This is to certify that the
dissertation entitled

ANALYSIS AND DESIGN OF RELIABLE AND STABLE
LINK-LAYER PROTOCOLS
FOR WIRELESS COMMUNICATION

presented by

SOHRAAB SOLTANI

has been accepted towards fulfillment
of the requirements for the

DOCTOR OF PHILOSOPHY degree in Computer Science

Major Professor's Signature

March 25, 2009

Date

ANALYSIS AND DESIGN OF RELIABLE AND STABLE LINK-LAYER PROTOCOLS FOR WIRELESS COMMUNICATION

By

Sohraab Soltani

A DISSERTATION

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

DOCTOR OF PHILOSOPHY

Computer Science

2009

ABSTRACT

ANALYSIS AND DESIGN OF RELIABLE AND STABLE LINK-LAYER PROTOCOLS FOR WIRELESS COMMUNICATION

By

Sohraab Soltani

Wireless links are error-prone and susceptible to noise imposed by fading, interference and mobility. Therefore, current wireless networks suffer from high packet loss and poor connectivity among users. Wireless link-layer protocol embedded in Medium Access Control (MAC) has a significant role in providing robust information delivery over wireless channels. A primary focus of popular wireless link-layer protocols is to achieve some level of reliability using ARQ or Hybrid ARQ mechanisms. However, these and other leading link-layer protocols largely ignore the stability aspect of wireless communication, and rely on higher layers to provide stable traffic flow control. This thesis investigates the problem of reliable and stable transmission over wireless channels and highlights the inadequacies of the current IEEE802.11 standard link-layer which attempts to recover from losses using retransmissions.

In this thesis, we aim to tackle the critical issues associated with the inefficiencies of current wireless link-layer protocols and pursue a paradigm shift in the conventional 802.11 link-layer design. We develop a new link-layer framework to provide both the reliability and stability for point-to-point contention free wireless communication. Using this framework, we introduced four

link-layer protocols: 1) Packet Embedded Error Control (PEEC) protocol, a link-layer protocol designed to ensure reliable wireless communication by reducing the number of retransmissions which essentially leads to improving system throughput. PEEC is layer oblivious and uses the packet formats of current IEEE802.11 standard; 2) Delay Constraint PEEC (DC-PEEC), an extension of the PEEC protocol that targets the flow control of realtime video traffic (in addition to reliability) in wireless communication. DC-PEEC adjusts its parameters to provide low-latency communication to satisfy the delay constraint (required by the video application) while utilizing the channel bandwidth effectively; 3) Automatic Code Embedding (ACE) link-layer protocol, the first effort to develop a theoretical framework for analyzing and designing a wireless link-layer protocol that targets system stability in conjunction with reliable communication. The ACE protocol uses a unique and optimal code embedding rate to construct coded link-layer packets in every transmission to ensure stability, reliability and maximum throughput; 4) Prioritized ACE (PACE), the ACE based stable-and-reliable link-layer that employs a novel rate-adaptive Low Density Parity Check (LDPC) channel codes while interacting with the higher layers to provide a dynamic decoder scheduling service over varying wireless channel condition. PACE provides prioritized wireless link-layer communication that takes into consideration the level of importance/impact of each packet to improve the overall performance.

Our analysis and results of various experimental scenarios show that these protocols significantly outperform all competing link-layer protocols. Our findings in this thesis indeed provide a clear evidence of the feasibility of designing stable and reliable link layer over point-to-point (single-hop) 802.11 channels; and more importantly the potential of achieving significantly improved throughput by using this type of link-layer.

To my parents, Reza and Afsaneh.

To my sisters, Soroor and Sima.

ACKNOWLEDGMENTS

I would like to thank my family for their great support. In particular, I like to recognize the guidance my father provided through the course of my studies. I also thank my advisor Professor Hayder Radha for showing me a successful path in my research. Shirish, Kiran, Ali, Usman, and all of my colleagues at the WAVES laboratory; it was a privilege working with you. Finally, I am sincerely thankful to Sally Newton for her encouragement in difficult times.

Keep away from people who try to belittle your ambitions. Small people always do that, but the really great make you feel that you, too, can become great.

Mark Twain

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Despite the unprecedented success and proliferation of wireless LANs over the past decade, there are a few arguably major shortcomings in the underlying link-layer protocols of well-established wireless systems. These shortcomings are expected to be exacerbated as the level of heterogeneity and high-bandwidth requirements of emerging applications increase dramatically. In particular, popular wireless link-layer protocols, such as the retransmission (ARQ) based approach employed by the IEEE 802.11 standard suite, are designed to achieve some level of reliability by discarding corrupted packets at the receiver and by performing one or more retransmission attempts until a packet is received error-free or a maximum number of retransmission attempts is reached. Many leading research efforts [32]- [39] have highlighted the inefficiencies of the current 802.11 link-layer protocol and proposed a variety of remedy solutions. Although recent proposed remedies for the wireless link-layer focus on some aspects of the reliability issue, they largely ignore the stability dimension and they especially ignore the heterogeneous nature/demands of data/applications at higher layers. Meanwhile, we believe that emerging and future wireless networks supporting high-end heterogeneous applications cannot afford piecemeal solutions. Hence, in this thesis we aim to tackle the critical issues associated with the inefficiencies of current wireless link-layer protocols and pursue a paradigm shift in the conventional 802.11 link-layer design. In particular,

we introduce a comprehensive framework that presents a thorough analysis and modeling of a generic link-layer point-to-point wireless communication which employs channel codes to provide and maintain a reliable and stable communication. It is our belief that achieving the ultimate objective in the development of a reliable and stable link-layer for heterogeneous wireless networks that overcomes the shortcomings of current link-layer standards demands fundamental and radical changes to the conventional link-layer protocol design. We highlight the key issues with the 802.11 link-layer protocol:

1. *Inefficient reliability*: The 802.11 ARQ approach discards corrupted packets that are mostly error-free, even when there is only a single bit error in a corrupted packet. Hence, the effective throughput of 802.11 systems can be significantly improved. This issue led many efforts to propose new link-layer and cross-layer protocols that utilize corrupted packets (or *partial packets*) instead of discarding them [40]- [52]. In addition to Hybrid ARQ (HARQ) based methods [43–45], examples of recent efforts for combating inefficiencies of ARQ-based wireless protocols include Partial Packet Recovery (PPR) [54], packet combining [32]- [47], Cross-Layer Design with Side-information (CLDS) [72]- [75], ZipTx [42]. Some of these approaches, such as PPR and packet combining, exploit physical layer information regarding the quality of individual bits to improve the probability of recovering corrupted packets. Others, such as CLDS and ZipTx utilize information available in current 802.11 link-layer protocols in conjunction with error correcting codes to recover corrupted packets. In particular, the work on CLDS demonstrates a significant increase in throughput by utilizing corrupted packets under current 802.11 systems and shows that the mere utility of binary side information (packet is corrupted or not), which is available in the current 802.11 link layer protocol, can increase the effective information-theoretic capacity significantly [5].

2. *Largely absent stability*: The ARQ approach is designed to provide "reliability in the long

run", where information could eventually be delivered to the destination. Even then, the link-layer does not guarantee delivery and the reliability burden (due to wireless errors) is carried by higher layers, especially for applications that require guaranteed delivery. More importantly, ARQ-based 802.11 link-layer and other recent protocols largely ignore the stability aspect of data communications in terms of maintaining a sustainable flow, which is critical for a dynamic and heterogeneous wireless environment. Although many leading efforts have addressed the reliability and associated throughput inefficiency shortcoming of current 802.11 link-layer (as highlighted above), current ARQ and many emerging link-layer protocols rely on (or arguably shift the problem to) higher layers to provide reliable and stable flow control for both realtime and non-realtime traffic. In conjunction with the inefficient reliability approach, this design strategy has led to a great deal of inefficiency in throughput and to other major technical issues and challenges at higher layers. A well-known example is the TCP over-wireless performance degradation phenomenon, which led to major research efforts and numerous studies in attempt to mitigate the shortcoming of the lower layers.

1.1 Overview of Contributions

This thesis introduces a comprehensive framework for analysis and design of wireless link-layer protocols to overcome the above shortcomings. Our objective is to develop wireless link-layer protocols which make use of incremental channel codes to (1) provide maximum reliability (high likelihood of successful recovery data at the receiver) while ensures an efficient utilization of available bandwidth in transmission of new data, (2) offer system stability by ensuring that higher layers are neither starved for information packets nor is there a glut of packets leading to buffer overflows.

Using this framework, we first focus on the reliability aspect of wireless link-layer communication and identify its capabilities and limitations. These analyses lead to the development of a novel link-layer Packet Embedding Error Control (PEEC) protocol [3]. Chapter 4 introduces the PEEC protocol which uses a simple feedback mechanism to adaptively estimate channel conditions and administer the transmission of (data and parity) symbols within a packet. Chapter 5 extends the PEEC protocol and presents a novel error control protocol for wireless link-layer that targets both reliability and traffic flow control for realtime video communication. Further, Chapter 6 proposes a paradigm shift where both reliability and stability are targeted using an Automatic Code Embedding (ACE) wireless link-layer protocol. To the best of our knowledge this is the first effort to develop a theoretical framework for analyzing and designing a wireless link-layer protocol that targets system stability in conjunction with reliable communication. In Chapter 7, we build on the ACE framework [2] to achieve preferred data recovery order across connections, while maintaining stable and reliable data flows in wireless networks.

Chapter 4 presents statistical analysis of a link-layer, point-to-point and contention-free wireless communication where a receiver stores corrupted packets in its buffer for future recovery. We develop suitable models for each component of this communication scheme, which includes *message*, *channel*, *distortion* and *buffer* models. We use these models to introduce the PEEC link-layer protocol. PEEC employs packet-embedded parity symbols (instead of retransmission) for error recovery. Also, PEEC uses certain flags embedded in the receiver feedback to determine the amount of redundancy necessary for the next transmission. PEEC utilizes acknowledgment flags (a *decoding* and a *buffer*) to assess the channel and the receiver buffer conditions in every transmission. Depending on this assessment, PEEC adaptively administers the transmission of the data and redundancy (parity) symbols such that: (1) the level of parity symbols guarantees high likelihood of successful recovery of new data as well as corrupted data at the receiver buffer; (2) the available bandwidth is efficiently utilized for the transmission of new data. Our experimental

simulations show that PEEC outperforms the leading error control protocols designed for wireless link-layer. PEEC is layer oblivious and compatible to IEEE802.11 standard link-layer packet formats.

