the updated CK-DATA can be used to determine the corruption status of the payload.

3.5.3. Impact of Link-level Channel on Capacity Deductions

In general δ, λ, ε are not independent of each other and are in fact functions of the underlying link-layer channel. Assuming a specific link-layer channel can indeed lead to more precise conclusions and render some of the regions (combinations of δ, λ, p) unachievable. In this section we consider simple example channels to exhibit as to how a specific link-layer channel model can facilitate more precise conclusions.

Link Layer Binary Symmetric Channel:

If we assume the link-layer channel to be a BSC channel with cross-over probability q and, let h and d represent the number of header and payload bits respectively, then we can say $\delta = 1-(1-q)^{(h+d)}$, $\lambda = 1-(1-q)^{h}$ and $\varepsilon = q$. Thus, the channel capacities are:

$$C_{CON} = (1-q)^{(h+d)} \tag{12}$$

$$C_{CLD} = (1-q)^{h} \cdot \left(1 - h_b\left(\left(1-(1-q)^{d}\right)\cdot q\right)\right) \tag{13}$$

$$C_{CLDS} = (1-q)^{h} \cdot \left(1 - \left(\left(1-(1-q)^{d}\right)\cdot h_b(q)\right)\right) \tag{14}$$

Similar to proposition 2, we can deduce a threshold $q_{\min}$, which divides the parameter space into 2 halves. Note that this is equivalent to deducing a threshold $d_{\min}$, since for a

given q and h as the payload size increases the susceptibility of the CON scheme relative to CLD increases. The following proposition proves the existence of such a threshold.

Proposition 3: $\forall\, q \in (0,0.5)$, $\exists\, d_{\min} \geq 0$ s.t., $d \geq d_{\min} \Leftrightarrow C_{CLD} \geq C_{CON}$. Furthurmore the equality occurs iff $d = d_{\min}$.

Proof: Note that $\left(1-(1-q)^d\right) \geq h_b\left(\left(1-(1-q)^d\right)\cdot q\right) \Leftrightarrow C_{CLD} \geq C_{CON}$. The above proposition can be proved by showing that the equation $\left(1-(1-q)^d\right) = h_b\left(\left(1-(1-q)^d\right)\cdot q\right)$ can at maximum have a single non-negative solution d_{sol}. Moreover it should be observed that $\lim_{d\to\infty} h_b\left(\left(1-(1-q)^d\right)\cdot q\right) = h_b(q)$ which is less than $\lim_{d\to\infty}\left(1-(1-q)^d\right) = 1 \;\; \forall\, q \in (0,0.5)$. Thus if a solution to the above equation does not exist it can be concluded that $C_{CLD} \geq C_{CON}$, while if a solution does exist than $\forall\, d > d_{sol}$, $\left(1-(1-q)^d\right) > h_b\left(\left(1-(1-q)^d\right)\cdot q\right)$.

The value of $d_{\min}$ even for $q = 0.4$ (value of $d_{\min}$ is strictly monotonic with q) is less than 20 bytes. Thus if the link-layer channel resembles a BSC the CLD scheme is better than CON almost always.

Link Layer 2- state Channel:

The deduction made above can be generalized by considering a channel over which all the packet transmissions are not necessarily susceptible to errors. Such a behavior is possible on time-varying channels and over wireless networks with dynamic topologies. Thus let's consider a channel over which only a fraction η number of the total

transmitted packets are susceptible to errors. When packet transmissions are indeed susceptible to errors, the behavior of the link-layer channel can be assumed to be equivalent to a BSC with crossover probability of θ. For such a channel it can be shown that $\varepsilon = \theta$, $\delta = \eta - (1-\theta)^{(h+d)}$ and $\lambda = \eta - (1-\theta)^{h}$. Thus it can be shown that the channel capacities of the three schemes are

$$C_{CON} = 1 - \eta + (1-\theta)^{(h+d)} \tag{15}$$

$$C_{CLD} = \left(1 - \eta + (1-\theta)^{h}\right) \cdot \left(1 - h_b\left(\frac{(1-\theta)^{h} \cdot \left(1-(1-\theta)^{d}\right) \cdot \theta}{1 - \eta + (1-\theta)^{h}}\right)\right) \tag{16}$$

$$C_{CLDS} = \begin{pmatrix} \left(1 - \eta + (1-\theta)^{(h+d)}\right) + \\ (1-\theta)^{h} \cdot \left(1-(1-\theta)^{d}\right) \cdot \left(1 - h_b(\theta)\right) \end{pmatrix} \tag{17}$$

It's again evident that a cross-layer approach can provide improvements in capacity. It should be noted that the expressions for the BSC example could be obtained from the above equations by setting $\eta = 1$. For channels with memory the capacity is determined by the available state information. If no state information is available then the capacity is the least ([41]-[42]). Thus equation (15), (16) and (17) can be considered to be lower bounds on capacity of link-layer channels with memory.

3.6. Conclusions

In this sub-part we analyzed the tradeoff involved in allowing the relay of corrupted packets to the application layer. Channel conditions under which HEEPs can provide improvement were identified. The capacity improvements were quantified and it was established that a dramatic increase in capacity can be achieved in many realistic

scenarios by using HEEPs. Similarly, performance of RS and LDPC based FEC in conjunction with HEEPs was shown to be significantly better than RS based FEC over conventional protocols (CON) under several channel conditions. H.264 based video simulations were used to clearly establish that improvement in capacity and packet throughput does actually translate into significant improvement in video quality. It was observed that video quality improvement provided by HEEPs can be dramatic and in excess of 6-10dB under several channel conditions.

It was observed that the performance of HEEPs can be affected by packet drops due to header corruption and increase in the bit error rate of the corrupted packets. It was shown that at high corruption levels the side-information provided by CK-DATA can improve the performance of HEEPs significantly. Combination of LDPC and CLDS was observed to provide the best performance. At high corruption levels absence of CK-DATA based side information render a CLD like HEEP to perform worse than CON.

Thus, on the basis of our observations, it can be recommended that:

- HEEPs should be further explored and deployed as they have the potential to be extremely useful in dramatically increasing the throughput of video applications
- Future designs of HEEPs should:
  - increase header robustness
  - not discard any channel state information from lower layers, which can be utilized as side-information at higher layers
  - use error control algorithms capable of efficiently utilizing the side-information from lower layers

B 4. UTILITY OF PRACTICALLY OBSERVABLE VARIABLES IN HEEPS

4.1. Motivation

In wireless networks the capacity of a link can vary with space and time. The link-capacity and quality plays an important role in determining the resource allocation in wireless networks. For example, consider the work by Majumdar et. al. in [46] related to design of unequal error protection (UEP) for video multicast. The design algorithm presented in [46] requires knowledge about the link-capacities of the users. Thus knowledge about link-capacity is useful for network planning. Additionally, since the link-quality can vary with time, channel state information (CSI) available on a more realtime basis can be utilized for network adaptation.

The above discussion highlights the importance of estimating link-quality. Under the traditional link-abstraction, the channel impairments observed at the link-level consist only of packet erasures. Thus the link quality is completely determined by the Packet Erasure Rate (PER), δ. In practice, it is easy to identify the packets that are dropped and hence the PER also can be easily determined. Consequently, a number of link-quality based optimizations can be readily employed with the traditional protocols. However, when cross-layer protocols are employed, the link-level channel is impaired by bit-corruptions. Thus the channel capacity is determined by the PER as well as the Bit Error Rate (BER) in the corrupted packets. The difficulty of determining the BER emerges from the fact that unlike a packet drop, a bit corruption is not "observable", i.e. a decoder is not aware of the exact location or count of bit corruptions. Additionally, the bits at the

link-level are simple ones and zeros; these bits do not have any inherent soft-information, such as the signal-strength, associated with them. Hence, it is essential to identify a practical mechanism to estimate the BER in the corrupted packets. Addressing this important problem is the primary focus of this chapter.

As has been elaborated above, in practice, the signal-strength of each bit or the BER is not observable. However, there may exist other variables such as the traffic load, the PER, checksum etc. which are observable. The correlation between these observable variables and the BER can thus be exploited for channel estimation. Hence, identification of suitable observable variables is one of the most important aspects of realizing a practical mechanism for link-quality estimation. The work in this chapter is motivated by the observation that commercial 802.11b devices are capable of associating a coarse measure, Signal-to-Silence Ratio (SSR), with each received packet. The SSR value is evaluated on the basis of a Received Signal Strength Indication (RSSI) which measures the signal strength during the preamble stage of the packet reception, and the Silence Value which measures the signal strength just prior to the packet reception. The objective of the presented work is to determine the efficacy of the CSI provided by these coarse SSR indications. The key steps in the contributions of this work are:

- **Correlation Modeling:** As the first step, we seek to establish a relationship between SSR indications and BER. In principle, we can analytically deduce such a relationship by assuming some channel model and device specific detector model (i.e. a model for the physical layer processing done at an 802.11b wireless receiver). Such an analytical approach is critically dependent on the accuracy of the device model. Accurate device models are typically hard to determine and additionally cumbersome to analyze. Hence,

in practice it is significantly easier to empirically identify the behavior of a device. Therefore we utilize actual 802.11b measurements to establish a device specific empirical relationship between SSR and BER. In general, the determination of such an empirical relationship has a receiver-specific training overhead associated with it. However our analysis will show that there exist scenarios where the empirical relationship can be determined in a reasonably link/infrastructure independent fashion. Therefore, based on our observations, we believe that, under a number of practical scenarios, the training overhead can be kept within modest limits

- **Cross-layer Network Planning**: The above determined relationship can be used to model the link-level error behavior with a Compound Binary Symmetric Channel (CBSC). The capacity of the CBSC can thus be used to estimate the cross-layer link capacity. These capacity estimates can in turn be used for cross-layer network planning. In general cross-layer network planning can include optimization of unequal error protection, retransmission limits, bandwidth and other resource allocations. Nevertheless, in this work we limit our attention to a simplistic yet representative illustration of the predictive utility of SSRs in network planning. With the example of a Reed-Solomon (RS) based FEC scheme, we show that SSR indications can be used in a non-realtime fashion to predict (a) the suitability of a channel coding rate for an individual or group of receivers, (b) the utility of cross-layer protocols for various receivers

- **Decode-side CSI:** In a WLAN, the radio-link quality can vary significantly on account of multi-path reflections, fading, receiver mobility, interference and variations in the surroundings. In such environments the BER in each packet can vary significantly. Information about the BER in a packet can help in enhancing the error recovery

performance. In the last chapter, we exhibited that even simplistic CSI, available from checksums, can significantly improve the efficiency of cross-layer protocols. In this work, we show that SSR indications can be used to provide further performance gains. We demonstrate this utility by considering an architecture where Low Density Parity Check (LDPC) code based FEC is used for transport of H.264 compressed video. On the basis of trace based simulations, we show that the CSI available due to SSR indications can significantly improve the error recovery performance of LDPC decoding algorithms. This improved error recovery performance is shown to provide substantial gains in multimedia quality, especially when the variations in link-quality are significant.

The organization of the remainder of this chapter is as follows: Section 4.2 develops important theoretical preliminaries to facilitate an easy understanding of the discussions in the following sections. Section 4.3 provides details about the methodology employed to collect the wireless error traces. It is important to highlight that all claims in this chapter have been validated on the basis of simulations with actual 802.11b wireless error traces and actual measurements of the SSR. In Section 4.4 we deduce an empirical relationship between SSR indications and the probability of bit error in a corrupted packet. This relationship is used to deduce SSR dependent capacity measures. We provide experimental and theoretical analysis exhibiting the efficacy of using these capacity measures for network planning. In Section 4.5 we explore the utility of SSR as a realtime predictive tool. We show that CSI available from SSRs can improve capacity. Additionally, on the basis of trace based simulations, we also exhibit the utility of SSRs in improving the error recovery performance and the multimedia quality. Finally in Section 2.6, we provide a summary of some of the key conclusions of this work.

4.2. Preliminaries

4.2.1. CBSC Model for MAC-to-MAC Channels

In this work we model the channel from the Medium Access (MAC) layer of the transmitter to that of the receiver as a CBSC. This model is used in the error recovery algorithms and also for estimating the link-capacity Thus, in this work; we define a CBSC as follows:

Definition 1: A Compound Binary Symmetric Channel (CBSC) is a discrete memory less symmetric channel defined over a binary input ($X = [0,1]$) and binary output ($Y = [0,1]$) alphabet. The channel behavior is described by a probability mass function (p.m.f) $f(s)$ on a set of states S, where each state $s \in S$ corresponds to a BSC with cross-over probability ε_s.