Chapter 5 develops an analytical framework for video communication which captures the behavior of realtime video traffic at the wireless link-layer while taking into consideration both reliability and latency conditions. Using this framework, we introduce a Delay Constraint Packet Embedded Error Control (DC-PEEC) protocol for wireless link-layer. DC-PEEC ensures reliable and rapid delivery of video packets by employing various channel codes to minimize fluctuations in throughput and provide timely arrival of video. The proposed effort begins by outlining a novel analytic framework to capture video traffic flow at the wireless link-layer. In particular, we develop a queuing model that captures realtime video traffic behavior under reliability and end-to-end delay constraint at the wireless link-layer. Specifically, we model the link-layer buffer as an M/G/1 queuing system with random-sized batch arrivals of video information having a single server representing the link-layer protocol with a general service-time distribution. Using this model, we find an operational code rate for DC-PEEC which guarantees video traffic flow with tolerable latency while utilizing the channel bandwidth effectively. We compare the performance of DC-PEEC with that of IEEE802.11 ARQ and HARQ in terms of PSNR gain under various realtime video communication setups. The simulation results suggest that DC-PEEC is a suitable wireless link-layer communication mechanism for realtime multimedia applications.

In Chapter 6, we propose a paradigm shift where both reliability and stability are ensured using an Automatic Code Embedding (ACE) wireless link-layer protocol. The proposed wireless ACE link-layer (a) employs a theoretically-sound framework and a corresponding strategy for embedding channel codes, using robust and well-defined code rates, in each packet; and (b) selects the code rates in an optimal and constrained manner to ensure reliability, stability, and maximum throughput. We believe that this work is the first to present a theoretical framework for analyzing

and designing a wireless link-layer protocol that targets system stability in conjunction with reliable communication. We begin by outlining a novel joint analytic framework to predict system behavior under ACE. Specifically, we first obtain an upper bound on operational code embedding rate that ensures reliability. Next, we develop a queuing model that captures system behavior under stability condition. In particular, we describe the link-layer behavior as an on-off source model using a two-state Continuous Time Markov chain (CTMC) model. We deploy fluid approximations to analytically characterize the buffer growth. By utilizing these models, we find a lower bound on operational code embedding rate which guarantees stable operation while utilizing the channel bandwidth effectively. An important conclusion of the above analysis is that various traffic demands (in terms of reliability and stability requirements) can be met using a packet-by-packet code embedding rate constraint that is independent of traffic type. This leads to simplistic, traffic-independent and elegant design rules for the ACE protocol, while providing reliability and stability in an optimal and joint manner. Our extensive analysis of ACE protocol over real channel traces collected on 802.11b WLANs for realtime and non-realtime traffic, TCP throughput and realtime video communication scenarios show that ACE significantly outperforms the conventional IEEE802.11 ARQ over varying wireless channels conditions.

Chapter 7 builds on the ACE framework [2] to achieve preferred data recovery order across connections, while maintaining stable and reliable data flows in wireless networks. Under the proposed Prioritized ACE (PACE) framework, our ACE based stable-and-reliable link-layer will employ a novel rate-adaptive Low Density Parity Check (LDPC) channel codes while interacting with the higher layers to provide a dynamic decoder scheduling service over varying wireless channel condition. Specifically, we develop a LDPC decoding model to capture the decoding process for link-layer traffic and use it to determine an optimal code selection strategy for maximal bandwidth utilization. Further, we find an optimal code embedding rate under the PACE framework to jointly meet the reliability, stability, and delay constraints of the wireless link-layer

communication. We classify heterogeneous link-layer traffic arrivals into different priority classes based on packet delay constraints and the distortion suffered. The traffic arriving in each priority class is modeled as a Poisson process. Consequently, we formulate the link-layer buffer as a multi-class $M/G/1$ priority queuing system where the decoding process (service process) of the PACE buffer is captured by nonhomogeneous geometric distribution [99]. Given the link-layer buffer model and the LDPC decoding model, we determine the optimal dynamic decoder scheduling under the PACE framework. This scheduling policy is a special case of a classic scheduling problem solved by Plambeck et al. in [100] and is asymptotically optimal. The PACE protocol incorporates the LDPC model and the dynamic scheduling policy into the original ACE protocol [2].

1.2 Organization of this Thesis

The rest of this thesis is organized as follows. Chapter 2 presents an overview of the related work. Chapter 3 provides background that is required to understand the material presented in this thesis. We introduce PEEC in Chapter 4. The DC-PEEC protocol developed for realtime wireless video communication is presented in Chapter 5. In Chapter 6, we present the ACE framework, the first analytical model that jointly targets reliability and stability of wireless link-layer communication. Chapter 7 analyzes the impact of prioritized wireless communication on the overall system performance and develops the PACE protocol. We conclude the thesis in Chapter 8.

CHAPTER 2

Related Work

Various link-layer protocols have been developed over the years to ensure reliability of wireless communication by using some sort of error detection and correction technique. In literature [1], error detection and error correction are defined as follows:

Definition 1. `Error Detection` *is the ability to detect the presence of errors caused by noise or other impairments during transmission from the transmitter to the receiver.*

Definition 2. `Error Correction` *is the information processing ability which result in the reconstruction of the original, error-free data at the receiver.*

There are two different ways to design an error correction protocol:

1. **Automatic repeat-request** (ARQ): The transmitter sends the data and also an error detection code, which the receiver uses to check for errors, and requests retransmission of erroneous data. In many cases, the request is implicit; the receiver sends an acknowledgement (ACK) of correctly received data, and the transmitter re-sends anything not acknowledged within a reasonable period of time.

2. **Forward error correction** (FEC): The transmitter encodes the data with an error-correcting code (ECC) and sends the coded message. The receiver never sends any messages back to

the transmitter. The receiver decodes what it receives into the "most likely" data. The codes are designed so that it would take an "unreasonable" amount of noise to trick the receiver into misinterpreting the data.

Popular link-layer protocols use either one of the above approaches or the combination of the two to provide reliability.

2.1 ARQ-based Schemes.

In this section, we present an overview of link-layer protocols that utilize automatic repeat request (ARQ) to recover erroneous packets.

2.1.1 The IEEE802.11 Standard

The IEEE 802.11 standard covers two layers of the OSI reference model: the *medium access control* (MAC) and the *physical* (PHY) layer. The fundamental function that provides fair access to the channel and best effort service is the distributed coordination function (DCF) that is based on a carrier sense multiple access with collision avoidance (CSMA/CA) algorithm. To deal with the collision problem, and other severe sources of errors such as interference, fading, and attenuation, the IEEE802.11 protocol incorporates positive acknowledgments: it incorporates frame check sequence (FCS) to detect errors and automatic repeat request (ARQ) to retransmit corrupted packets. If no ACK is returned (or FCS fails), the frame is scheduled for retransmission, until a maximum retransmission limit is reached.

The IEEE802.11 ARQ protocol discards corrupted packets without regard to the number and location of the errors. This approach is suitable for wireless channels with relatively low bit error rates (BER) because the likelihood of receiving consecutive corrupted packets is small and the original packet could be delivered after few retransmissions. However, for channels with

more severe error conditions (and arguably more realistic), IEEE802.11 ARQ causes multiple retransmissions (even if there is a single error in a packet) which in turn leads to the transmission of a large number of redundant (correct) data. As a result, the overall throughput deteriorates steadily and rapidly with increasing average channel BER.

2.1.2 Enhanced ARQ Strategies

In the current IEEE 802.11 MAC standard no attempt is made to correct erroneous packets: error detection provided by the FCS requires a retransmission even for a single erroneous bit. A relatively simple and standard-compatible way to improve the reliability of WLAN communications is to retain received erroneous frames which are normally discarded by the standard ARQ. Memory ARQ schemes combine several of such corrupted packets at the receiver to attempt to reconstruct the original error free packet. The average number of combined copies varies according to the channel condition, thus the effective degree of protection is dynamic. Each information packet, in fact, contains a parity check sequence for error detection so that the receiver can determine when to stop the packet combining algorithm because the original packet has been fully recovered.

Unlike conventional IEEE802.11 receivers, in the described methods the receiver stores an erroneous received packet before requesting a retransmission. For the purpose of error control, every different data MAC Protocol Data Unit (MPDU) can be identified by the 16-bit sequence control field that indicates its sequence number. Using the 32-bit FCS field at the receiver, the received packet can be checked for errors. If it is error-free, a positive acknowledgment is sent to the transmitter, inhibiting further retransmissions and the packet can be forwarded to the next hop or to the application level. If it is not, the packet is dropped, but stored in the receiver buffer waiting for a retransmission. All the received packets are then processed by the error correction algorithm to be described. If the procedure is not able to recover a correct packet further retransmissions are necessary. Transmissions are repeated till a correct frame is received, the data

in the cumulative buffer of received packets is correctable, or the maximum retransmission limit is reached.

XOR Combining

A first combination scheme, here referred to as xor combining [33, 34], consists in xor-ing two erroneous copies to locate errors in both packets. The decision process then involves a brute-force bit-by-bit inversion of the located bit error positions and checking for correctness using the FCS. When two copies are erroneous this operation fails if there is at least one bit position in which both copies have an error, or alternatively, if the total number of erroneous locations exceeds a given N_{max}. To make the algorithm implementable, in fact, an upper limit of computational complexity is defined which in practice is limited to values of N_{max} to 10, 11, or 12. Given a buffer size greater than two packets, more than one combination of packets is available for xor-ing. If no error recovery is possible, however, a retransmission is sought. This algorithm suffers from the performance limitations due to its high complexity.

Majority Combining

A second combination scheme, hereafter majority combining (MA-k) [35, 36], is proposed to overcome the performance limitation of working on packet pairs only and uses the last three received erroneous copies of a packet. If xor combining fails, then a bit-by-bit majority decision can be performed to construct a new packet where a bit is one if it is one at least in two of the combined copies. The error correction algorithm succeeds if no bit-error overlaps occur. This idea can then be extended to cover combinations of more than three copies where a majority decision over the last k packets is used as an estimate of the transmitted bits.

Generalized Majority Combining

The concept of combining packets to obtain a more reliable estimate of the transmitted bits can offer a significant advantage especially in applications where repeated packets can present a significantly different error rate such as in case of a wireless connection. For these applications, the possibility of selecting packets allows to avoid severely damaged ones. For instance in [37], all the possible combinations of k stored packets are considered by exploiting the availability of an error detection code to verify the correctness of their combination. For example, assume that, for a given information packet, the maximum number of transmissions allowed is four and the received packets corresponding to the four transmissions are A, B, C, and D. In this generalized majority combining scheme, after the last transmission, the receiver can combine all the possible triplets together with several choices, i.e., ABC, ABD, BCD, ACD. Each combination does not use all available packets, but it is able to obtain good recovery by avoiding potentially highly damaged packets. For instance, a combined ABC or BCD packet may have fewer errors than a combined ABCD packet.

The analysis in [38–40] shows the xor and majority combining schemes are IEEE MAC compatible, however the improvement of the throughput is not remarkable.