In the subsequent sections, it will be shown that we can associate a SSR indication and a checksum indication Z with each received packet. Here Z is just a binary indication such that Z = 0 when the packet is corrupted and Z = 1 when the packet does not contain any bit-corruptions. We show that the (Z, SSR) tuple can be used to define the states of a CBSC channel. Furthermore, on the basis of actual measurements, we train each of the (Z, SSR)-BSC states, i.e. we associate a probability of bit error $\varepsilon_{(Z,SSR)}$ with each state. These trained models are then used to describe the channel behavior observed by each receiver in terms of an equivalent CBSC. We argue that the BSC parameter $\varepsilon_{(Z,SSR)}$ associated with each state (Z, SSR) can be assumed to be (link) invariant. Consequently,

the state definitions of the equivalent CBSC models remain identical for all the receivers. However the p.m.f governing the frequency of the states can vary. In practice, we can observe the frequency of the (Z, SSR) indications to determine the p.m.f $f(\cdot)$ associated with each receiver. As shall be shown in the following sub-section, this p.m.f can be used along with the cross-over probabilities of each state to determine the link-capacity. In section 4.4 we show that such capacity estimates can be used for network planning. Additionally, since each packet has an (Z, SSR) indication associated with it, our trained models can also be used in real-time for estimating the channel state in each packet. In section 4.5 we use such real-time CSI to associate soft-confidence values with the received bits. These soft-confidence values are then used to improve the error recovery performance of LDPC decoding algorithms.

4.2.2. Capacity of CBSC and the role of side-information

In the absence of any channel state information, the CBSC reduces to a BSC with cross-over probability $p = \sum_{s \in S} f(s) \cdot \varepsilon_s$, which represents the cross-over probability averaged over all the states. Thus the capacity of a CBSC is given by

$$C = 1 - h_b(p) = 1 - h_b\left(\sum_{s \in S} f(s)\varepsilon_s\right) \tag{18}$$

However, it can be shown that in presence of decoder-side channel state information the capacity increases. We refer the reader to the work in [43] for detailed discussion on capacity in presence of side-information. As discussed in [43], the core insight behind evaluating the capacity of a CBSC in presence of decoder-side CSI is that its capacity is

equivalent to a channel whose input is given by $x \in X$ while the output is given by $(s, y) \in S \times Y$. Thus, the capacity of CBSC with decoder-side channel state information is given by

$$C_{CSI} = \sum_{s \in S} f(s)\left(1 - h_b(\varepsilon_s)\right) \tag{19}$$

Furthermore, in practice, under certain protocol designs, the CSI may be available to a receiver in a lumped form. For example, in the previous chapter we utilize only the checksum as side-information. This is equivalent to the information about states $(Z = 0, SSR)$ being lumped into a single a state (Z = 0) and that of states $(Z = 1, SSR)$ being lumped into a state (Z = 1). The lumped states can be represented by a decomposition of the set S into mutually disjoint and collectively exhaustive subsets $\{S_i\}$, i.e. $S_i \subseteq S$, $\bigcup_i S_i = S$, $S_i \cap S_j = \phi$. Thus, when CSI is available to a receiver in a lumped form the decoder does not have exact state information, but is aware of the subset S_i to which the current channel state belongs. The probability of occurrence of a lumped state can be expressed as $f(S_i) = \sum_{s \in S_i \subseteq S} f(s)$. Similarly the average probability of bit-error in each lumped state can be expressed as $\varepsilon_{S_i} = \sum_{s \in S_i \subseteq S} f(s)\varepsilon_s$. The capacity of CBSC with such lumped side-information is equivalent to the capacity of a CBSC channel with states $\{S_i\}$ and perfect decoder-side CSI. Hence the capacity is:

$$C_{CSI-lumped} = \sum_i \left(\left(\sum_{s \in S_i} f(s) \right) \left(1 - h_b \left(\sum_{s \in S_i} f(s)\varepsilon_s \right) \right) \right) = \sum_i \left(f(S_i)\left(1 - h_b\left(\varepsilon_{S_i}\right)\right) \right) \tag{20}$$

At this point, it should be noted that the capacity function is a convex function. Jensen's Inequality [40] states that for any convex function $g(\cdot)$, $E[g(\cdot)] \geq g(E[\cdot])$ with equality occurring if and only if $g(\cdot)$ is constant over the entire domain. Thus with the help of Jensen's Inequality we can easily show that capacity of CBSC increases with decoder side CSI. The capacity gain is reduced if the state information is lumped. Therefore we can write:

$$C_{CSI} \geq C_{CSI-lumped} \geq C \tag{21}$$

where $C_{CSI} = C_{CSI-lumped}$ if and only if all the states in every lumped state are identical, i.e. $\forall\, i,a,b$ such that $a,b \in S_i$ we have $\varepsilon_a = \varepsilon_b = \varepsilon_{S_i}$. Similarly, $C_{CSI} = C_{CSI-lumped} = C$ if and only if all the states of a CBSC have an identically description. Consequently, the gains provided by side-information increase with variations in channel behavior, i.e. with variations in the cross-over probability.

4.3. 802.11b Measurements

To demonstrate the efficacy of the schemes proposed in this work, we base our analysis on actual wireless error traces. (These traces build upon on the data set used in [25]). As shown in Figure 12 our experimental setup consisted of an 802.11b Access Point (AP) operating in Distributed Coordination Function (DCF) mode with RF output power set at 18dBm. A station is connected to the AP using 100Mbps Ethernet and acts like a server, while another wireless station serves as a Line of Sight (LoS) client. The LoS client is placed in proximity to the AP to avoid packet drops and hence retransmissions. We utilize a third wireless station to collect the required traces. This station acts as "sniffer"

and overhears all the packet transmissions to the LoS client. At the sniffer we used a DWL 122 wireless card based on Prism 2.5 chipset with linux-wlan-ng-0.2.1pre21 device driver [27]-[29]. The Prism based card enables operation in "monitor" mode which enables delivery of all the MAC frames without any filtering, including those with failed Frame Check Sequence (FCS). The source code of the "sniffer's" device driver was further modified to dump all the corrupted/uncorrupted frames from the kernel buffer space. The traces collected at the sniffer and the LoS clients were compared offline to obtain the error trace. We embedded specific bit patterns in each transmitted packet to enable the identification of all the corrupted packets. Packet dissectors were implemented to ensure that only packets pertinent to our wireless experiment are processed.

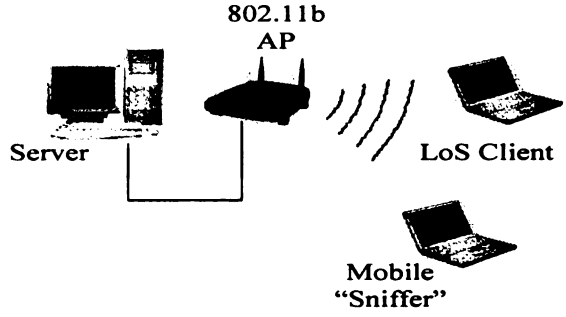
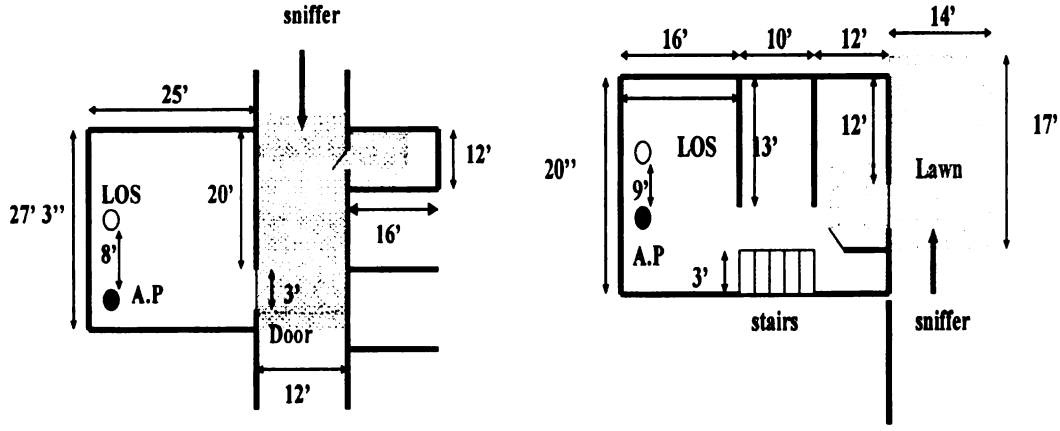


Figure B.12 Trace collection setup



(a) Office Setup (b) Home Setup

Figure B.13 Trace collection environments

The Prism2.5 device measures Received Signal Strength Indication (RSSI) value and "Silence" value at the antenna of the radio hardware. The RSSI is measured for 10μs6 during the preamble stage of MAC frame reception. The Silence value measures power before the start of the frame. In the radio device that we used, both these values are recorded in a single byte representing the signal strength in dbm. We define the SSR to be the difference between the RSSI value and the Silence value. Thus we associate an SSR measure in dB with each of the received packets. In addition, we also associate a binary indication Z with each packet. The value of Z is determined on the basis of a checksum failure. As mentioned previously, we set Z = 0 if the packet is corrupted and Z = 1 if the packet is uncorrupted.

The work environments where traces were collected have been classified into "Home" and "Office" setup. This nomenclature is primarily determined by location of the author's home and office. Both these locations were separated by a distance greater than 3 miles and thus there wireless environments are assumed to be completely uncorrelated. To assist further visualization, Figure 13 provides a rough sketch of both the wireless environments:

Home Setup: This setup consisted of an almost interference free home environment with one or two neighboring APs on (widely separated) channels. The specific house in which we collected the traces is a two-storey town house with a basement. The house is located at a corner and thus the interfering APs (if any) were at a significant distance from the

---

[6] The duration for which the signal strength is measured is determined by the radio-hardware and cannot be altered.

sniffer. The walls of the house were wooden and hence as shown in Figure 13 the sniffer machine and the AP were separated by 0 to 3 wood walls.

Office Setup: Our office setup is located in the Engineering Building at Michigan State University. Thus our collection setup was surrounded by 3 to 4 neighboring APs. During our trace collection, we ensured that none of these APs were operating on a channel identical to our experiment. However, due to the high density of the APs, it was infeasible to completely avoid co-channel interference.

The traces utilized in this chapter are a part of a larger effort to collect wireless measurements. However for the purpose of this draft we consider about 2 million packets collected in each of the setups. These packets were transmitted using the 802.11b PHY data rate of 11Mbps. The packet transmission rate was maintained at 384 packets/sec in the Home Setup and at nearly 1000 packets/sec in the Office Setup. Thus we collected traces for a total time of ~87 minutes at home and ~34 minutes at the office. These traces were not collected in a contiguous manner. The total trace collection time was broken into contiguous collection times varying between 3 to12 mins. In each of the contiguous periods, the traces were collected by a human subject holding the laptop at waist height. The human subject had a meandering mobility (less than 2-10feet/minute).

In the subsequent sections, we exhibit that the utility of cross-layer protocols is the most pronounced when the SSR values are in the range of 8-14dB. We deduced this fact from some preliminary analysis and thus a deliberate effort was taken to concentrate most of our trace collection effort in this SSR range.

4.4. Cross-layer Network Planning

In this section we evaluate the empirical relationship between SSR indications and the link-level channel impairments. This relationship is utilized to associate a protocol dependent capacity measure with each SSR value. We then show that such capacity measures can be used to provide predictive utility for network planning.

4.4.1. Relationship of SSR with Link-level Impairments:

The states of the CBSC are defined on the basis of the (Z,SSR) tuple. Thus in this section we are primarily interested in associating a BSC parameter $\varepsilon_{(Z,SSR)}$ with each of the states. In addition, we shall also be interested in evaluating the conditional probability $\delta_{(SSR)}$ of a packet being corrupted given an SSR value. This conditional measure will be utilized in the following sub-section, to establish a relationship between capacity and SSR. Note that $\delta_{(SSR)}$ represents the probability of a packet being dropped by a traditional protocol.

Our trace collection methodology associates a (Z,SSR) -tuple with each received packet. Hence the packets in our traces can be classified on the basis of the (Z,SSR) label. Thus $\varepsilon_{(Z,SSR)}$ can be obtained by averaging the observed BERs over all the packets with a given (Z,SSR) label. Additionally, we evaluate the frequency $f(\cdot)$ of each state (Z,SSR). These frequencies can then be used to determine the $\delta_{(SSR)}$ in a straight forward manner using the following expression:

$$\delta_{(SSR)} = \frac{f(Z=0,SSR)}{f(Z=0,SSR)+f(Z=1,SSR)} \tag{22}$$

Loss Rate: Figure 14 shows the result of empirically evaluating $\delta_{(SSR)}$. It can be observed that above 14dB the probability of packet being corrupted is less than 10%, while below 8dB the probability of a packet being corrupted is almost 100%. Thus we refer to SSR values below 8dB as the bad range, the values above 14dB as the good range and the values from 8 to 14dB as the transition range. Cross-layer protocols are required only when a significant number of packets get dropped due to bit corruptions. Hence from here on we focus our analysis primarily on channel condition where the SSR value is less than 15dB.