2.2 Hybrid ARQ Schemes

Prior work in information theory has discussed the concept of hybrid ARQ which employs various codes including Reed Solomon and LDPC for error correction [48]- [53]. In the simplest version of HARQ, type-I HARQ [49], the sender encodes the packet payload with an error-correction code prior to the transmission. Accordingly, the receiver requests for a retransmission when the decoding of the received packet fails. In type-II hybrid ARQ (HARQ-II) [50], each packet payload is encoded to a codeword and is punctured before transmission. Upon decoding failure, the receiver

buffers the packet and sends a negative acknowledgment (NACK). In response to the NACK, the sender sends additional redundancy symbols which the receiver recombines with the associated packet in the buffer and reattempts to decode the combined packet. The HARQ-II is similar to the PEEC and ACE protocols since these schemes achieve recovery through the transmission of additional redundancy. However the HARQ-II is not designed for IEEE802.11 MAC environment and is not adaptive with respect to channel condition. In addition, HARQ-II does not address throughput stability issues raised by varying traffic demand.

In addition to Hybrid ARQ (HARQ) based methods [43–45], examples of recent efforts for combating inefficiencies of ARQ-based wireless protocols include Partial Packet Recovery (PPR) [54], packet combining [32]- [47], Cross-Layer Design with Side-information (CLDS) [72]- [75], ZipTx [42]. Some of these approaches, such as PPR and packet combining, exploit physical layer information regarding the quality of individual bits to improve the probability or recovering corrupted packets. Others, such as CLDS and ZipTx utilize information available in current 802.11 link-layer protocols in conjunction with error correcting codes to recover corrupted packets. In particular, the group work on CLDS demonstrated a significant increase in throughput by utilizing corrupted packets under current 802.11 systems. Furthermore, the work on CLDS showed that the mere utility of binary side information (packet is corrupted or not), which is available in the current 802.11 link layer protocol, can increase the effective information-theoretic capacity significantly [5].

2.3 Cross-Layer Approach

In recent years, many papers in multimedia applications have proposed cross-layer mechanisms to overcome performance limitations imposed by conventional protocols. For instance UDP Lite [73], tried to improve the bandwidth utilization by making adjustments to the protocol stack

at the transport and the link layers which relies on the error-resilient nature of multimedia content. The analysis of the hybrid Erasure-Error protocols (HEEPs) in [5] shows that cross-layer protocols in general provide capacity improvement in many realistic scenarios and can significantly improve the overall performance as measured by video quality. However, a significant drawback of the cross-layer protocols is that their implementations require major modifications in transport and application layers.

The primary objective of the related studies presented in this section is to provide reliability of wireless communication. In this thesis, we analyze and develop novel link-layer protocols for wireless link-layer that in addition to providing reliability, they target other aspects of wireless communication such as flow control for delay constrained traffic, stability and traffic prioritization.

CHAPTER 3

Background

This chapter provides the background that is required to understand the contributions of this thesis.

3.1 802.11b Wireless Networks

Due to their high data rates and use of the time-tested TCP/IP protocol suite, 802.11b networks have experienced widespread deployment. These LANs are finding their way into homes and businesses ubiquitously. However, like other wireless technologies, 802.11b networks also suffer from severe quality degradation in the presence of physical obstructions and inter-symbol-interferences. Two modes of operation are supported in 802.11 networks [30, 31]: (1) ad hoc mode in which wireless nodes can communicate with each other directly, and (2) infrastructure mode in which wireless nodes are arbitrated using a central entity called an access point (AP).

All 802.11b-complaint networks support four basic physical layer data rates of 1 Mbps, 2 Mbps, 5.5 Mbps and 11 Mbps. Increase in the data rate reduces the robustness of the 802.11b physical layer. In the infrastructure mode, if the number of retransmission requests exceeds a certain threshold, the AP drops down to a lower data rate than its current data rate. For retransmissions, 802.11b relies on a 32-bit frame check sequence (FCS) that computes checksum over the entire

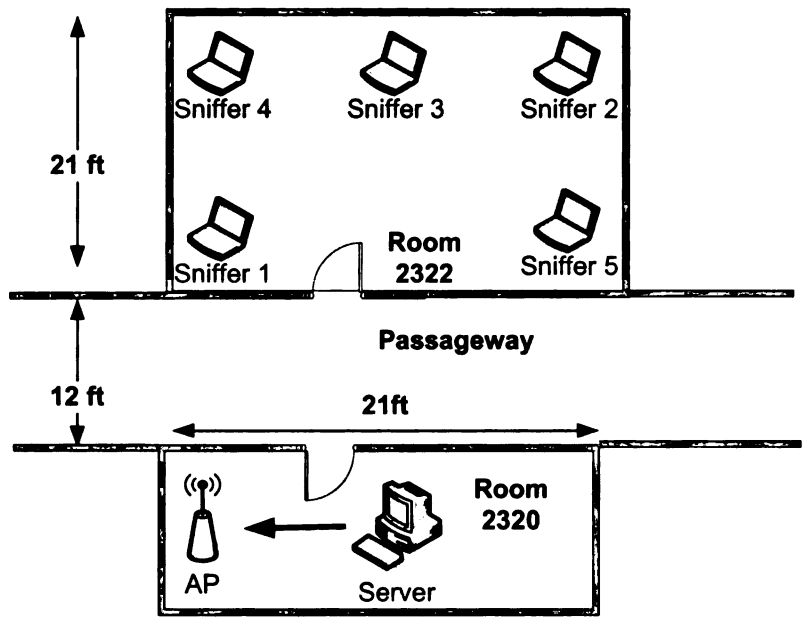


Figure 3.1: Trace collection setup.

MAC layer frame. Positive acknowledgement (ACK) frames are employed to signal successful transmission of data frames. If a frame fails checksum then it is dropped at the receivers MAC layer. The sender after timing out schedules a retransmission.

3.2 Trace Collection

Throughout this thesis, the experimental evaluations are conducted using real channel traces that are collected over the wireless setting depicted in Fig. 3.1. It can be described as follows: five wireless receivers were used to simultaneously collect error traces on an 802.11b WLAN. These receivers are placed in different locations in a room. The access point (AP) is located across the hallway from the room. A wired sender is used to send multicast packets with a predetermined payload on the WLAN; multicasting disabled MAC layer retransmissions. Each trace comprised of one million packets with a payload of 1,000 bytes each. At the physical layer, the auto rate selection feature of the AP was disabled and for each experiment the AP was forced to transmit at a fixed data rate. Each trace collection experiment was repeated for different physical layer (PHY)

Table 3.1: The average BER for different channel traces.

	2Mbps	5.5Mbps	11Mbps
500kbps	0.008	0.007	0.0094
750kbps	0.0001	0.0102	0.0090
900kbps	0.005	0.006	0.0186
1024kbps	0.0100	0.0038	0.0231

data rates (i.e., 2Mbps, 5.5 Mbps and 11Mbps). For a specific PHY data rate, we have collected traces using four transmission rates: 500kbps, 750kbps, 900kbps and 1024kbps respectively. We collected 41 traces over different receivers. However for brevity, Table 3.1 shows the channel average BERs associated to 12 of these traces.

We used Prism 2.5 Chipset WiFi adapter which allows us to modify the receiver's MAC layer device to pass corrupted packets to higher layers. To capture packets at high transmission rates, packet dissectors were implemented inside the device drivers. These packet dissectors ensured that only packets pertinent to our wireless experiment are processed, while all other packets are dropped. In addition to a packet header and payload information, for each packet two additional parameters (1) *Signal strength* (S), (2) *Silence Value* (N) were logged at the receivers. These parameters are used to calculate the Signal to Silence Ratio (SSR) value (i.e., $SSR = S - N$) observed with each packet.

3.3 Discrete-Time Markov Chains

Markov chains are employed to model statistical data with short-term temporal dependence. Let a stochastic process X_n take on values denoted by non-negative integers $0, 1, \cdots, N$. If $X_n = i$ then the process is said to be in state i at time n . Whenever the process is in state i there is a fixed

probability that the next state of the process will be state j . If that probability can be expressed as

$$P(X_{n+1} = j|X_n = i, X_{n-1} = i_{n-1}, \cdots, X_1 = i_1, X_0 = i_0) = P(X_{n+1} = j|X_n = i), \tag{3.1}$$

for all states $i_0, \cdots, i_{n-1}, i, j, n \geq 0$, then X_n is a Markov Chain.

The property given in 3.1 is commonly referred to as the Markov Property. Thus, for a Markov chain the conditional distribution of any future state X_{n+1}, given the past states $X_{n-1}, \cdots, X_1, X_0$ and the present state X_n , is independent of the past states and depends only on the present state. Equation (3.1) is also referred to as homogeneity property since it ensures that the transition probabilities do not vary with time.

Let $p_{i,j} = P(X_{n+1} = j|X_n = i)$ denote the probability of transiting to state j from i. Since $p_{i,j}$ represents a probability measure, it exhibits the following properties: (i) $p_{i,j} \geq 0, i, j \geq 0$, and (ii) $\sum_j^N p_{i,j} = 1, i = 0, 1, \cdots, N$. The probability of transiting to the next state can be represented in a matrix form. This matrix is referred to as the one-step state transition probability matrix.

The steady-state or stationary probabilities of a Markov chain represent the long-run proportion of the time spent in each state. Once the transitional probabilities of a Markov chain are known, the steady-state probabilities of being in a particular state are the unique non-negative solutions of the following linear system of equations:

$$\sum_i^N \pi_i p_{i,j} = \pi_j, \quad j = 1, \cdots, N \tag{3.2}$$

$$\sum_{j=1}^N \pi_i = 1. \tag{3.3}$$

For stationary Markov chains, the steady-state and transition probabilities do not vary with time. Throughout this thesis, we use stationary Markov chain for modeling wireless channel in every transmission.

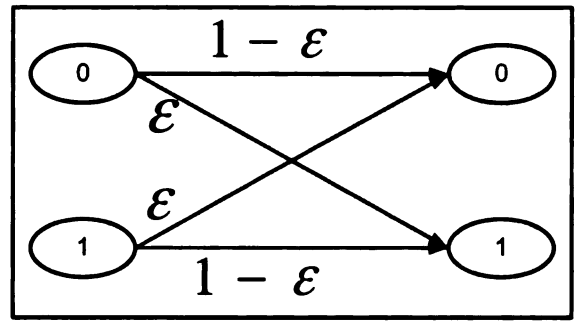


Figure 3.2: A binary symmetric channel with crossover probability ϵ.

3.4 Binary Symmetric Channel

A binary symmetric channel (or BSC) is a common communication channel model used in coding theory and information theory. In this model, a transmitter wishes to send a bit (a zero or a one), and the receiver receives a bit. It is assumed that the bit is usually transmitted correctly, but that it will be "flipped" with a small probability (the "crossover probability"). This channel is used frequently in information theory because it is one of the simplest channels to analyze.

The BSC is a binary channel; that is, it can transmit only one of two symbols (usually called 0 and 1). (A non-binary channel would be capable of transmitting more than 2 symbols, possibly even an infinite number of choices) The transmission is not perfect, and occasionally the receiver gets the wrong bit. Many problems in communication theory can be reduced to a BSC. On the other hand, being able to transmit effectively over the BSC can give rise to solutions for more complicated channels.

The BSC channel in Fig. 3.2 with crossover probability ϵ is a channel with binary input and binary output and probability of error ϵ; that is, if X is the transmitted random variable and Y the

received variable, then the channel is characterized by the conditional probabilities

$$P(Y = 0|X = 0) = 1 - \epsilon \tag{3.4}$$

$$P(Y = 0|X = 1) = \epsilon \tag{3.5}$$

$$P(Y = 1|X = 0) = \epsilon \tag{3.6}$$

$$P(Y = 1|X = 1) = 1 - \epsilon. \tag{3.7}$$

It is assumed that $0 \leq \epsilon \leq 0.5$. If $\epsilon > 0.5$, then the receiver can swap the output (interpret 1 when it sees 0, and visa versa) and obtain an equivalent channel with crossover probability $1 - p \leq 0.5$. The capacity of the channel is $1 - H(\epsilon)$, where $H(\epsilon)$ is the binary entropy function [55].

3.5 Markovian Wireless Channel

A *channel model* describes the process under which errors are introduced in a transmitted packet over a wireless link. Packets are transmitted during discrete time slots τ_i, $i = 1, 2, \cdots, +\infty$, which we refer to as transmission intervals. During the i^{th} transmission interval, a message is transmitted over a Binary Symmetric Channel (BSC) with cross-over BER ϵ_i. To derive a channel model for all transmission intervals, we assume that each ϵ_i of a particular τ_i is valued from a finite set F_N with length N: $\epsilon_i \in F_N, |F_N| = N$. As a result, we can consider the channel model as a combination of N various BSCs with unique BERs (i.e., $\epsilon_l \neq \epsilon_j$ for $l \neq j$ $l, j = 1, 2, \cdots, N$). In every τ_i, the channel is in one of the N possible states $(S_1, \cdots, S_N)$ where each state corresponds to a particular BSC. Based on these settings, we can model a wireless channel as a discrete Markov chain with N states where each state is a representation of a BSC with a particular BER. Fig 3.3 shows a Markovian channel model. We assume a homogenous and stationary Markov chain with transitional probability matrix $\mathbf{P}$ and the limiting probabilities $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_N)$. Under this

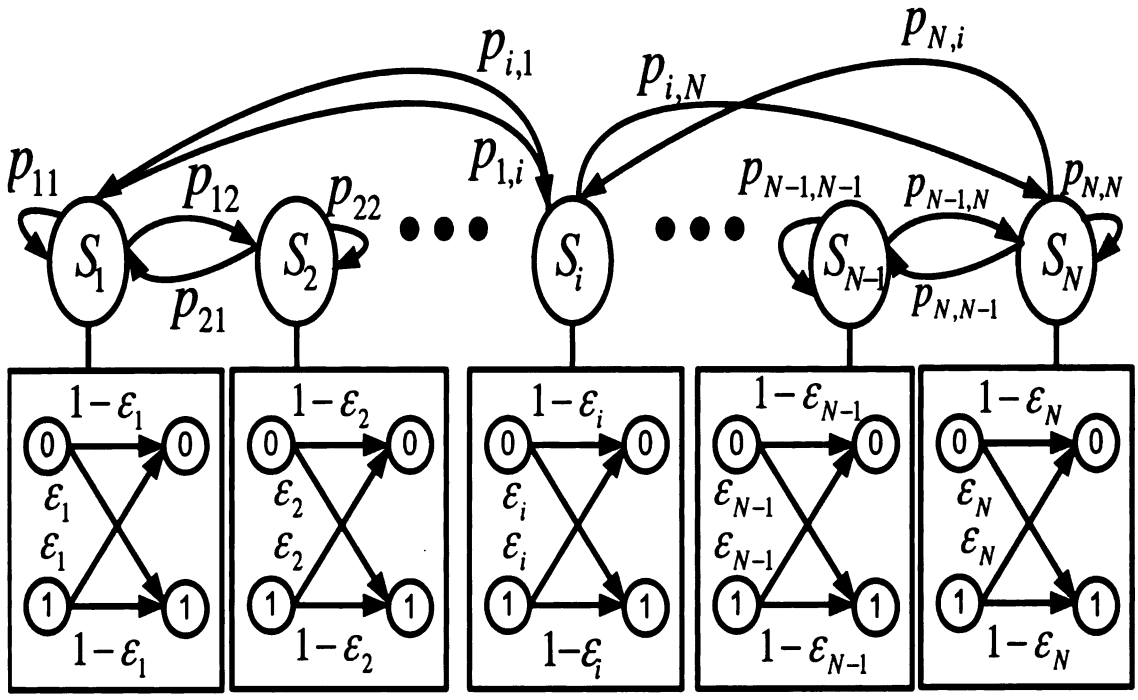


Figure 3.3: Markovian Channel.

model, the steady state average BER $\bar{\epsilon}$ and packet error rate (PER) ς is

$$\bar{\epsilon} = \sum_{i=0}^{N} \pi_i \epsilon_i \tag{3.8}$$

$$\varsigma = \sum_{i=0}^{N} \pi_i (1 - (1 - \epsilon_i)^n), \tag{3.9}$$

where n is the length of the transmitted packet.

The probability that the channel is in states $S_i = r$ and $S_{i+1} = t$ in τ_i and τ_{i+1} is:

$$Pr\{S_i = r, S_{i+1} = t\} = \pi_r p_{rt}. \quad r, t = 1, 2, \cdots, N \tag{3.10}$$

The Markovian channel model can be trained on real channel traces by using the statistics of previous transmission intervals. This captures the effects of multipath fading and interference on the channel BER in every transmission interval using a single aggregated model [19,21].

The capacity of a BSC channel with cross-over probability ϵ_i is $1 - H(\epsilon_i)$ [55]. Using the

steady state probabilities π_i the average channel capacity in any interval is determined as follows:

$$C = \sum_{i=1}^{N} \pi_i (1 - H(\epsilon_i)). \tag{3.11}$$

The channel capacity gives an upper bound on the average (reliable) information transmission rate for the wireless channel under consideration.

CHAPTER 4

PEEC: A Channel-Adaptive Feedback-Based Error Control Protocol for Wireless MAC Layer

In this chapter, we investigate the problem of reliable wireless link-layer communication over a lossy channel. Specifically, we introduce an analytical framework that presents a thorough analysis and modeling of a generic link-layer point-to-point wireless communication which employs channel codes to provide and maintain a reliable communication. Our objective is to develop a new wireless link-layer framework which make use of incremental channel codes to provide maximum *reliability* (high likelihood of successful recovery data at the receiver) while ensures an efficient utilization of available bandwidth in transmission of new data.

This framework comprises various models including *communication*, *message*, *distortion* and *buffer* models. The analysis provided under this framework leads to the development of a novel link-layer Packet Embedding Error Control (PEEC) protocol [3].

PEEC employs packet-embedded parity symbols (instead of retransmission) for error recovery. Further, PEEC uses certain flags embedded in the receiver feedback to determine the amount of

redundancy necessary for the next transmission. PEEC is similar to the HARQ-II protocol in the sense that in both schemes, the recovery is achieved through the transmission of additional redundancy information; however PEEC utilizes acknowledgment flags, a *decoding* and a *buffer*, to assess the channel and the receiver buffer conditions in every transmission. Depending on this assessment, PEEC adaptively administers the transmission of the data and redundancy (parity) symbols such that: (1) the level of parity symbols guarantees high likelihood of successful recovery of new data as well as corrupted data at the receiver buffer; (2) the available bandwidth is efficiently utilized for the transmission of new data. Another design objective of PEEC is its compatibility with conventional higher-layer transport and application layer protocols. That is, unlike cross-layer protocols, PEEC does not pass corrupted packets up to the higher layers of the protocol stack.

4.1 Communication Scheme

In this section, we describe a general contention free communication model in which one transmitter and one receiver are communicating over a wireless link. We define a transmission interval τ_i as the duration in which a transmitter sends the i^{th} message M_i and receives its corresponding acknowledgment ACK_i. A transmitter sends a new message after the reception of an acknowledgment (ACK). In our analysis we assume that the ACK message is error-free. This assumption is reasonable since the size of the ACK messages are small and often protected by FEC [102]. Alternatively, the ACK frames can be transmitted over low PHY rates to reduce the probability of corruption.

4.1.1 Sender Side

During τ_i, a sender transmits a message which is represented by the tuple $M_i = (C_i(k_i, x_i), \mathbf{y}_i)$ where k_i represents the number of data symbols which are not being retransmitted. In each τ_i, a transmitter encodes k_i with parity symbols x_i creating a codeword $C_i(k_i, x_i)$. We refer to these parity symbols as type-I parity. The receiver utilizes x_i to decode C_i. Upon successful decoding, C_i is extracted and k_i data symbols are passed up to the higher layer. The error correction fails when the decoding operation fails as indicated in FCS. In that case, the receiver stores C_i in its buffer and issues a request for more parity symbols. The transmitter also sends additional (type-II) parity symbols denoted by $\mathbf{y}_i$. The receiver utilizes $\mathbf{y}_i$ symbols to recover old corrupted codewords accumulated in its buffer (e.g., $C_j, j = 1, \cdots, i-1$).

4.1.2 Receiver Side

We assume that the receiver has a finite buffer containing m rooms. That is, the receiver can accommodate up to m corrupted messages waiting for recovery. If a newly corrupted packet finds all rooms in the buffer occupied, it does not enter the buffer and is dropped.

We assume that there are $s, 1 \leq s \leq m$, decoders available in the receiver. The status of the receiver is reported to the transmitter via certain flags in an acknowledgment message. Specifically, $1 + s$ flags are encapsulated in every acknowledgement. Let $F_i[k], k = 0, \cdots, s$ represent values of these flags in ACK_i. We refer to the first flag (i.e., $F_i[0]$) as a *decoding flag*, indicating the decoding status of the transmitted message in τ_i; $F_i[0] = 1$ when decoding of M_i was *not* successful. The remaining s flags (i.e., $F_i[k], k = 1, \cdots, s$) are called *buffer flags*. Each buffer flag is associated with a particular decoder in the buffer and represents the status of that decoder. For instance, if the j^{th} decoder is busy then $F_i[j] = 1$.