Bit-Error Rate: A checksum indication of Z = 1implies that a packet is error free. Hence, the BSC parameter associated with each state $(Z=1, SSR)$ is set to zero. Consequently, for notational convenience we shall often represent the BSC parameter $\varepsilon_{(Z=0,SSR)}$ by $\varepsilon_{(SSR)}$. Figure 15 shows the result of empirically evaluating $\varepsilon_{(SSR)}$.

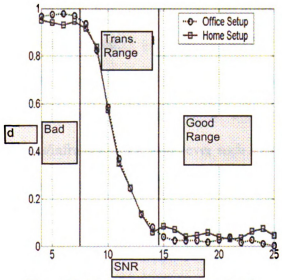


Figure B.14 Empirically determined relationship $\delta_{(SSR)}$

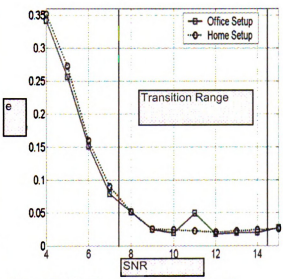


Figure B.15 Empirically determined relationship $\varepsilon_{(SSR)}$

From Figure 14 and Figure 15 it can be observed that the relationships $\delta_{(SSR)}$ and $\varepsilon_{(SSR)}$ are reasonably infrastructure invariant. Such infrastructure invariance may not be valid for all possible classes of home and office environments. However, we would like to highlight that the Home and Office setups chosen for this chapter have crucial differences:

- The topology of these setups, determined by the size of the rooms, walls and other objects are completely distinct. (For example, in the office environment, there is a significant number of metal objects, such as desks, file cabinets, etc, which cause significantly higher number of reflections and multi-path effects when compared to the "Home" environment.)
- The packet transmission rates (used to collect the error trace) are also completely distinct.
- The number of interfering APs surrounding our setups are very different
- A distance greater than 3 miles separates our home and office setups. Hence we believe that the wireless environments of these infrastructures are completely uncorrelated.

Therefore, on the basis of empirical evaluations, we believe that in practice it should still be feasible to determine a common single model for a large variety of links/infrastructures. Whenever such a situation is not feasible, we recommend the use of an infrastructure specific channel model.

4.4.2. Capacity gain of cross-layer protocols:

The above derived empirical relationships can be used to estimate the capacity gain provided by cross-layer protocols at various SSR values. As discussed previously, we determine the inherent link-level capacity on the basis of an equivalent CBSC. Thus, if we assume that the SSR indications observed on a link remain constant then the frequency of states $(Z=0,SSR)$ and $(Z=1,SSR)$ in an equivalent CBSC are given by $\delta_{(SSR)}$ and $1-\delta_{(SSR)}$ respectively. Therefore, in accordance with equation (1), the inherent link-level capacity is given by

$$C_{cross-layer}(ssr)=1-h_b\left(\sum_{(Z,ssr)} f(Z,ssr)\varepsilon_{(Z,ssr)}\right)=1-h_b\left(\delta_{(SSR)}\varepsilon_{(SSR)}\right) \qquad (23)$$

In general cross-layer protocols cannot recover information from all the corrupted packets. Under certain circumstance the identity of packet cannot be established and hence the packet has to be dropped. Hence in general the capacity of cross-layer protocols is lesser than the inherent capacity. However, techniques such Header Detection [22] can be used to reduce the proportion of such packet drops to negligible levels. Therefore in this chapter we assume that the capacity of cross-layer protocols is equivalent to the inherent capacity. Furthermore, note that the capacity under a traditional protocol (i.e. a traditional link-layer abstraction) is completely determined by the loss rate. Therefore, as a function of SSR, the capacity of a traditional protocol is given by:

$$C_{traditional}(ssr)=\left(1-\delta_{(SSR)}\right) \qquad (24)$$

Figure 16 shows the capacities evaluated on the basis of the above equations. It can be observed that the cross-layer protocols provide negligible capacity gain in the good range and for SSR values lesser than 5dB. However, for the transition range and parts of the bad range (i.e. from 5-15dB) the increase in capacity can vary from a minimum of 0.05 to a maximum in excess of 0.5. Thus it is evident that the utility of employing a cross-layer protocol can vary significantly with change in the average SSR. Such variations in utility have to be taken into consideration while planning a network.

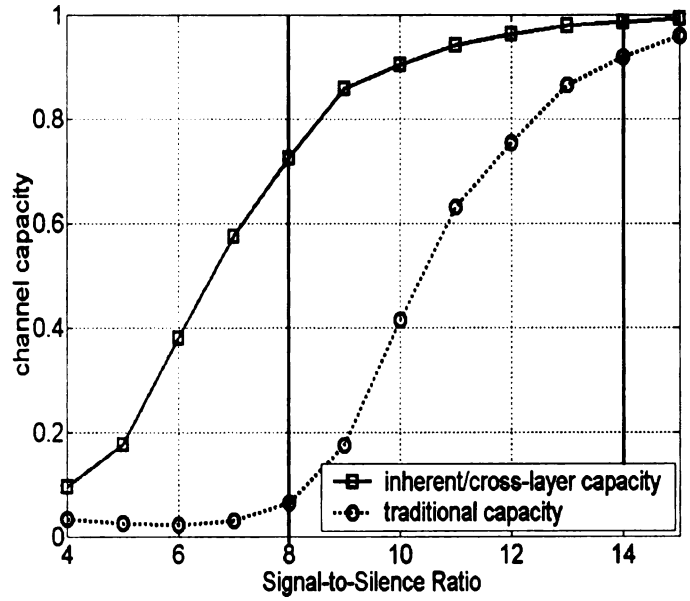


Figure B.16 Capacity gain due to cross-layer protocols

4.4.3. Cross-layer network planning

In this section, we demonstrate the utility of SSR as a predictive tool for network planning. To do so we emulate the communication setup described by Figure 17. In Figure 17 a server is transmitting (video) application packets, protected by a systematic Reed-Solomon (RS) code based FEC, to receivers placed in diverse locations in a

network. The channel between the transmitter and various receivers is simulated on the basis of 4 trace samples each, of 20, 000 packets, collected in the Home and Office setup. The traces from the Home Setup are represented by (H1, H2, H3, H4), while those from the Office Setup are represented by (O1, O2, O3, O4). Our objective is to show that the SSR statistics of each trace are sufficient to predict (a) the suitability of the chosen channel coding rate (b) the necessity for employing cross-layer protocols.

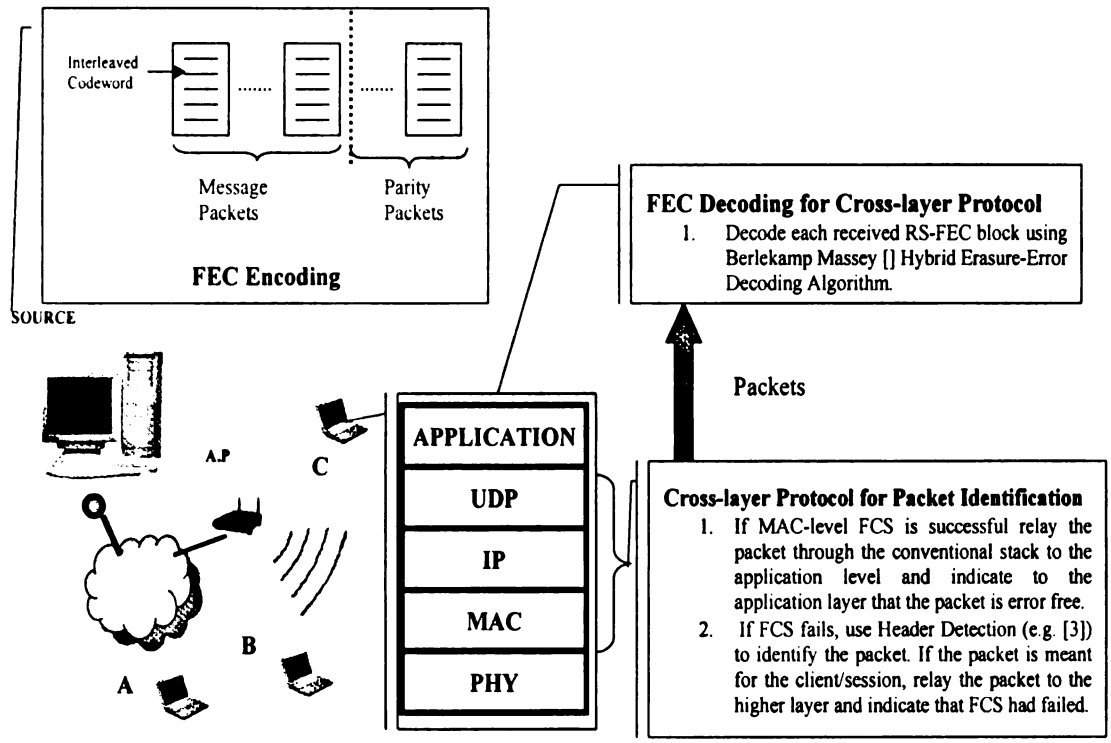


Figure B.17 System Model for Cross-layer Packet recovery

The FEC schemes used in this chapter are realized by interleaving codewords across packets. These FEC schemes can be completely characterized by, N the packet block length, the code length n, code rate R, the packet size s in bits and the symbol size b (in bits) on which the code has been implemented. Thus each FEC block consists of $(N.R)$ message packets and $(N \cdot s)/(b.n)$ codewords. The parameters chosen for the RS

based FEC used in this section are $N=100$, $n=200$, $R=0.8$, $s=8192$ and $b=8$. We evaluate the performance of such a FEC scheme, in conjunction with cross-layer as well as conventional protocols. The decoding algorithm used in conjunction with a traditional protocol is an erasure decoding algorithm, while that employed with cross-layer protocols is a hybrid erasure-error decoding algorithm. Neither of these algorithms utilize any realtime CSI. Table I shows the performance of the FEC scheme in terms of $goodput = 1 - P_d$ where P_d is the probability of error in decoding a codeword.

Table B.I Protocol dependent goodput of RS based FEC for various 802.11b traces

	H1	H2	H3	H4	O1	O2	O3	O4
Traditional	52%	99%	11%	74%	74%	40%	92%	34%
Cross-layer	75%	99%	24%	81%	83%	62%	94%	47%
Improvement	28%	0%	13%	7%	9%	22%	2%	13%

Table B.II Distribution of packets in different SSR ranges for various 802.11b traces

	H1	H2	H3	H4	O1	O2	O3	O4
bad	1.8%	0.2%	32.6%	6.8%	2.8%	12.2%	1.4%	13%
good	37.4%	97.5%	28%	66.9%	66.7%	16.5%	91.3%	43%
transition	60.8%	2.3%	39.4%	26.3%	30.5%	71.3%	7.3%	44%

From Table I it can be clearly seen that the utility of a cross-layer approach can vary greatly. Observations on traces:

(a) H1 and O2 exhibit significant utility for cross-layer protocols.

(b) H4 and O1 would lead to a claim of moderate improvement.

(c) H3 and O4 also show moderate improvement, but the performance of both protocols is poor.

(d) H2 and O3 would lead us to believe that there is no need for cross-layer protocols. Studies prior to this work (e.g. [14]) have completely ignored the role of SSR or SNR in influencing the utility of cross-layer protocols. Cross-layer always add additional architectural and processing complexity. A system administrator or network designer may choose to employ cross-layer protocols only when necessary, or may choose to place a workstation with higher computational capability in locations where cross-layer protocols are essential. Thus variations in performance, exhibited by the RS based FEC, if not explained/predicted appropriately may make the task of cross-layer network planning very cumbersome.

The relationship established in the previous subsection can now be used to predict/explain the variation in the utility of cross-layer protocols. On the basis of Figure 16, we should not expect any advantage in employing cross-layer protocols if a large percentage of the packets are in the good and bad range. In other words, for cross-layer protocols to be of any practical utility, they should be employed only when the channel conditions are primarily in the transition range. Moreover, even though cross-layer protocols provide capacity gains in a part of the bad range, this performance gain will not

be realized with a coding rate of 0.8. Consequently from Table II it can be clearly seen that cross-layer protocols when employed over traces:

(a) H1 and O2 show significant benefit in performance because almost 2/3 of the packet blocks are in the transition range

(b) H4 and O1 show moderate benefit in performance because the number of packets in the transition range is lesser (about 1/3)

(c) H2 and O3 show negligible improvement in performance because almost all packets are in the good range.

(d) H3 and O4 provide a performance better than traditional. However the goodput is still very poor because a large percentage of the packets are in the bad range.

Thus with an appropriate site-survey a network designer can clearly identify the utility of employing cross-layer protocols in different locations in the network.

4.5. Utility of SSR as a Real-time Predictive Tool

The channel conditions observed on a wireless link can frequently deviate from the average behavior. Consequently the channel impairments endured even by temporally adjacent packets can vary significantly. Our discussion in section 2.2 implies that the decoder-side CSI can lead to significant capacity gains under such varying channel conditions. In the previous chapter we have considered a cross-layer scheme which utilizes only the checksum indication Z as side-information. We refer to such a scheme as an "SSR-unaware" cross-layer scheme. In this section we show that CSI available from SSR indications can provide additional capacity gains. We employ LDPC code based FEC simulations on our wireless traces to exhibit the improvement in error recovery. Finally, H.264 based video simulations are used to show the benefits.

4.5.1. Cross-layer capacity gain due to side-information:

An SSR-aware scheme, utilizes the (Z, SSR) tuple as side-information. From equation (19) of section 4.2 we can express the capacity of an SSR-aware protocol as:

$$C_{SSR-Aware} = \sum_{(Z,SSR)} f(Z,SSR)\left(1-h_b\left(\varepsilon_{(Z,SSR)}\right)\right)$$
$$= \sum_{SSR} f(Z=1,SSR) + \sum_{SSR} f(Z=0,SSR)\left(1-h_b\left(\varepsilon_{(SSR)}\right)\right) \quad (25)$$

Meanwhile, an SSR-unaware scheme only utilizes the indication Z as side-information. This is equivalent to utilizing the (Z, SSR) information in a lumped form, where the lumped states are represented by $Z_0 = \{(Z,SSR)|Z=0\}$ and $Z_1 = \{(Z,SSR)|Z=1\}$. Thus with the help of equation (20) from section 4.2 we can express the capacity of an SSR-unaware protocol as:

$$C_{SSR-unware} = \sum_{i=0}^{1} f(Z_i)\left(1-h_b\left(\sum_{(Z,SSR)\in Z_i} f(Z,SSR)\varepsilon_{(SSR,Z)}\right)\right)$$
$$= f(Z_1) + \left(1-h_b\left(\sum_{SSR} f(Z=0,SSR)\varepsilon_{(SSR)}\right)\right) \quad (26)$$

Inequality (21) in section 4.2 directly implies that $C_{SSR-aware} \geq C_{SSR-unaware}$. Additionally, comparing (25) and (26) it can be deduced that the difference in capacity gain due to SSR will increase when (a) $f(SSR, Z=0)$ are not negligible (i.e. a

significant number of the packets are corrupted) and (b) when there is a substantial variation in SSR indications. To clearly illustrate this fact we consider an example:

Example 1: Let us assume that a link has an average SSR value of 10dB and the variations in link-quality can be expressed by the following truncated Gaussian distribution on the SSR indications:

$$if \quad 4 \le SSR \le 16 \quad f(SSR) = \frac{\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{(SSR-10)}{2\sigma^2} \right)} \right)}{\sum_{SSR=4}^{16} \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{(SSR-10)}{2\sigma^2} \right)} \right)} \quad , else \; f(SSR) = 0$$

Note that $f(Z=0, SSR) = \delta_{(SSR)} f(SSR)$ and $f(Z=1, SSR) = \left(1 - \delta_{(SSR)}\right) f(SSR)$.

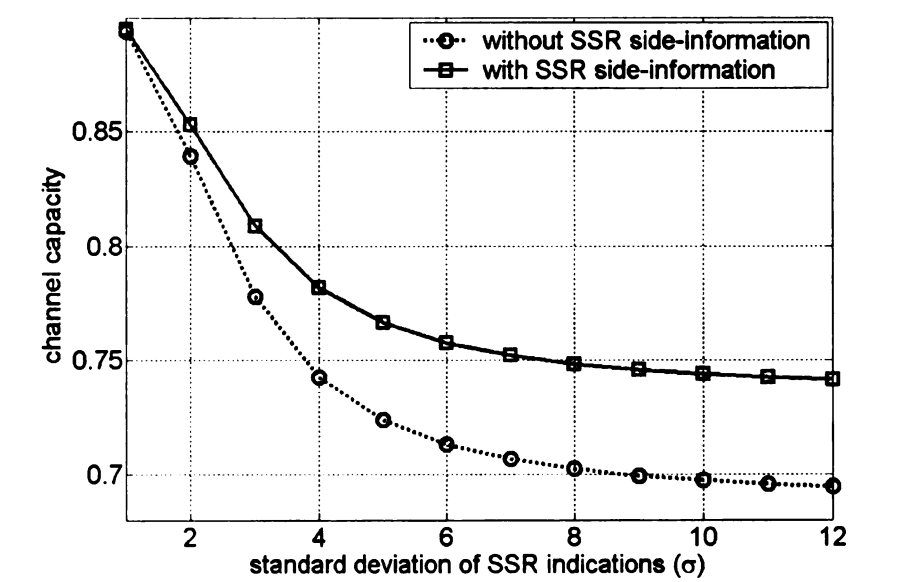


Figure B.18 Capacity gain due to side-information

Thus we can evaluate the capacity of the SSR-aware and SSR-unaware scheme with the help of (25), (26) and the empirical relationships established in the previous section. Figure 18 plots the capacities for the considered example. It can be clearly seen that the capacity gain increases as the standard deviation σ increases.

The above conclusions can be formalized with the help of the following formula:

Holder's Defect Formula ([44]):

If $g:[a,b]\longrightarrow \mathbf{R}$ is twice differentiable and if we have the bounds

$$0 \le m \le g''(x) \le M \quad for\ all\ x \in [a,b]$$

then for any real values $a \le x_1 \le x_2 \le \cdots \le x_n \le b$ and any non-negative reals p_k, $k = 1,2,\ldots,n$ with $\sum_{i=1}^{n} p_i = 1$, there exists a real value $\mu \in [m,M]$ for which one has the formula $\sum_{k=1}^{n} p_k g(x_k) - g\left(\sum_{k=1}^{n} p_k x_k\right) = \frac{1}{4}\mu \sum_{j=1}^{n}\sum_{k=1}^{n} p_j p_k \left(x_j - x_k\right)^2$

Let's choose $p_k = f(SSR = k | Z = 0) = f(SSR = k)\,\delta_{(SSR=k)} / f(Z_0)$ where

$f(Z_0) = \sum_{k=1}^{14} f(Z = 0, SSR = k)$, $x_k = \varepsilon_{(Z=0,SSR=k)} = \varepsilon_{(SSR=k)}$, $g(x_k) = 1 - h_b(x_k)$

and $[a,b] \equiv [0,0.5]$. For such a choice we have $m = 4$ and hence

$$\begin{pmatrix} C_{SSR-aware} \\ -C_{SSR-unaware} \end{pmatrix} \ge f(Z_0) \sum_{j=1}^{14}\sum_{k=1}^{14} \left(f(SSR_j | Z = 0) f(SSR_k | Z = 0) \left(\varepsilon_{(SSR=j)} - \varepsilon_{(SSR=k)}\right)^2 \right)$$

$$\Rightarrow C_{SSR-aware} - C_{SSR-unaware} \ge f(Z_0) \cdot \mathrm{var}\left(\varepsilon_{(SSR)} | Z = 0\right)$$

where, $f(Z_0)$ is the probability of a packet being corrupted and $\text{var}\left(\varepsilon_{(SSR)}|Z=0\right)$ is the conditional variance of the cross-over probability $\varepsilon_{(SSR)}$. At this point, we assume that the relationship between $\varepsilon_{(SSR)}$ and SSR is convex, then by re-applying Holder's formula we get

$$\Rightarrow C_{SSR-aware} - C_{SSR-unaware} \geq f(Z_0) \cdot \mu^{"} \cdot \text{var}\left(SSR|Z=0\right)$$

where $\mu^{"} \approx \min_k \left(0.25 \cdot \left(\varepsilon_{(k+2)} + \varepsilon_{(k-2)} - 2\varepsilon_{(k)}\right)\right)$ (numerical approximation of the second derivative).

From the above discussion we should expect the SSR aware scheme to provide significant performance gains when the SSR values lie primarily in the transition range and exhibit substantial variations. Our simulations in the subsequent subsections further validate this claim.

4.5.2. Trace-based simulations

High-level System Description and Experimental Methodology:

To illustrate the practical realtime utility of an SSR_aware scheme we consider a Video Communication System described by Figure 19. In the system described in Figure 19 a remote-server multicasts H.264 compressed video using an LDPC code based FEC scheme. A receiver can employ either an SSR_aware or an SSR_unaware cross-layer scheme. We compare the performance of these schemes by employing trace based simulations. In accordance to the discussion in the previous sub-section, we focus our analysis on traces that contain at least 20% packets in the transition range. We chose 6

traces (3 from each environment) of 20,000 packets each to emulate the wireless channel. In addition, we use the following FEC and video specifications:

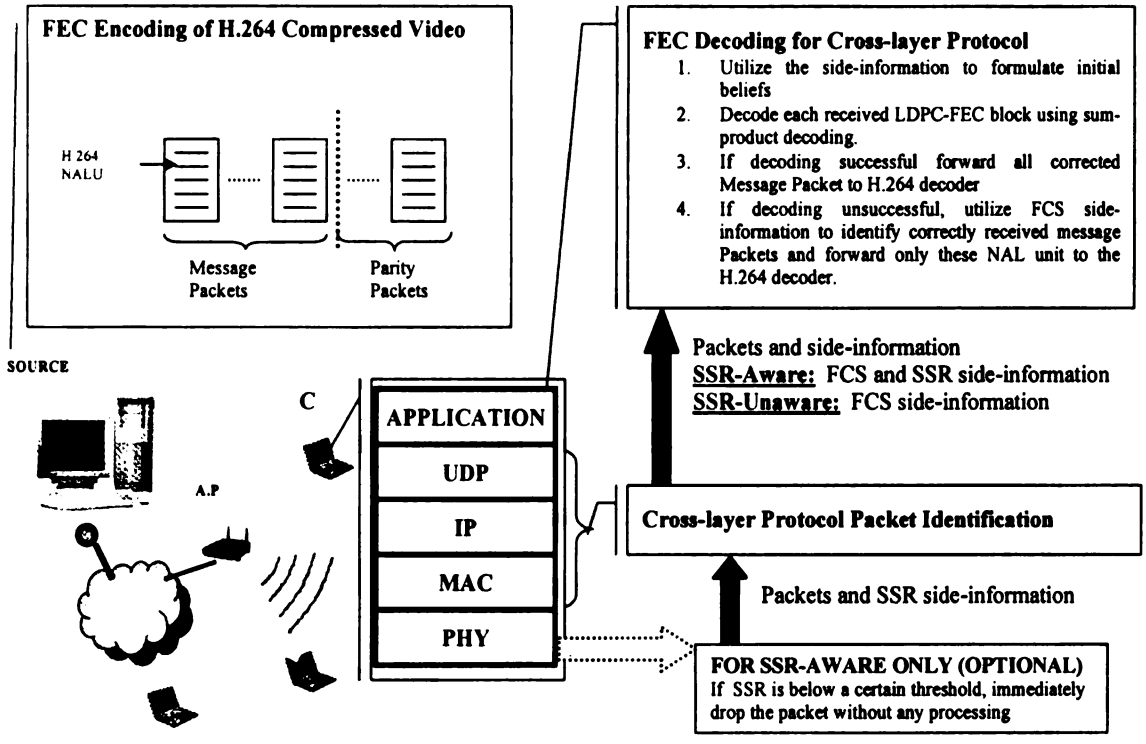


Figure B.19 System Model for Cross-layer Packet Recovery with Side-Information

FEC: The Progressive-Edge-Growth (PEG) algorithm based on large-girth tanner graphs was used to determine check matrices for all the LDPC codes [38] used in our FEC scheme. The channel codes we employed consisted of 8192 message bits and the redundancy was increased in steps of 512 bits. Thus we studied the performance of codes from rate R= 0.94 to 0.5, packet block length N = 136 to 256, code length n = 8704 to 16384. We consider only binary codes, thus b = 2. As our trace consists of a packet

payload of 1024 bytes, s = 8192 bits. Thus the total LDPC codewords in a single FEC block were 128.

Video Encoding: For the purpose of our video simulations we used streams compressed using H.264 (version 9.1 of the reference software) [30]. To emulate conditions where the wireless traces were collected we use video streams with a bit-rate of 2.1 Mbps for the simulations on Home Data, and bit-rate of 5.4 Mbps[7] for simulations on Office Data. We report results for test sequences "Mobile" and "Stefan". The resolution of these sequences was "CIF". We use a GOP (IBPBP) size of 30 frames, frame frequency of 30 fps. Test sequences are repeated thrice to provide a long enough playout sequences. The compressed stream was mapped to the FEC scheme by embedding a Network Adaptation layer Unit (NALU) in each interleaved codeword as shown in Figure 19. All standard error concealment features were turned on. In the event of data/frame loss the previous correctly received frame was copied and also used as a reference for the motion vectors.