To perceive the functionality of this communication model, consider the example given in

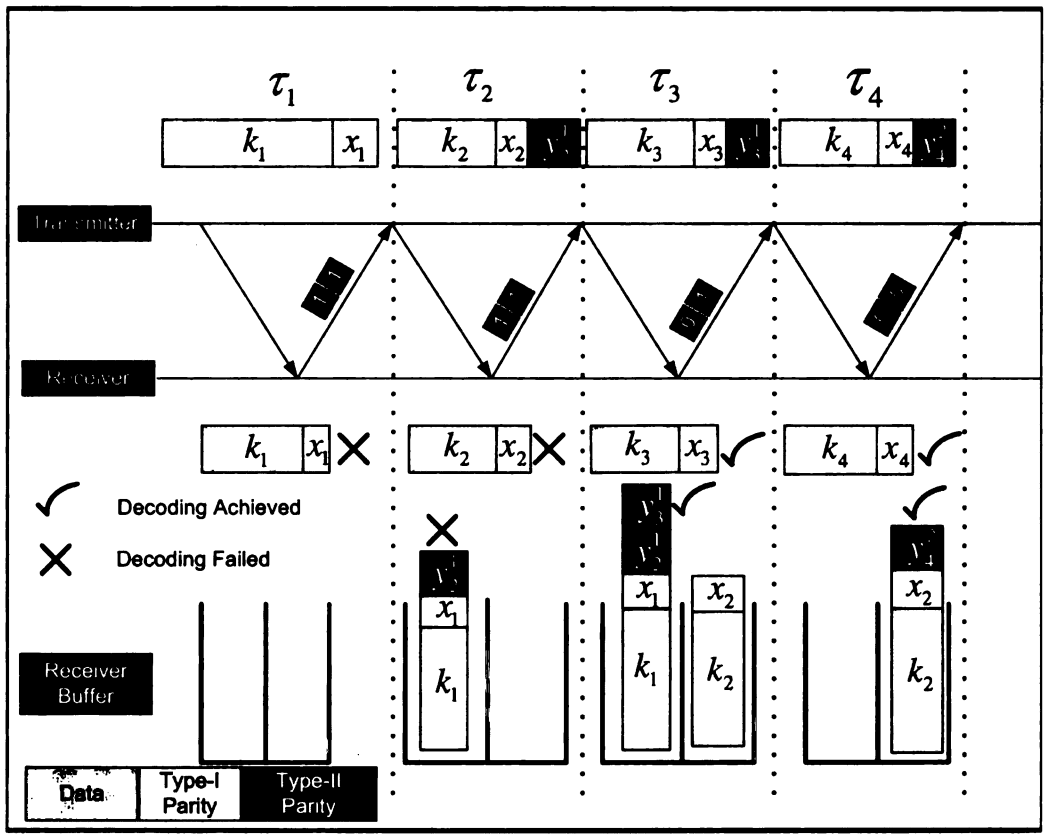


Figure 4.1: An example of communication model consists of four transmission intervals.

Fig. 4.1. A short communication consisting of four transmission intervals, buffer capacity of two and one decoder at the receiver is shown. During the first transmission interval τ_1, a message $M_1 = (C_1(k_1, x_1), \mathbf{y}_1)$ is sent. There are no type-II parity symbols in M_1, because there is no prior corrupted message in the receiver buffer, so $\mathbf{y}_1 = \mathbf{0}$. A receiver that fails to decode C_1, stores C_1 in its buffer and sends an acknowledgment $ACK_1 = (1, 1)$. In τ_2, the transmitter sends $M_2 = (C_2(k_2, x_2), \mathbf{y}_2 = \{y_2^1\})$. The receiver uses x_2 to decode C_2 and employs type-II parity symbols y_2^1 (y_i^j denote additional parity for $C_j, j < i$ transmitted in τ_i) in addition to x_1 to decode C_1. The receiver acknowledges $ACK_2 = (1, 1)$, indicating decoding failure of C_2 and $C_1(k_1, x_1 + y_2^1)$. As a result, in τ_3, the sender sends $M_3 = (C_3(k_3, x_3), \mathbf{y}_3 = \{y_3^1\})$. Note that since there is only one decoder in use, the sender transmits type-II parity which corresponds to a particular message that the decoder is serving. In τ_3, the receiver successfully decodes C_3

using x_3 and $C_1(k_1, x_1 + y_2^1 + y_3^1)$. Now, since decoding of the recent transmitted message was successful, the receiver sets the decoding flag to zero (i.e., $F_3[0] = 0$) but at the same time since the decoder is waiting for type-II parity symbols to perform decoding on C_2, the buffer flag is set to one (i.e., $F_3[1] = 1$); so $ACK_3 = (0, 1)$. Accordingly, in τ_4, the sender transmits $M_4 = (C_4(k_4, x_4), \mathbf{y}_4 = \{y_4^2\})$. The receiver decodes C_4 using x_4 and $C_2 = (k_2, x_2 + y_2^4)$ successfully; so $ACK_4 = (0, 0)$.

The communication model described above represents a general communication mechanism that utilizes packet-embedded parity symbols instead of retransmission to retrieve corrupted data. For this communication scheme, developing a protocol that estimates and adjusts the amount of necessary redundancy in every transmission, requires several controlling subtleties that we need to handle carefully to achieve high performance. For instance, depending on the feedback from the receiver, a transmitter should send enough type-I parity symbols to increase the likelihood of reliable reception. On the other hand, a reasonable amount of type-II redundancy is required to correct corrupted packets and prevent buffer overflow. Sending excessive redundancy will deteriorate average bandwidth utilization; meanwhile, transmitting too few parity symbols may result in unsuccessful decoding and will increase the number of corrupted messages accumulated in the buffer. Therefore, the amount of redundancy transmitted over a channel has a critical impact on overall performance of this communication scheme. To that end, we propose an error control mechanism called Packet Embedded Error Control protocol (PEEC). In the next section, we study underlying statistical characteristics of this communication model and develop necessary models which provide essential tools for the design and analysis of PEEC. Note that, throughout this chapter, the terms "packet" and "message" are used interchangeably.

4.2 Theoretical Settings

In this section, we investigate the statistical nature of the communication scheme described in the previous section to find a suitable model for each of its components. We introduce three models: a *message model* that represents the distribution of the parity and data symbols in a particular message; *a distortion model* that measures the distortion level of a message after a transmission; and *a buffer model* which models the receiver buffer as a queuing system.

4.2.1 Message Model

Let random variables K, X and Y represent the number of data, type-I and type-II parity symbols in a message. Consider a message $M = (K, X, Y)$ containing n symbols. That is, any message M is represented by a vector of three random variables and has a fixed length $K+X+Y = n$. Let p_K be the probability measure that a particular symbol in a message is a data symbol. Similarly, let p_X (p_Y) denote this likelihood for type-I (type-II) parity symbol. Therefore, K, X and Y each, has a binomial distribution with parameters (n, P_K), (n, P_X) and (n, P_Y) respectively.

Finding a model for a message M is equivalent to finding a distribution of the vector (K, X, Y). Since K, X and Y are binomial random variables, a multinomial distribution can be used as an accurate approximation of the distribution of $M = (X, Y, Z)$ with the probability mass function

$$Pr\{K = k, X = x, Y = y\} = \frac{n!}{k!x!y!} p_K^k p_X^x p_Y^y, \quad k + x + y = n$$

where $p_K + p_X + p_Y = 1$.

In practice, since n is a large number, we can approximate the distributions of K, X, and Y by Poisson distributions with parameters $\lambda_K = np_K$, $\lambda_X = np_X$ and $\lambda_Y = np_Y$. So for instance, K has the following probability mass function:

$$P\{K = k\} = e^{-\lambda_K} \frac{\lambda_K^k}{k!} \quad k = 0, 1, 2, \cdots, n. \tag{4.1}$$

X and Y carry similar probability mass function with rates λ_X and λ_Y respectively.

4.2.2 Distortion Model

Suppose the transmitter sends a message $M_i = (k_i, x_i, \mathbf{y}_i)$ to the receiver through the channel which is a BSC with BER ϵ_i during the transmission interval τ_i. So each symbol in M_i can be distorted independently from the other symbols. Thus, the distortion of each symbol has a Bernoulli distribution with parameter ϵ_i. As a result, the distortion level of M_i in a transmission interval τ_i, denoted by D_i, has a binomial distribution with parameters (n, ϵ_i). In practice, n is a relatively large number, and ϵ_i is very small. Accordingly, we can approximate D_i with a Poisson distribution with rate $\lambda_{D_i} = n\epsilon_i$.

The receiver decodes M_i and utilizes x_i to correct possible errors in the message. Depending on the decoding algorithm, the receiver can correct some of the corruption proportional to the number of parity symbols in the message. That is, if M_i carries x_i parity symbols, then the receiver is capable of retrieving up to $\alpha \times x_i$ error symbols in the message. Here α measures the expected error-correcting capability of a particular decoder. For example, the error-correcting capability of Reed-Solomon codes is half as many as redundant symbols (i.e., $\alpha = 0.5$).

The distortion level D_i is random and unknown to the receiver and therefore the notion of partial recovery is unrealistic in error correction. That is, the receiver can either correct all errors in M_i and acknowledges successful decoding or just assumes that no recovery is achieved. So the level of distortion in M_i *after decoding*, denoted by U_i, is

$$U_i = \begin{cases} 0 & D_i \leq \alpha x_i \\ D_i & \text{otherwise} \end{cases} \tag{4.2}$$

Equation (4.2) shows that the distribution of U_i is equivalent to the distribution of D_i *truncated* on αx_i (see Fig. 4.2). So, the probability of successful decoding of M_i is equivalent to the probability that $U_i = 0$. That is,

$$P(U_i = 0) = P(D_i \leq \alpha x_i) = \sum_{d=0}^{\lfloor \alpha x_i \rfloor} e^{-\lambda_{D_i}} \frac{\lambda_{D_i}^d}{(d)!}. \tag{4.3}$$