Utilizing CSI for LDPC decoding:

An accurate initialization of LLR plays a key role in the performance of LDPC codes and thus improved CSI can help in improving the LDPC decoding. As described in section II, we can utilize the CSI to improve the accuracy of the LLR initializations. In this work, we assume that state of the channel remains identical throughout the packet.

---

[7] The source bitrates we have chosen do not necessarily represent practical choices. Our choice of bitrates was purely based on a motive to provide a single contiguous source ("compressed video + FEC" source) that could occupy almost all the available bandwidth.

Thus (Z, SSR) tuple indication can be used to associate a channel state with each bit in the packet. Depending upon the cross-layer methodology employed at the receiver these channel states can be used for LLR initialization in the following manner:

- For the SSR_aware scheme if a bit is received in state $(Z=0, SSR)$ set $L(c) := (-1)^{y} \log\left(\frac{1-\varepsilon_{SSR}}{\varepsilon_{SSR}}\right)$
- For the SSR_unaware scheme if a bit is received in state $(Z=0, SSR)$ set $L(c) := (-1)^{y} \log\left(\frac{1-\varepsilon_{Z_0}}{\varepsilon_{Z_0}}\right)$ where $\varepsilon_{Z_0} = \sum_{SSR} f(SSR, Z=0)\varepsilon_{(SSR)}$. Note that, the value of ε_{Z_0} depends upon the frequencies $f(SSR, Z=0)$. These frequencies vary with each link and trace. Hence for the purpose of our simulations we empirically deduce these frequencies for each trace.
- For both the schemes, if the bit is received in state $(Z=1, SSR)$ set $L(c) := (-1)^{y} \infty$. In practice we replace the infinity with a large finite value.
- Similarly, for both the schemes, if the bit belongs to an erased packet, set $L(c) := 0$

Results and Analysis:

The results of our experiments are tabulated in Table III. The observations in Table III are are similar to the capacity predictions made in the previous sections. It can be observed that, the performance improvements provided by the SSR_aware scheme are negligible in the good and bad SSR ranges. Never the less, in the transition range, the SSR_aware scheme consistently provides an improved error recovery performance. The

code failures are reduced by a minimum of 1.2% to a maximum of 11%. It is also important to note that the performance improvement provided by the SSR_aware scheme increases with variations in the SSR values. The standard deviation σ of the SSR values was found to be at least 3dB in all the traces presented here[8]. The significance of σ can be especially appreciated by observing the error recovery performance for trace 6. In trace 6, even though the average channel conditions are "good", the variation in SSR value is significant enough for an SSR_aware scheme to exhibit improved error recovery performance.

Table III also shows the improvement of video quality in terms of PSNR. It was observed that the average PSNR quality over 900 frames typically improves by 1-2db. However, in many instances, for large periods in the traces the SSR is entirely in the good range. Thus the average PSNR does not tell the complete story. Hence for a better evaluation, Figure 20 (a), Figure 21 (a), Figure 22 (a) and Figure 23 (a) provide temporal snapshots for experiments on trace 1 and 4.

In the temporal snapshots it can be clearly observed that there can be multiple GOPS over which the PSNR of an SSR_aware scheme is better than an SSR_unaware scheme by 5-15dB. Such a big difference in video quality is perceptually visible and a typical difference in block-distortion in the two schemes is exhibited by the frame captures in Figure 20 (b), Figure 21 (b), Figure 22 (b) and Figure 23 (b). The SSR_aware scheme

---

[8] In our extensive experimentation with 802.11b, over millions of packets, we have observed that SSR deviation is always significant. This deviation can be observed to increase due to presence of walls or (even limited) mobility.

Figure 22 (b) and 6dB in Figure 23 (b). In addition to the block distortion, the motion discontinuity in an SSR_unaware scheme was also significantly higher. Thus it can be clearly observed that under dynamic channel conditions, SSR can be used as a realtime predictive tool to significantly improve the performance of cross-layer multimedia protocols.

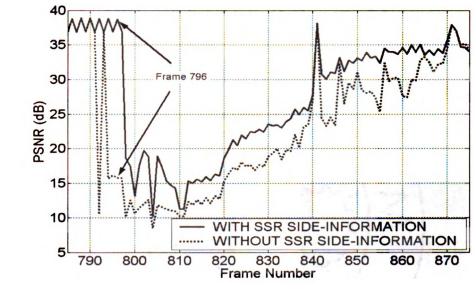
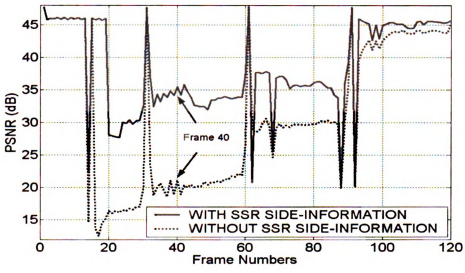


Figure B.20 Results of transmitting Stefan 2.1Mbps video over trace 4. (a) temporal snapshot (b) frame 796 corresponding to the temporal snapshot. Left: Current Scheme with SSR based CSI, Right: Previous Scheme.

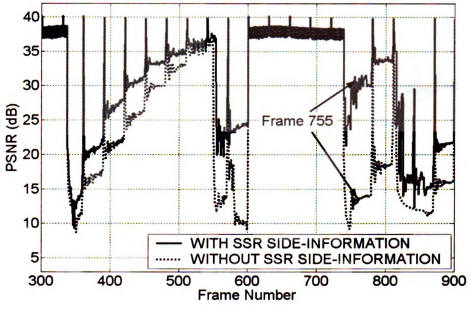


(a)

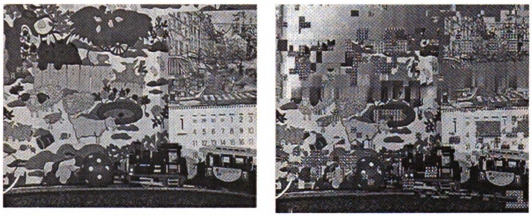
35.47 dB **21.09 dB**

(b)

Figure B.21 Results of transmitting Stefan 5.4Mbps video over trace 4. (a) temporal snapshot (b) frame 40 corresponding to the temporal snapshot. Left: Current Scheme with SSR based CSI, Right: Previous Scheme.



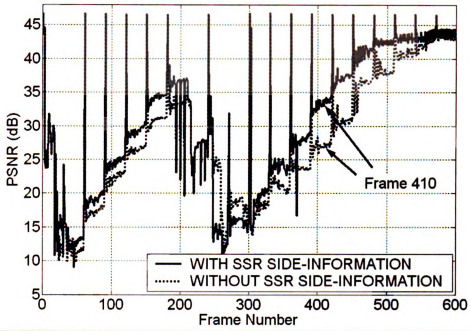
(a)



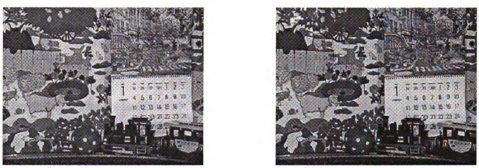
31.38 dB **13.30 dB**

(b)

Figure B.22 Results of transmitting Mobile 2.1Mbps video over trace 1. (a) temporal snapshot (b) frame 800 corresponding to the temporal snapshot. Left: Current Scheme with SSR based CSI, Right: Previous Scheme.



(a)



33.93 dB **27.57 dB**

(b)

Figure B.23 Results of transmitting Mobile 5.4Mbps video over trace 1. (a) temporal snapshot (b) frame 410 corresponding to the temporal snapshot. Left: Current Scheme with SSR based CSI, Right: Previous Scheme.

Table B.III Improvement in Cross-layer Performance: Error Recovery and Video Quality

Trace		1	2	3	4	5	6
WLAN Infrastructure		Home Setup			Office Setup		
Standard Deviation in the observed SSR values (dB)		3.2	3.2	7.7	3.8	4.1	7
Channel Coding Rate		0.8	0.66	0.94	0.8	0.66	0.95
Source Bit-rate		2.1 Mbps			5.4 Mbps		
Percentage Code Failures[⊥][⌜]	**good**	-	3.6	5.4	0.0	0.0	3.8
	SSR Range	-	5	7.0	0.0	0.0	6.0
	transition	0.5	4.0	14	7	1.9	-
	SSR Range	2.5	7.8	16	18	3.1	-
	bad	-	-	-	100	100	-
	SSR Range	-	-	-	100	100	-
	Total	0.5	3.9	11	6	1.7	3.8
	SSR Range	2.5	7.5	13	16	2.8	6.0
Ave. PSNR GAIN(dB)[⌝]	**Stefan**	5.5	0.93	1.19	1.6	0.78	0.82
	Mobile	6.78	0.75	0	2.3	2.07	1.37

[⊥]In the above table the column entry of *percentage code failures* has been divided into sub-columns good, transition and bad. The results in these sub-columns are obtained by classifying an FEC packet block based on the SSR averaged over the block.

[⌜] The highlighted sub-columns present the results corresponding to the "*SSR_Unaware*" scheme.

[⌝] The PSNR gain describe the average improvement in video quality provided by the "*SSR_aware*" scheme. The PSNR has been averaged over 900 frames. Thus in certain cases, despite the existence of specific GOPs where the gain was in excess of 10dB, the average gain is negligible.

4.6. Conclusions

In this chapter, with the help of analysis and experimentation with actual 802.11b traces, we have exhibited the predictive utility of SSR values in the design of cross-layer protocols. SSR values can be used on the basis of site-survey techniques to facilitate a Cross-layer Network Plan. Moreover, it has also been shown that SSR values can be used on realtime basis, to provide vital CSI for individual packets. Such realtime CSI can be used to improve the error recovery under dynamic channel conditions. Under certain realistic wireless channel conditions where the link-quality has significant variations, use of SSR can lead to an improvement in video quality in excess of 5-10 dB. Having established the predictive utility of SSRs, in future, we intend to use this tool to help design cross-layer equivalents of a variety of standard multimedia communication protocols. We consider one such example in the following chapter.

B 5. CROSS-LAYER INFORMATION EXCHANGE

5.1. Motivation

Network Coding (NC) has found a wide variety of applications since it was introduced by Ahlswede et. al. [48]. Many recent research efforts have shown significant utility for combining NC with the broadcast nature of wireless networks (e.g. [52], [54]). Majority of the current studies, focus their analysis on packet networks, where the only possible channel impairment (if any) is a packet drop. Nevertheless some recent studies (e.g. [53]) have shown that coding may provide performance gains even on noisy channels, i.e. on channels that lead to symmetric errors. In this chapter we extend this analysis to channels which exhibit time-varying behavior. In paritcular we develop an Adaptive Network Coding scheme for the Mutual Information Exchange Problem.

5.1.1. Network Coding for Mutual Exchange of Information

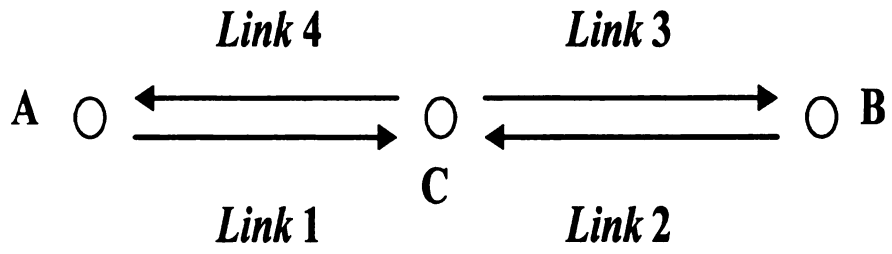


Figure B.24 Mutual Exchange Between Two Nodes

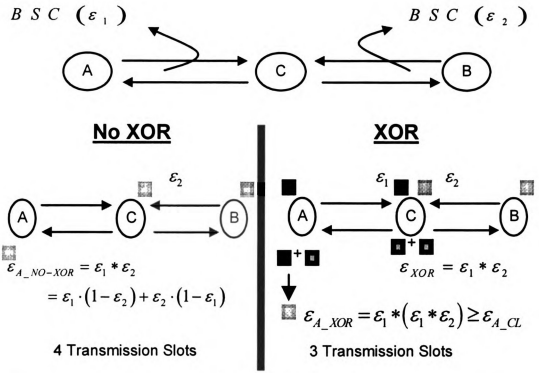
We illustrate the work presented in this paper in terms of the simple canonical two-hop network shown in Figure 24. As shown in Figure 24, the network consists of two sources

A and B with an intermediate router C connecting them. Transmitter A (Transmitter B) wants to send some information to Receiver B (Receiver A). In a conventional network, each packet from Transmitter A (Transmitter B) to Receiver B (Receiver A) would have to be transmitted once each over links 1 and 3 (links 2 and 4). The "bottleneck" in communication is the exchange of information that has to take place at bridge/router C. Network Coding can take advantage of the broadcast nature of wireless communication to save one transmission. Suppose that sources A and B transmit packets X and Y respectively, then the intermediate router C performs network coding by simply XOR-ing the packets and broadcasting $Z = X \oplus Y$ over the links 3 and 4 simultaneously. Receiver A (B) can easily obtain a copy of packet Y (X) on the basis of a simple XOR operation $Y = X \oplus Z$ ($X = Y \oplus Z$).