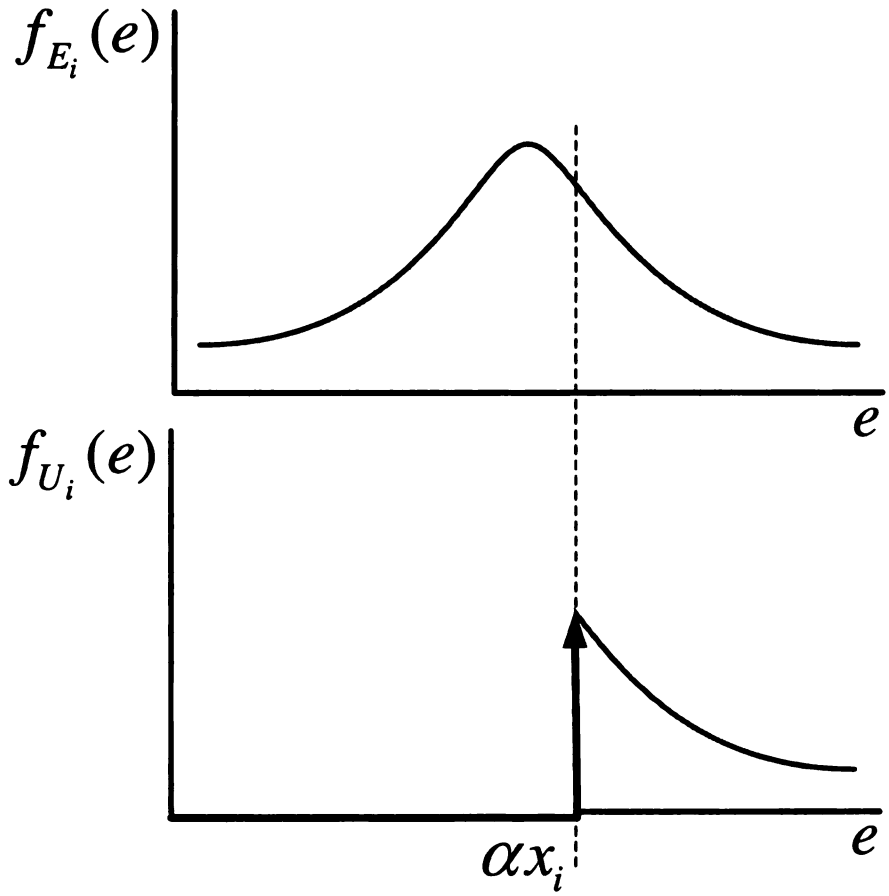


Figure 4.2: The density of error *after decoding* is truncated on αx_i.

Equation (4.3) clearly illustrates that the probability of successful decoding is directly related to the amount of available redundancy (i.e., x_i) and the decoder error-correcting capability (i.e., α). This relationship is presented in the following lemma.

Lemma 1. *In a transmission interval τ_i with error probability ϵ_i, the ν likelihood of successful decoding of a message M_i with type-I parity probability distribution p_{X_i} can be achieved if*

$$p_{X_i} \geq \frac{1}{n\alpha}\left(n\epsilon_i + \sqrt{n\epsilon_i}\phi^{-1}(\nu)\right),$$

where $\phi^{-1}(\nu)$ is the ν-quantile of the standard normal distribution.

Proof. See Appendix A. □

4.2.3 Buffer Model

The receiver has a finite buffer with the capacity of m messages. A message enters the buffer when its decoding with type-I parity symbols has failed. Meanwhile, a message leaves the buffer when its decoding with additional type-II parity symbols was successful or it is timed out. Any particular message in the buffer is being served with an individual decoder where a decoder uses incoming type-II parity symbols to correct that message. Let us assume there are s identical decoders in the receiver that can serve up to m messages in the buffer simultaneously. The buffer can be modeled as the $M/M/s/m$ queuing system with s servers and m buffer slots. Each server (decoder) serves a particular costumer (message) in the queue and the queue has the capacity m. In this system, if a new corrupted packet finds all of the slots busy, it does not enter the queue and considered as lost to the system.

The $M/M/s/m$ queuing system assumes that messages arrive according to the Poisson process with rate λ and the service time has exponential distribution with mean $\frac{1}{\mu}$. To determine these rates, we let λ_i and μ_i represent the corresponding arrival and service rates when a channel is in state S_i with BER ϵ_i. According to the distortion model, a message enters the buffer if $D_i > \alpha X_i$. Thus, by using Equation (4.3), arrival rate is

$$\lambda_i = Pr\{D_i > \alpha X_i\} \tag{4.4}$$

$$= 1 - \sum_{x=0}^{n} Pr\{X_i = x\} Pr\{D_i \leq \alpha x\} \tag{4.5}$$

$$= 1 - \exp\left[-n(\epsilon_i + p_{X_i})\right] \sum_{x=0}^{n} \sum_{d=0}^{\lfloor \alpha x \rfloor} \frac{n^{d+x} \epsilon_i^d p_{X_i}^x}{d!x!}. \tag{4.6}$$

Any particular decoder utilizes type-II parity symbols in addition to the original type-I parity symbols to decode a particular message. So the service rate is equivalent to the probability of suc-

cessful decoding using type-I and type-II parity symbols. As a result, according to the distortion model,

$$\mu_i = Pr\{D_i \leq \alpha(X_i + Y_i)\}.$$

Recall that $Z_i = X_i + Y_i$ which is the sum of two independent Poisson variables, is in fact a Poisson random variable with rate $\lambda_{Z_i} = \lambda_{X_i} + \lambda_{Y_i}$. Correspondingly, the service rate is

$$\mu_i = \exp\left[-n(\epsilon_i + p_{Z_i})\right] \sum_{z=0}^{n} \sum_{d=0}^{\lfloor \alpha z \rfloor} \frac{n^{d+z} \epsilon_i^d p_{Z_i}^z}{d!z!}, \tag{4.7}$$

where $p_{Z_i} = p_{X_i} + p_{Y_i}$.

The channel is in the state S_i with the probability of π_i, therefore the overall arrival and service rates for the buffer is

$$\lambda = \sum_{i=1}^{N} \pi_i \lambda_i \tag{4.8}$$

$$\mu = \sum_{i=1}^{N} \pi_i \mu_i. \tag{4.9}$$

From equations (4.6) and (4.7), we observe that λ_i and μ_i are functions of the parity symbols distribution parameters p_{X_i} and p_{Y_i}. Each pair of p_{X_i} and p_{Y_i} introduces a particular value for λ_i and μ_i. This verifies our intuitive claim that average amount of parity symbols allocated in every transmission has a critical impact on the behavior of the buffer and ultimately the overall performance.

4.3 PEEC Protocol

The error-combating scheme of the communication model of Section 4.1 is Forward Error Correction (FEC). That is, a receiver uses redundancy symbols that are embedded in the transmitted message to correct errors. In contrast to the IEEE standard ARQ mechanism, the receiver does

not drop a packet unless its buffer is full. As the steady state analysis of the receiver buffer shows, the average amount of redundancy in a message has direct impact on the buffer behavior and the overall performance. Transmitting too few parity symbols in every transmission may result in frequent decoding errors, causing buffer overflow. On the other hand, transmitting many parity symbols may result in degrading the bandwidth utilization (more parity symbols are transmitted than data symbols). In both scenarios, the average amount of data that are decoded successfully and delivered to the upper layer decays. So, it is important for an error-control mechanism to adaptively find an accurate tradeoff between the number of data and parity symbols transmitted in every interval.

In this section, we present the PEEC protocol that uses the receiver acknowledgement to determine the amount of redundancy necessary for the next transmission. Specifically, PEEC utilizes *decoding* and *buffer* flags to assess the status of the channel condition and the receiver buffer. Depending on this assessment, PEEC evaluates the amount of data symbols and parity symbols necessary to send in the next transmission. Fig. 4.3 illustrates the PEEC in more details. It shows that the PEEC performs two important procedures namely *Channel State Estimation* and *Redundancy Allocation.*

4.3.1 Channel State Estimation

PEEC uses the *decoding* flag of the acknowledgment of the previous transmission interval (i.e., $F_{i-1}[0]$) to estimate the state of the channel in the next transmission interval τ_i. According to the Markovian wireless channel model presented in Chapter 3, each state is a representation of a BSC with a particular BER. Correspondingly, PEEC uses $F_{i-1}[0]$ to guess the BER of BSC in τ_i. For instance, a decoding failure ($F_{i-1}[0] = 1$), indicates high BER, meaning that, the amount of type-I parity symbols in the message was not sufficient to correct distortion. Similarly, a successful decoding ($F_{i-1}[0] = 0$) shows that the BER was low. The error-correcting capability

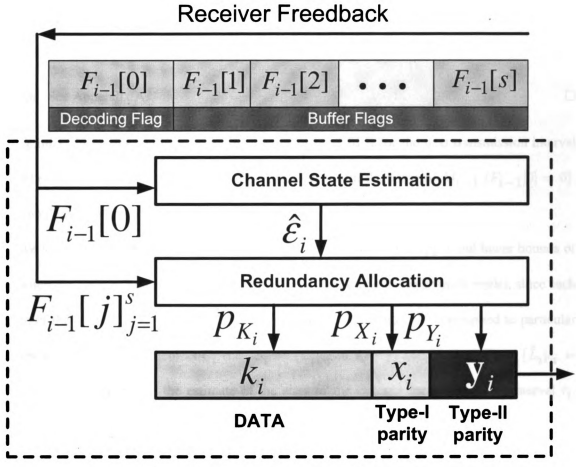


Figure 4.3: The architecture of the PEEC protocol

of a decoder and the amount of parity symbols that are sent in the previous transmission have a direct impact on our estimation. The boundedness of the channel state estimate is given by the following two Lemmas. Note that, we use the notation $\hat{\epsilon}$ to represent ϵ estimate.

Lemma 2. *In τ_{i-1}, if a decoder with error-correcting capability α decodes a message with type-I parity symbol rate $p_{X_{i-1}} = g(\hat{\epsilon}_{i-1})$ successfully, then for τ_i, $\hat{\epsilon}_i$ has the upper bound*

$$\hat{\epsilon}_i \leq g^{-1}\left(\frac{\hat{\epsilon}_{i-1}}{\alpha}\right).$$

Proof. See Appendix A □

Lemma 3. *In τ_{i-1}, if a decoder with error-correcting capability α fails to decode a message with type-I parity symbol rate $p_{X_{i-1}} = g(\hat{\epsilon}_{i-1})$, then the estimation of BER for τ_i has a lower*

bound

$$\hat{\epsilon}_i \geq \alpha g(\hat{\epsilon}_{i-1}).$$

Proof. See Appendix A. □

Lemma 2 provides an upper bound for the estimate of BER for the next transmission interval τ_i when the decoding flag in ACK_{i-1} indicates successful decoding of M_{i-1} ($F_{i-1}[0] = 0$). Lemma 3 determines a lower bound when $F_{i-1}[0] = 1$.

We let $\hat{\epsilon}_i^{(U)} = g^{-1}\left(\frac{\hat{\epsilon}_{i-1}}{\alpha}\right)$ and $\hat{\epsilon}_i^{(L)} = \alpha g(\hat{\epsilon}_{i-1})$ represent the upper and lower bounds of $\hat{\epsilon}_i$ when $F_{i-1}[0] = 0$ and $F_{i-1}[0] = 1$ respectively. According to the channel model, since each state of the channel corresponds to a particular BER, so $\hat{\epsilon}_i^{(U)}$ and $\hat{\epsilon}_i^{(L)}$ correspond to particular states which we denote them as $\hat{U}_i$ (i.e., $S_i = \{\hat{U}_i | \epsilon_i = \hat{\epsilon}_i^{(U)}\}$) and $\hat{L}_i$ (i.e., $S_i = \{\hat{L}_i | \epsilon_i = \hat{\epsilon}_i^{(L)}\}$). Let $\hat{S}_i$ represent the estimate of the state of the channel for transmission interval τ_i. According to Lemma 2, when $F_{i-1}[0] = 0$, a candidate for $\hat{S}_i$ should be selected from a set $A = \{1, 2, \cdots, \hat{U}_i\}$. To make the best guess, we calculate the joint probability of $S_{i-1} = \{r | \epsilon_r = \hat{\epsilon}_{i-1}\}$, which represents the state of the channel in τ_{i-1} and $S_i = \{w | w \in A\}$, which represents one of the possible state of the channel in τ_i. Using equation (3.10), we choose the state that introduces the maximum joint likelihood as our best estimate for the state of the channel in τ_i. Therefore,

$$\hat{S}_i = \arg\max_w Pr\{S_{i-1} = r, S_i = w\}. \quad w \in A$$

Using similar technique, we can find the best $\hat{S}_i$ when $F_{i-1}[0] = 1$.

4.3.2 Redundancy Allocation

PEEC uses $\hat{S}_i$ along with *buffer* flags (i.e., $F_{i-1}[j]$ $j = 1, \cdots, s$) in ACK_{i-1} to determine the distribution of data and redundancy symbols of the message in τ_i, namely p_{X_i}, p_{Y_i}, and p_{K_i}.

PEEC uses $\hat{S}_i$ to determine p_{X_i}. Recall that, p_{X_i} is the rate of type-I redundancy in the message. That is, p_{X_i} represents the minimum number of symbols required to correct each data symbol transmitted over the BSC channel with BER $\hat{\epsilon}_i$. For this BSC, there is an uncertainty about the correctness of each transmitted symbols imposed by $\hat{\epsilon}_i$. Conceptually, we can resolve this uncertainty by sending additional information to the receiver which shows whether the channel caused an error. This is equivalent to sending a redundancy that informs the receiver about the BSC status with the transmission of every symbol. This redundancy have a minimum number of symbols that is the same as the entropy of $\hat{\epsilon}_i$. Thus, we have

$$p_{X_i} = H(\hat{\epsilon}_i) = \hat{\epsilon}_i \log_2(\frac{1}{\hat{\epsilon}_i}) + (1 - \hat{\epsilon}_i) \log_2(\frac{1}{1 - \hat{\epsilon}_i}).$$

According to our communication model each buffer flag in the acknowledgment message ($F_{i-1}[j]$ $j = 1, \cdots, s$) indicates the status of a particular decoder in the receiver. Let $p_{Y_i} = (p_{Y_{i1}}, p_{Y_{i2}}, \cdots, p_{Y_{is}})$, where $p_{Y_{ij}}$ shows the rate of type-II redundancy symbols allocated for the j^{th} decoder. Intuitively if $F_{i-1}[j] = 0$ then $p_{Y_{ij}} = 0$. That is, it is unnecessary to allocate redundancy for an idle decoder. On the other hand, if $F_{i-1}[j] = 1$, it means that the j^{th} decoder is working on a specific message $M^{(j)}$ in the buffer. To determine $p_{Y_{ij}}$, let us assume that $M^{(j)}$ has been sent in the transmission interval τ_{i-1}. Therefore, from the sender perspective, the uncertainty of corruption in $M^{(j)}$ is dictated by $\hat{\epsilon}_{i-1}$ (the estimated BER of BSC in τ_{i-1}). Furthermore, every symbol in type-II parity symbols associated with $M^{(j)}$ (say $y^{(j)}$) transmitted in τ_i might be distorted with the probability $\hat{\epsilon}_i$. Hence, the overall uncertainty of corruption in the new message $(M^{(j)} + y^{(j)})$ is dictated by the combination of the BERs of BSCs in τ_{i-1} and τ_i. This is equivalent to assuming that the message $(M^{(j)} + y^{(j)})$ has been transmitted over a single BSC which is composed of two cascaded BSCs with parameters $\hat{\epsilon}_{i-1}$ and $\hat{\epsilon}_i$. So, the overall error in $(M^{(j)} + y^{(j)})$ is caused by a channel with BER,

$$\hat{\epsilon}_i^{(j)} = \hat{\epsilon}_{i-1}(1 - \hat{\epsilon}_i) + \hat{\epsilon}_i(1 - \hat{\epsilon}_{i-1}).$$

Thus,

$$p_{Y_{ij}} = H\left(\hat{\epsilon}_i^{(j)}\right).$$

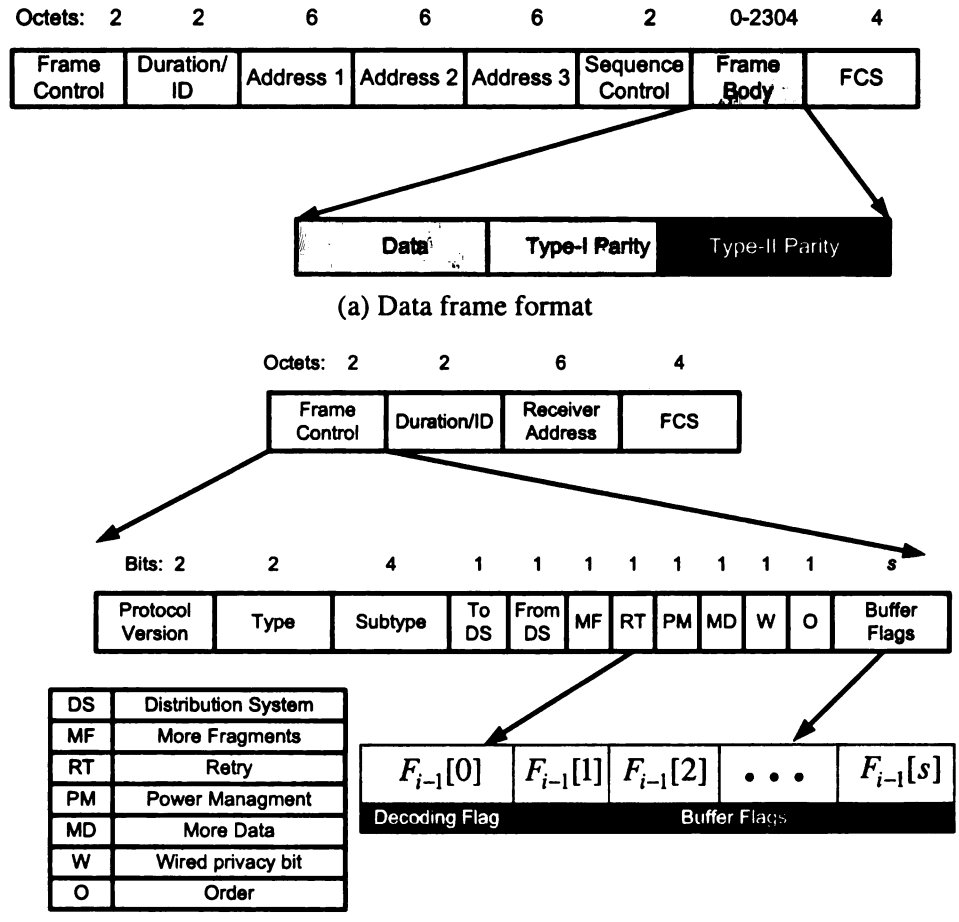
Notice that similar technique is applicable for the messages that have entered the buffer before τ_{i-1} and have already utilized some amount of type-II parity symbols. According to the message model, a valid message distribution has to satisfy $p_{K_i} + p_{X_i} + p_{Y_i} = 1$. Thus,

$$p_{K_i} = 1 - H(\hat{\epsilon}_i) - \sum_{j=1,\, F_{i-1}[j]=1}^{s} H\left(\hat{\epsilon}_i^{(j)}\right).$$

The transmitter constructs a message M_i according to the message distribution specified by PEEC. This distribution is adaptively computed in every transmission interval based on the channel and the receiver buffer conditions. Therefore, the overall throughput of the communication model increases when the PEEC protocol is utilized. To ascertain the capability of PEEC, in the next section, we perform experimental analysis for the proposed protocol using real channel traces. Meanwhile, in the next subsection, we describe the changes necessary in the current IEEE 802.11 standard to satisfy PEEC requirements.

4.3.3 MAC Frame Structure

The PEEC protocol is designed to provide reliable data delivery at the IEEE802.11 link-layer. Fig. 4.4 shows the IEEE802.11 data and ACK frame formats. The IEEE802.11 link-layer only employs an ARQ mechanism, and hence, a Frame Check Sequence (FCS) is used for error detection. For the proposed scheme, we modify the data and ACK frame structures as shown in Fig. 4.4 to incorporate fields that used by PEEC. First, we modify the *frame body* field and divide it into three fields for data, type-I and type-II parity bits. Further, we modify the *frame control* (FC) field in the ACK frame to incorporate *decoding flag* and *buffer flags*. Specifically, we redefine the *retry* field to serve as the *decoding flag*.



(a) Data frame format

(b) ACK frame format

Figure 4.4: IEEE802.11 MAC frame formats and corresponding modifications necessary for PEEC.

As eluded above, when the *decoding flag* is set to zero, this signals to the transmitter that the last transmitted packet was decoded successfully. However, when the *decoding flag* is set to one, this signals to the transmitter that additional decoding bits are (still) needed for the last transmitted packet (which was just transmitted and now is waiting to be correctly decoded at the receiver buffer). The *buffer flags* indicate the status of one or more (up-to s) previously corrupted packets, which are being served by one or more s decoders at the receiver. Recall that these s corrupted packets were not successfully decoded in prior attempts. Therefore, depending on the number of decoders utilized, we increase the length of the FC field by s bits for additional s *buffer flags*. In the next section, we observe that in practice, by utilizing a single *buffer flag* that corresponds

to a decoder with error-correcting capability close to that of perfect codes, the feedback in the proposed scheme can be limited to only one additional bit to the current IEEE802.11 standard FC field.

4.4 Experimental Analysis

In this section, we evaluate the performance of the PEEC protocol on real channel traces collected on an 802.11b WLAN described in chapter 3. First, we analyze the impact of the number of decoders and their error-correcting capability factor on the performance. Next, we compare the performance of the PEEC protocol with the IEEE standard ARQ, enhanced ARQ schemes and HARQ mechanisms in a practical scenario where the receiver uses a single decoder. Finally, we evaluate the performance of PEEC using an adaptive LDPC decoder. The performance measure is *throughput* (the proportion of transmitted data bits that are successfully delivered to the higher layers).