5.1.2. Impact of Bit Error Rate on Optimal Processing

The above discussion clearly exhbits that when the transmissions over the channels are error free, NC leads to a throughput improvement. More specifically it can be easily verified that coding at node C leads to a throughput improvement by a factor (4/3) over traditional forwarding. Even when the channels are noisy, if complete channel decoding and re-encoding is employed at the intermediate node C the throughput gain due to NC is self-evident and identical to the error-free case i.e. the throughput improves by a factor of (4/3). Hence as stated previously, in this work we concentrate on the scenario in which processing at intermediate nodes is limited to only XOR-ing of received packets.

To motivate the discussion on noisy channels without processing, for now, let the channels be described by Binary Symmetric Channels (BSCs) as shown in Figure 25. As



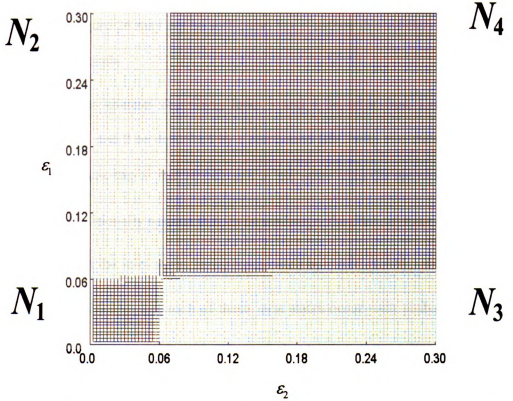
FigureB.25 Illustrated on top is a two hop topology used by nodes A and B to exchange mutually independent information. The illustrations at the bottom highlight the difference in evolution of the BERs in the packets, depending upon the coding/forwarding function employed at the relay node C.

illustrated in Figure 25, in the absence of complete processing, XOR-ing of corrupted packets can actually lead to error amplification. If the error-rates are sufficiently high, the capacity loss due to error amplification can offset the throughput gain on account of employing NC. Additionally it should be noted that the error amplification is not identical for both A and B; implying that network coding functions may not be equally fair to the intended receivers. To illustrate this phenomenon, we obtain Figure 26 by comparing the throughput capacity provided to each receiver. The expressions used to evaluate the throughput capacity are given by:

$$C_{A\_No-XOR} = C_{B\_No-XOR} = (1/4)\left(1 - h_b(\varepsilon_1 * \varepsilon_2)\right)$$

$$C_{A\_XOR} = (1/3)\left(1 - h_b(\varepsilon_1 * \varepsilon_1 * \varepsilon_2)\right) \tag{27}$$

$$C_{B\_XOR} = (1/3)\left(1 - h_b(\varepsilon_1 * \varepsilon_2 * \varepsilon_2)\right)$$



FigureB.26 To the left we partition the joint parameter space defined by the BSC parameters $(\varepsilon_1, \varepsilon_2)$ into 4 regions depending upon the benefit of XOR-ing to each of the eventual receivers. (N_1) XOR-ing improves throughput capacity to both the receivers (N_2) XOR-ing beneficial only to B (N_3) XOR-ing beneficial only to A (N_4) Forwarding a better strategy for both A and B.

In Figure 26, it can be observed that coding is not always the best function. Furthermore, it should be observed that under certain channel conditions neither of the functions is mutually beneficial.

5.1.3. Adaptive Processing

The observations clearly motivate the need to alter the coding functionality in accordance to the corruptions in the packets.

For static channels, depending on the BSC parameters, for regions 1 and 4 a dominant optimal policy could be adopted. However under time-varying channel conditions a direct application of Figure 26 may not be sufficient. We illustrate this subtlety with the help of the following example.

Example: Let the channel describing links 3 and 4 be error free. Let the channels describing links 1 and 2 be described by a compound BSC with two states. In one state the channel behaves like a $BSC(0.04)$ and in the other state the channel behaves like a $BSC(0.3)$. Additionally assume that, in an exchange, the channel state of link 1 and 2 are identical, with the joint states being represented by $\{BSC(0.04), BSC(0.04)\}$ and $\{BSC(0.3), BSC(0.3)\}$. Let each of these joint states occur with probability 0.5. We do not assume any transmitter side Channel State Information (CSI). However we assume that each node in the network including node C has receiver-side CSI. Hence node C can alter the NC function in accordance to the corruption levels in the received packets. For the above-described channel, the ANC policy obtained by a direct application of Figure 26 dictates that 50% of the times the optimal NC function is FORWARDING, while for the remainder the optimal function is CODING. Each time the received packets are corrupted by a $BSC(0.3)$, node C relays the packets without XOR-ing. As against this the packets are XOR-ed and broadcast when they are

corrupted by a $BSC(0.04)$. The average throughput capacity under such a scheme is given by $\frac{\textit{Average}\text{ Information Rate recd. per exchange}}{\textit{Average}\text{ No. of tranmissions per exchange}} = \frac{\tau}{(1/R)}$

$$= \frac{0.5\cdot(1-h_b(0.3))+0.5\cdot(1-h_b(0.04*0.04))}{0.5\cdot 3+0.5\cdot 4} = 0.1040$$

It can be easily verified that the best deterministic scheme is not represented by the above describe ANC. An ANC scheme that employs a coding function for the state $\{BSC(0.3), BSC(0.3)\}$ and a forwarding function for the state $\{BSC(0.04), BSC(0.04)\}$ performs better and provides a throughput capacity of 0.1109. The throughput capacity provided by only coding is 0.1046, while that provided by only forwarding is 0.1096.

From the above discussion, it is quite clear that ANC if designed appropriately can lead to performance benefits. However it is equally evident that a direct application of optimal rules at specific channel conditions may lead to the sub-optimal ANC performance. In this chapter we setup an optimization problem that formally allows us to design an efficient ANC scheme.

5.2. Preliminaries

5.2.1. Time Varying Channel

In this chapter we assume that the time varying channel model can be described by a quasi-static Compound BSC (CBSC). This channel model has been discussed in the previous chapter. Thus as defined in Chappter 4, we assume that channel behavior

remains static over the duration of transmission of single slot or packet, i.e. all the bits in a single packet are corrupted by an identical BSC. However the BSC parameter may vary across packets to form the compound BSC. Also remember that the CBSC can be described by a mass function $\hat{f}(\cdot)$ defined on some finite discrete subset S of the interval $[0,0.5]$, such that, $\hat{f}(p)$ represents the probability of the channel being described by a BSC with parameter p. For mathematical convenience we shall often utilize the continuous equivalent of $\hat{f}(\cdot)$ given by

$$f(p) = \begin{matrix} 0 & \forall p \notin S \\ \hat{f}(\mathrm{p})\delta(p) & \forall p \in S \end{matrix} \tag{28}$$

Cascade of Time Varying Channels:

Cascade of BSCs is a BSC, hence the cascade of CBSC is also a CBSC. Given two CBSC channels f_1 and f_2 the equivalent CBSC channel f_{eq} is given by

$$f_{eq}(p) = \sum_{\substack{p_1, p_2 \text{ such that} \\ p = p_1 * p_2}} \left(f_1(p_1) \cdot f_2(p_2)\right) \tag{29}$$

The equivalent channel of a cascade of more than two channels is obtained by a repeated operation of the above equation. From here on we represent f_{eq} by $f_1 \otimes f_2$, where $\otimes$ represents the operation described by (29). Additionally, since cascades of BSC are commutative, the cascades of CBSC are also commutative. Hence the equivalent channel obtained by cascade of more than two channels can just be represented as $f_1 \otimes f_2 \otimes f_3 \cdots$. Additionally we shall often use a shortened notation $f_1 \otimes p_2$ to represent the equivalent channel that is obtained by cascading f_1 with a $BSC(p_2)$.

5.2.2. Network Model and Exchange Rules

We assume a two-hop network topology as done in Figure 25. However instead of assuming the channels to be BSC, we assume the channels to CBSC, such that, the channels from A to C, C to A, B to C, C to B are described the mass functions $\hat{f}_{AC}$, $\hat{f}_{CA}$, $\hat{f}_{BC}$, $\hat{f}_{CB}$ respectively. The domains, over which the functions $\hat{f}_{AC}$, $\hat{f}_{CA}$, $\hat{f}_{BC}$, $\hat{f}_{CB}$ are defined, are given by S_{AC}, S_{CA}, S_{BC}, S_{CB} respectively. As done previously, we assume that the communication in the network takes place in terms of "*exchanges*". Each exchange consists of packet transmission from A to C and B to C followed by packet transmissions from C. The number of packet transmissions from C are dependent upon the "*exchange rules*" being employed at C and the corruption levels in the packets received by C. Thus we assume that receiver-side CSI is available at all the nodes.

Let Λ be set of coding functions that can be employed at a relay node. An exchange rule describes the probability with which a coding function is employed, given the corruption levels in the packets received at the relay node. For the model considered here the coding functions operate only on two packets. Thus for each coding function we define a "*rule*" as a function

$$\lambda_{c_i} : [0,0.5] \times [0,0.5] \rightarrow [0,1] \tag{30}$$

such that $\lambda_{c_i}(p_1, p_2)$ represents the probability of coding function c_i being employed when a packet received from A is corrupted by $BSC(p_1)$, while that received from B is corrupted by $BSC(p_2)$. The rules so defined thus satisfy the following restriction

$$\sum_{c_i \in \Lambda} \lambda_{c_i}(p_1, p_2) = 1 \quad \forall p_1, p_2 \in [0, 0.5] \tag{31}$$

In this work we restrict our attention to the functions $\{c_1 \equiv CODING \text{ or } XOR-ing,\ c_2 \equiv FORWARDING\}$. Since we employ only two possible functions, from here on $\lambda_{c_1}(p_1, p_2)$ is simply represented by $\lambda(p_1, p_2)$, while $\lambda_{c_2}(p_1, p_2)$ is represented by $\bar{\lambda}(p_1, p_2) = 1 - \lambda(p_1, p_2)$. Two important special cases of exchanges rules are given by

$$\lambda(p_1, p_2) = 1 \quad \forall p_1, p_2 \in [0, 0.5] \tag{32}$$

$$\lambda(p_1, p_2) = 0 \quad \forall p_1, p_2 \in [0, 0.5] \tag{33}$$

where (32) represents a scheme that always employs CODING, while (33) represents a scheme that always employs FORWARDING.

5.2.3. Performance Measurement

The throughput capacity provided to each receiver is determined by the corruption in the received packets and also by the rate at which the packets are received. The distribution of BERs in the packets received by a node can be mapped to an equivalent *reception* CBSC, f_{eq}. For such a channel the Ergodic capacity [42] is given by

$$\tau = \int_0^{0.5} f_{eq}(p)\left(1 - h_b(p)\right) dp \tag{34}$$

The equivalent channel will be dependent on the employed exchange rule λ. We shall describe how we can obtain the equivalent channel shortly, but for now let's assume that we have some mechanism to do so. As explained above the throughput capacity will

also be determined by the rate at which packets are received. We can measure the rate by evaluating the average number of slots used in each exchange and then taking the reciprocal. Thus the rate, R is given by

$$R = \frac{1}{\left(\int_0^{0.5} \int_0^{0.5} \begin{pmatrix} 3 \cdot f_{AC}(p_1) f_{BC}(p_2) \lambda(p_1, p_2) + \\ 4 \cdot f_{AC}(p_1) f_{BC}(p_2) \bar{\lambda}(p_1, p_2) \end{pmatrix} dp_1 dp_2 \right)} \tag{35}$$

Hence the average throughput capacity is given by:

$$C = \tau \cdot R \tag{36}$$

The eventual performance measure we are interested in will be determined by a linear combination of the throughput capacity provided to each receiver.

Determining the equivalent channel:

The equivalent channels can be easily determined when the exchange rule is always:

(i) FORWARDING: $f_{A\_eq} = f_{BC} \otimes f_{CA}$, $f_{B\_eq} = f_{AC} \otimes f_{CB}$ or

(ii) CODING: $f_{A\_eq} = f_{BC} \otimes f_{CA} \otimes f_{AC}$, $f_{B\_eq} = f_{AC} \otimes f_{CB} \otimes f_{BC}$

The rates for the above rules are determined by the function itself (1/4 for CODING, 1/3 for FORWARDING) and hence the throughput capacity can be easily determined.

In the following we illustrate the determination of the equivalent reception CBSC for an arbitrary exchange rule. To do so, first consider the scenario when f_{AC} and f_{BC} are represented by $BSC(p_{AC})$ and $BSC(p_{BC})$. For such a channel conditions the equivalent reception channels are given by:

$$f_{A\_eq}(p_1,p_2)=\lambda(p_1,p_2)(f_{CA}\otimes(p_{AC}*p_{BC}))+\bar{\lambda}(p_1,p_2)(f_{CA}\otimes p_{BC})$$
$$f_{B\_eq}(p_1,p_2)=\lambda(p_1,p_2)(f_{CB}\otimes(p_{AC}*p_{BC}))+\bar{\lambda}(p_1,p_2)(f_{CB}\otimes p_{AC}) \quad (37)$$

The above scenario can now be generalized to the case where f_{AC} and f_{BC} are arbitrary CBSC as

$$f_{A\_eq}=\sum_{p_1\in S_{AC},p_2\in S_{AC}}\left(\hat{f}_{AC}(p_1)\cdot\hat{f}_{BC}(p_2)\left(f_{A\_eq}(p_1,p_2)\right)\right)$$
$$f_{B\_eq}=\sum_{p_1\in S_{AC},p_2\in S_{AC}}\left(\hat{f}_{AC}(p_1)\cdot\hat{f}_{BC}(p_2)\left(f_{B\_eq}(p_1,p_2)\right)\right) \quad (38)$$

5.3. Problem Formulation

The design of an efficient ANC scheme requires the design of an optimal exchange rule $\lambda(p_1,p_2)$. The optimal exchange rule in general is a non-parametric function described on the two-dimension space $[0,0.5]\times[0,0.5]$. Thus to make the problem tractable we shall quantize the parameter space $[0,0.5]\times[0,0.5]$ into m suitable rectangular intervals. Such interval can be represented as

$$I_i\equiv[p_{1i}-\Delta_{1i},p_{1i}+\Delta_{1i}]\times[p_{2i}-\Delta_{1i},p_{2i}+\Delta_{2i}]\ \forall i\in[1,m]$$

For such a quantization we define the mass function

$$\tilde{f}_i=\int_{I_i}f_{AC}(p_1)f_{BC}(p_1)dp_1dp_2$$

which represents the probability of receiving a pair of packets corrupted by BSCs with parameter values in the interval I_i. Additionally we restrict our attention to exchange rules that remain constant over each interval. Thus an exchange rule can now be

represented by identifying a λ_i for each Interval. The equivalent reception channels can thus be approximated as

$$\tilde{f}_{A\_eq} = \sum_{i=1}^{m} \lambda_i \left(\tilde{f}_i \cdot \left(f_{CA} \otimes (p_{1i} * p_{2i}) \right) \right) + \sum_{i=1}^{m} \overline{\lambda_i} \left(\tilde{f}_i \cdot (f_{CA} \otimes p_{2i}) \right)$$

$$\tilde{f}_{B\_eq} = \sum_{i=1}^{m} \lambda_i \left(\tilde{f}_i \cdot \left(f_{CB} \otimes (p_{1i} * p_{2i}) \right) \right) + \sum_{i=1}^{m} \overline{\lambda_i} \left(\tilde{f}_i \cdot (f_{CB} \otimes p_{1i}) \right) \tag{39}$$

Let,

$$a_i = \tilde{f}_i \cdot \left(E_{f_{CA} \otimes (p_{1i} * p_{2i})} \left[1 - h_b(p) \right] \right), \ \overline{a_i} = \tilde{f}_i \cdot \left(E_{(f_{CA} \otimes p_{2i})} \left[1 - h_b(p) \right] \right)$$

$$b_i = \tilde{f}_i \cdot \left(E_{f_{CB} \otimes (p_{1i} * p_{2i})} \left[1 - h_b(p) \right] \right), \ \overline{b_i} = \tilde{f}_i \cdot \left(E_{(f_{CB} \otimes p_{2i})} \left[1 - h_b(p) \right] \right) \tag{40}$$

Thus the Ergodic Capacity for the equivalent reception channels is thus suitably approximated by

$$\tau_A = \sum_{i=1}^{m} a_i \lambda_i + \sum_{i=1}^{m} \overline{a_i} \overline{\lambda_i}, \ \tau_B = \sum_{i=1}^{m} b_i \lambda_i + \sum_{i=1}^{m} \overline{b_i} \overline{\lambda_i} \tag{41}$$

Now, also observe that the rate given in equation 10 can also be approximated as

$$R = 1 \Big/ \left(\sum_{i=1}^{m} \lambda_i \left(\tilde{f}_i \cdot 3 \right) + \sum_{i=1}^{m} \overline{\lambda_i} \left(\tilde{f}_i \cdot 4 \right) \right) \tag{42}$$

Thus from equations (41) and (42) we can state the problem of designing optimal exchange rules as the following optimization problem

Linear-Fractional Program (LFP):

$$\text{maximize } F = \frac{\sum_{i=1}^{m} \alpha_i \lambda_i + \sum_{i=1}^{m} \overline{\alpha_i} \overline{\lambda_i}}{\sum_{i=1}^{m} \beta_i \lambda_i + \sum_{i=1}^{m} \beta_i \overline{\lambda_i}} \text{ (in variables } \lambda_i, \overline{\lambda_i}) \tag{43}$$

subject to $\lambda_i + \bar{\lambda}_i = 1; \quad 0 \le \lambda_i, \bar{\lambda}_i \le 1 \quad for \quad i := 1,\ldots,m$ where

$\alpha_i = (\mathrm{w})a_i + (1\text{-w})b_i$, $\bar{\alpha}_i = (\mathrm{w})\bar{a}_i + (1\text{-w})\bar{b}_i$, $\beta_i = \tilde{f}_i \cdot 3$, $\bar{\beta}_i = \tilde{f}_i \cdot 4$ are constants.

We solve the above LFP, by utilizing the following optimization problems

Linear-Program (LP):

$$F(R) = \max_{\lambda_i, \bar{\lambda}_i} \sum_{i=1}^{m} (R\alpha_i)\lambda_i + \sum_{i=1}^{m} (R\bar{\alpha}_i)\bar{\lambda}_i \tag{44}$$

$$\textit{subject to} \quad \lambda_i + \bar{\lambda}_i = 1;\ 0 \le \lambda_i, \bar{\lambda}_i \le 1 \ \textit{for}\ i := 1,\ldots,m\,, \quad \sum_{i=1}^{m} \beta_i\lambda_i + \sum_{i=1}^{m} \beta_i\bar{\lambda}_i = R$$

Single-Variable Convex-Problem:

$$F = \max_{R} F(R) \ \textit{subject to}\ 0.25 \le R \le (1/3) \tag{45}$$

We obtain the solution for problem (43), by solving (45) as a single variable optimization problem on a constrained domain. In our approach, the gradient direction at the boundaries is evaluated to first verify that the optimum is not at a boundary. Following this we adopt the following iterative algorithm:

Algorithm 1:

Initialize: $R_{left} = 0.25,\ R_{right} = 0.33$

Iterative Update: $R_{new} = \left(F(R_{left})R_{left} + F(R_{right})R_{right}\right)$

$If\ R_{left} > R_{right}\ R_{right} := R_{new}\ ELSE\ R_{left} := R_{new}$

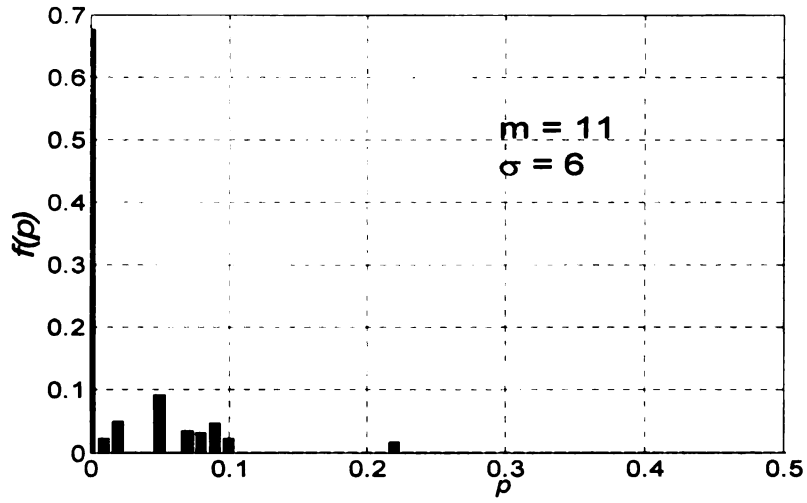
Stopping Criteria: *Stop if* $\left|R_{left} - R_{right}\right| < \Delta_{threshold}$

Note that the enumeration of the above algorithm requires us to solve an instance of problem (44) in each iteration. Also note that the quantized exchange rule obtained in

accordance to the above approach can be applied over the entire parameter space with the help of suitable interpolation.

5.4. Numerical Results

5.4.1. Measurement Based Channel Models



FigureB.27 A CBSC obtained by utilizing the empirical relationships $\varepsilon(SSR)$ from Figure 14 and $\delta(SSR)$ from Figure 15, along with $g(SSR)$ described by mean m=11 and standard deviation $\sigma = 6$.

The time-varying channel models we use to illustrate the utility of the proposed approach are derived from actual 802.11b and 802.15.4 wireless measurements done in Chapter 4 and [56] respectively. As discussed in Chapter 3, in these measurements, it is observed that practical radio devices are capable of associating a Signal-to-Silence Ratio (SSR) with each received packet. Empirical measurements thus allow the evaluation of the following relationships:

- $\delta(SSR)$ which represents the probability of receiving a corrupted packet with a particular SSR indication.

- $\varepsilon(SSR)$ which represents the average BER in corrupted packets with a particular SSR indication.

We derive our channel models by equating Signal to Noise Ratio (SNR) with SSR[1], assuming some distribution $g(SNR)$ on the variation of SNR. The time-varying channel is given by $\hat{f}(0) = \sum_{SSR} g(SSR)(1-\delta(SSR))$, $\hat{f}(\varepsilon(SSR)) = \delta(SSR)g(SSR)$.

In practice it is observed that the SSR/SNR variations can often be approximated by a Gaussian distribution. Thus in this work we concentrate on channel models obtained by assuming a Gaussian distribution for $g(SNR)$. Figure 27 provides an example.

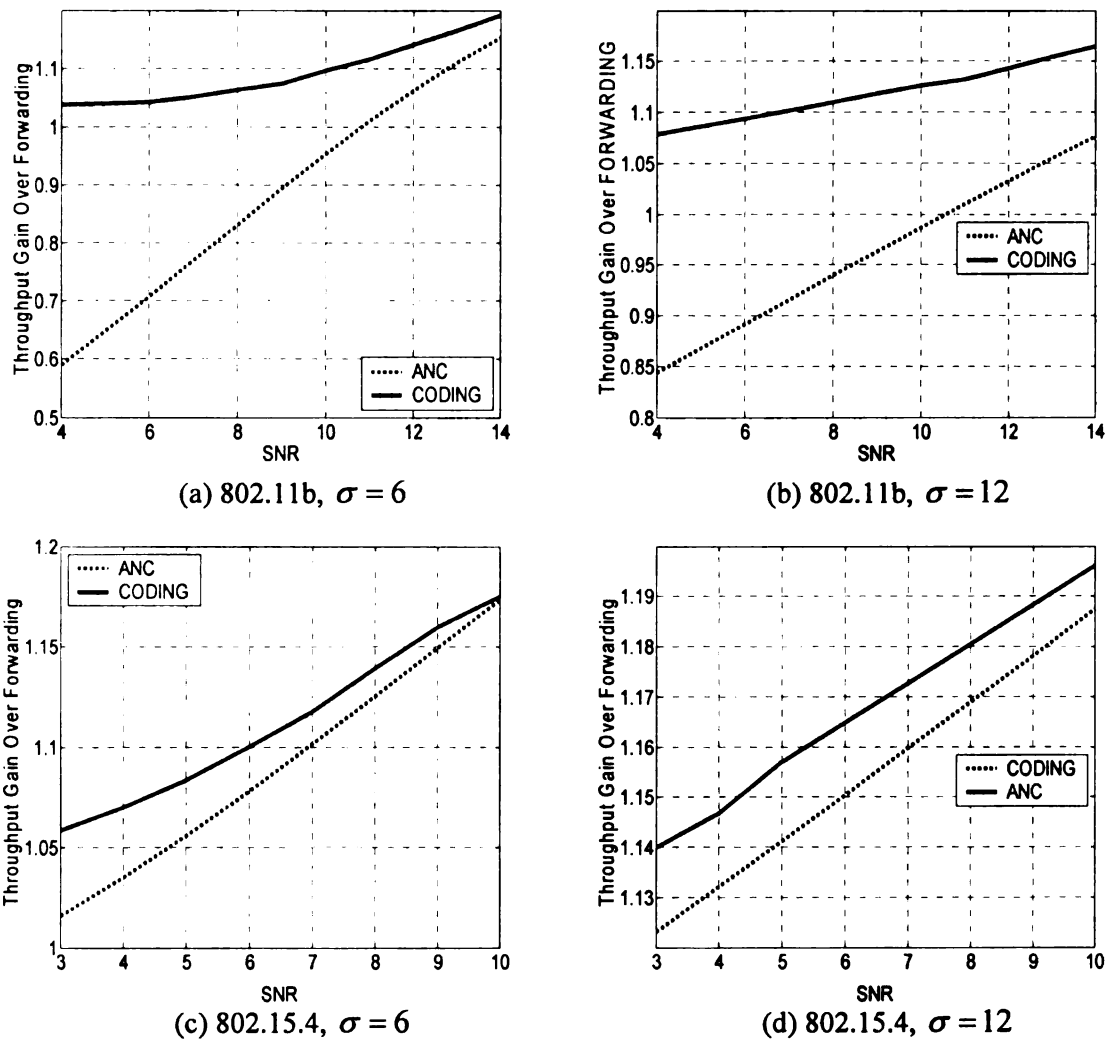
5.4.2. Performance Comparisons

We compare the performance of ANC with only FORWARDING and only CODING in terms of the ratio of the throughput capacities[2] provided by each of the schemes. Thus Figure 28 plots the ratio of the throughput capacity under ANC and CODING with that under FORWARDING. As should have been expected, the performance of CODING can be worse than FORWARDING in the low SNR region of the 802.11b plots. However, in all the plots in Figure 28 it can be observed that the performance of ANC is always better than both the non-adaptive schemes. On 802.15.4 channels ANC provides a modest improvement of 1-2% over CODING and 5-20% gains over FORWARDING. On 802.11b, ANC provide a significant improvement over both non-adaptive schemes. A minimum gain of 5% is consistently seen on the entire range of studied channel

[1] In this work it is assumed that SSR and SNR are expressed in dB.

[2] We focus on w = 0.5 and hence comparison in terms of throughput capacity are sufficient to describe the performance of both the receivers.

conditions. On 802.15.4 networks, ANC provides a gain of 5-15% over FORWARDING and 5-60% on CODING. Additionally, it can be observed that the performance gains of ANC increase with variation in the channel conditions. For SNR values higher than a certain threshold, the performance of ANC and CODING is almost identical. Thus the performance benefits of ANC are expected to be more pronounced in challenged wireless networks



(a) 802.11b, $\sigma = 6$

(b) 802.11b, $\sigma = 12$

(c) 802.15.4, $\sigma = 6$

(d) 802.15.4, $\sigma = 12$

FigureB.28 Illustrated above is the throughput gain provided by ANC and only CODING over Forwarding. Each SNR value represents a CBSC obtained from $g(SSR)$ with m=SNR. To the left are the plots with lower channel variance and to the right are the plots with higher channel variance.

6. PART-B REFERENCES

[1] S. Shakkottai, T. S. Rappaport and P. C. Karlsson ``Cross-layer Design for Wireless Networks", IEEE Communications magazine, October, 2003.

[2] M. van der Schaar, S. Krishnamachari S. Choi, X. Xu, "Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs", IEEE Journal on Selected Areas of Communication, vol. 21, Dec. 2003.

[3] I. Kozinetsev and J. McVeigh, "Improving last-hop multicast streaming video over 802.11", BroadWiM 2004.

[4] Y. Wu, P. A. Chou, Q. Zhang, K. Jain, W. Zhu, and S.-Y. Kung, "Network planning in wireless ad hoc networks: a cross-layer approach," IEEE Journal on Selected Areas in Communications, special issue on wireless ad hoc networks, vol. 23, no. 1, pp. 136-150, Jan. 2005.

[5] M. van der Schaar and S. Shankar, "Cross-Layer Wireless Multimedia Transmission: Challenges, principles, and new paradigms," IEEE Wireless Communications Magazine, vol. 12, no. 4, pp. 50-58, August 2005.

[6] I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk and H. Sips, "Optimized Video Streaming over 802.11 by Cross-layer Signaling," IEEE Communication Magazine, Special Issue on Cross-Layer Design, Jan. 2006.

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

[8] L. Larzon, M. Degermark, and S. Pink, "Efficient Use of Wireless Bandwidth for Multimedia Applications," IEEE MoMUC, November 1999.

[9] L. Larzon, M. Degermark, S. Pink, "The UDP lite protocol," IETF Internet Draft, February 2001.

[10] A. Singh, A. Konrad, A. D. Joseph, "Performance Evaluation of UDP Lite for Cellular Video," NOSSDAV, 2001.

[11] H. Zheng and J. Boyce, "An Improved UDP Protocol for Video Transmission Over Internet-to-Wireless Networks," IEEE Trans. on Multimedia, vol. 3, no. 3, pp. 356–365, September 2001.

[12] G. Ding, H. Ghafoor and B. Bhargava, "Resilient Video Transmission over Wireless Networks", 6th IEEE International Conf. on Object-oriented Real-time Distributed Computing, Japan, May 2003.

[13] H. Zheng, "Optimizing Wireless Multimedia Transmissions through Cross Layer Design," IEEE ICME, July 2003.

[14] S. A. Khayam, S. S. Karande, H. Radha and D. Loguinov, "Performance Analysis and Modeling of Errors and Losses over 802.11b LANS for High-Bitrate Real-Time Multimedia," Signal Processing: Image Communication, vol. 18, Aug 2003.

[15] A. Servetti, J.C. De Martin, "Link-Level Unequal Error Detection for Speech Transmission over 802.11b Networks," Proc. Special Workshop in Maui (SWIM), Maui, Hawaii, January 2004.

[16] E. Masala, M. Bottero, J.C. De Martin, "Link-Level Partial Checksum for Real-Time Video Transmission over 802.11 Wireless Networks", Proceedings of 14th International Packet Video Workshop (PVW), Irvine, CA, December 2004.

[17] E. Masala, M. Bottero, J.C. De Martin, "MAC-Level Partial Checksum for H.264 Video Transmission over 802.11 Ad Hoc Wireless Networks" Proceedings of IEEE 61st Semiannual Vehicular Technology Conference (VTC), May 2005.

[18] E. Masala, A. Servetti, J.C. De Martin, "Standard Compatible Error Correction for Multimedia Transmissions over 802.11 WLAN", Proc. of IEEE International Conference on Multimedia & Expo (ICME), July 2005.

[19] W. Jiang, "A Bit Error Correction without Redundant Data: a Mac layer technique for 802.11 Networks", WinMee 2006.

[20] R. Riemann and K. Winstein, "Improving 802.11 Range with Forward Error Correction", MIT-CSAIL-TR-2005-011

[21] F. Pauchet, C. Guillemot, M. Kerdranvat, S. Pateux, P. Siohan, "Conditions for SVC bit error resilience testing" Joint Video Team (JVT) 17th Meeting: Nice, FR, 14-21 October, 2005

[22] S. A. Khayam, S. Karande, M. U. Ilyas and H. Radha, "Header Detection to Improve Multimedia Quality over Wireless Networks," accepted for publication in IEEE Transactions on Multimedia.

[23] D. B. Faria, D. R. Cheriton, "No Long-term Secrets: Location-based Security in Overprovisioned Wireless LAN, ACM Workshop HoT-NeTs, Nov. 2004.

[24] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan, "Measurement-based Characterization of 802.11 in a Hotspot SettingMeasurement-based Characterization of 802.11 in a Hotspot SettingMeasurement-based Characterization of 802.11 in a Hotspot SettingMeasurement-based Characterization of 802.11 in a Hotspot Setting," SIGCOMM 2005 Workshop, E-WIND, Aug. 2005.

[25] Utpal Parrikar, "On SNR Aware Analysis and Modeling of 802.11b Link-level Residue Errors", Master's Thesis, Electrical and Computer Engineering Department, Michigan State University.

[26] ISL3873: Wireless LAN Integrated Medium Access Controller with Baseband Processor Intersil Corporation, 2000.Application Note FN4868.

[27] http://www.linux-wlan.orghttp://www.linux-wlan.org

[28] http://www.wi-fiplanet.com/tutorials/article.php/1116311http://www.wi-fiplanet.com/tutorials/article.php/1116311

[29] http://www.awe-communications.com/Network/WLAN/index.html

[30] ISO/IEC JTC 1/SC29/WG11 and ITU-T SG16 Q.6, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," Doc. JVT-G050, March 2003.

[31] S. A. Vanstone and P. C. van Oorschot, "An Introduction to Error Correcting Codes with Applications," Kluwer Academic Publishers.

[32] R. H. Morelos-Zaragoza, "The Art of Error Correcting Coding," John-Wiley & Sons Ltd.

[33] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", Computer Communication Review, April 1997.

[34] J. C. Bolot, S. Fosse-Parisis, D. Towsley, "Adaptive FEC-based error control for Internet telephony," Proceedings of IEEE INFOCOM '99, vol. 3, pp. 1453-1460, 1999.

[35] M. El-Khamy and R. J. McEliece, "Iterative algebraic soft decision list-decoding of Reed-Solomon codes," *to appear* in IEEE Journal on Selected Areas in Communications.

[36] IEEE Transactions on information theory, Special Issue on codes on graphs and iterative algorithms, vol. 47, Issue 2, Feb 2001.

[37] W.E. Ryan, "An introduction to LDPC codes," in CRC Handbook for Coding and Signal Processing for Recoding Systems (B. Vasic, ed.), CRC Press, 2004.

[38] Xiao-Yu Hu, Evangelos Eleftheriou, Dieter M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs", IEEE Transactions on Information Theory, vol. 51. no. 1, pp. 386-398, 2003.

[39] http://www.cs.toronto.edu/~radford/ldpc.software.html

[40] T. M. Cover and J. A Thomas, "Elements of Information Theory," Wiley Series in Telecommunications.

[41] M. Mushkin and I. Bar-David, "Capacity and Coding for the Gilbert-Elliot Channels," IEEE Transactions on Information Theory, vol. 35, Nov. 1989.

[42] A. Goldsmith and P. Varaiya, "Capacity, Mutual Information, and Coding for Finite-State Markov Channels," IEEE Transaction On Information Theory, vol 42., No. 3, pp. 868-886, 1996.

[43] S. C. Draper, B. Frey, and F. R. Kschischang, "Rateless Coding for Non-Ergodic Channels with Decoder Channel State Information", www.eecs.berkeley.edu/~sdraper/researchDir/pdf/ratelessTimeVary.pdf

[44] J. Michael Steele, "An Introduction To The Art Of Mathematical Inequalities: The Cauchy-Schwarz Master Class", Cambridge University Press, 2004.

[45] ISO/IEC JTC 1/SC29/WG11 and ITU-T SG16 Q.6, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," Doc. JVT-G050, March 2003.

[46] A. Majumdar, D. G. Sachs, I. V. Kozinetsev, K. Ramchandran, "Multicast and unicast realtime video streaming over wireless LANs", IEEE Transaction on Circuits Systems and Video Technology. Vol. 6, 2002.

[47] P. Gupta and P. R. Kumar, "The capacity of wireless networks", IEEE Transactions on Information Theory, IT-46(2):388-404, March 2000.

[48] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, "Network information flow," IEEE Trans. on Information Theory, vol. 46, pp. 1204-1216, 2000

[49] S.-Y. R. Li, R. W. Yeung, and N. Cai. "Linear network coding".

[50] R. Koetter, M. Medard, "An Algebraic Approach to Network Coding", Transactions on Networking, October 2003.

[51] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Médard, and N. Ratnakar, "Network coding for wireless applications: A brief tutorial," In Proc. International Workshop on Wireless Ad-hoc Networks (IWWAN) 2005, May 2005.

[52] Y. Wu, P. A. Chou, S.-Y. Kung, ``Information exchange in wireless networks with network coding and physical-layer broadcast," Microsoft Technical Report, MSR-TR-2004-78, Aug. 2004.

[53] D. Tuninetti, C. Fragouli, "Processing along the way: forwarding vs.coding," ISITA, Parma. 2004.

[54] Y. E. Sagduyu, A. Ephremides, "Joint Scheduling and Wireless Network Coding," NETCOD 2005.

[55] S. Boyd and L. Vandenberghe, "Convex Optimization," Cambridge University Press, 2004.

[56] M. U. Ilyas, "Collection Of Residual Error Traces In An IEEE 802.15.4 Low Rate Wireless Personal Area Network," TROOO1 -MSU-ECEWAVES-ILYASMUH