4.4.1 Multiple Decoders & Error Correcting Capability Factors

In the communication model in Section 4.1, it is assumed that there are $s, s > 0$ decoders operating at the receiver buffer. In addition, the error correction capability of each decoder is measured by the parameter α. For instance, for perfect codes $\alpha = 0.5$, since they can correct half as many errors as there are parity symbols in a message. Intuitively, we should get a better performance with multiple decoders in place where each decoder is using an error-correcting code with capability close to perfect codes. That is, we expect that the overall throughput monotonically increases with respect to s and α. Fig. 4.5 shows the PEEC throughput for the channel trace with PHY rate of 11Mbps and transmission rate 1024kbps. This figure illustrates the change in throughput with respect to the variations of s and α. Interestingly, we observe that the throughput slightly increases

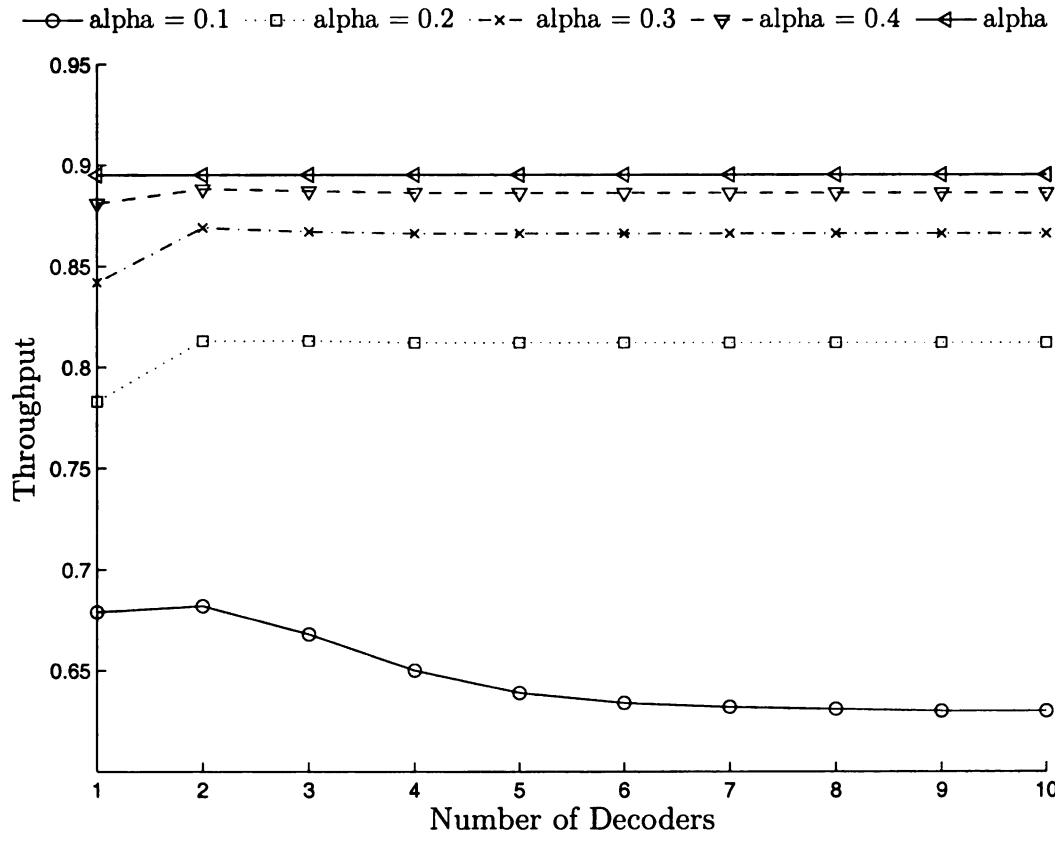


Figure 4.5: Average PEEC throughput for different α values with respect to the number of decoders (s)

when two decoders are used, but it remains unvaried as the number of decoders increases. This result suggests that the service time for the buffer does not improve dramatically as the number of decoders increases.

To investigate this, using equation (4.7), we compute the average service time measured by the number of transmission intervals for this channel trace. Fig. 4.6 illustrates that the reduction in average service time is relatively slower for large number of decoders. Moreover, the average service time is longer when decoders are using codes with lower α. Further, when there are multiple decoders in place, the sender is obliged to transmit parity symbols requested by each decoder. As a result, the average amount of parity symbols embedded in each message will increase with respect to the number of decoders; because the service time is not significantly improving, it is possible that using multiple decoders degrades the overall throughput. This phenomenon is observed in

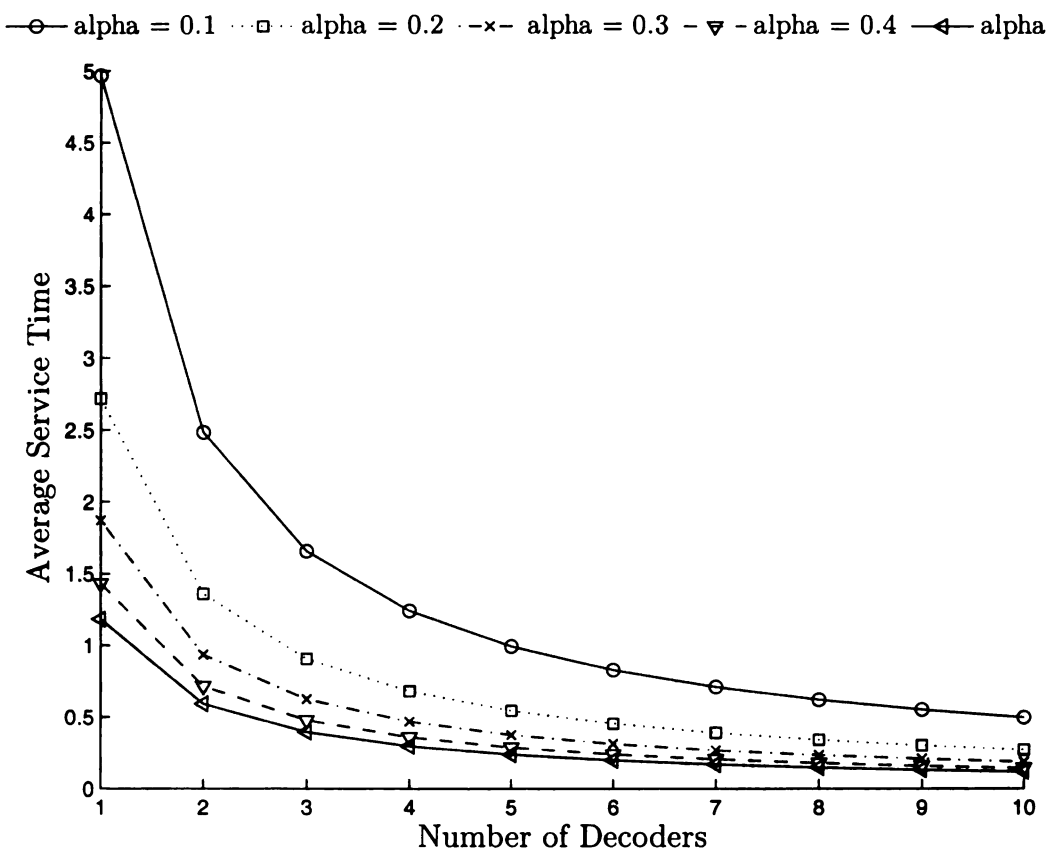


Figure 4.6: Average service time (measured by the number of transmission intervals) for different α values with respect to the number of decoders (s).

Fig. 4.5 for $\alpha = 0.1$. Therefore, using multiple decoders is not only ineffective in improving the throughput but also may reduce the overall performance.

It is also observed in Fig. 4.5 that the performance increases significantly when a decoder is utilizing a code with decent error-correcting capability. So, we conclude that the only parameter that plays a critical rule in improving the performance is α. This result is desirable in practice since using more than one decoder in every receiver in the network is expensive and impractical. This analysis shows that in order to achieve a better throughput, one should employ a decoder with error-correcting capability close to that of perfect codes rather than multiple decoders.

Table 4.1: The throughput comparison of enhanced ARQ schemes.

Trace	IEEE Standard ARQ	MA-3	MA-5	GMA-4	GMA-5
1024kbps	0.2217	0.2904	0.3053	0.3236	0.3261
750kbps	0.8412	0.8434	0.8434	0.8437	0.8437
900kbps	0.7554	0.7768	0.7795	0.7812	0.7816
500kbps	0.7354	0.7551	0.7562	0.7585	0.7586

4.4.2 Comparison with Contemporary Protocols

Various error-combating schemes have been adopted in both physical layer and link layer for wireless systems. The current IEEE 802.11 MAC standard uses a conventional ARQ technique [29]: a single erroneous bit will result in the retransmission of the whole packet. To enhance the ARQ performance, other techniques such as xor combining (XOR) [33] [34], majority combining algorithms (MA) [35] [36] and generalized majority combining algorithms [37] (GMA) have been developed. Although these schemes are not standard protocol, but they have been used in various applications such as multimedia. These methods share a common premise and that is to store the corrupted packets in the receiver and combine them to restructure the original packet. A majority combing scheme on the other hand uses the last k received copies of a packet. In this technique, a bit-by-bit majority decision is performed to reconstruct a packet. That is, for MA-3 (i.e., $M = 3$), a bit is one in particular location in a reconstructed packet if at least two copies of the packet contain ones in that location. In the generalized majority combining scheme, all the possible combinations of M stored packets are considered. The reader can refer to Chapter 2 for more detailed description of the functionality of these schemes.

For the set of experiment evaluations presented here, we consider the maximum transmission limit (MTL) to be six. That is, for every packet there is at most five retransmissions allowed. The copies of a particular packet will be dropped if the original packet cannot be reconstructed in this period. Correspondingly, in our communication scheme, a particular packet is dropped from

the receiver buffer if its delay exceeds the MTL. We consider IEEE standard ARQ, MA-3, MA-5, GMA-4 and GMA-5. The IEEE standard ARQ requests for a retransmission if the received packet is corrupted and if the number of requested retransmissions is below the MTL. The MA-3 performs the majority combining algorithm if the retransmissions of the second and third copies have failed. The MA-3 algorithm always performs majority decision on the last three received copies. The MA-5 is employed if the retransmission and MA-3 schemes have failed to recover the corrupted packet. Similarly, the GMA-5 is applied if MA-5, GMA-4, MA-3 and retransmission schemes have failed. So, we expect that the GMA-5 produces a better performance than the other schemes. Table 4.1 compares the the throughput of these protocols over four channel traces collected at sniffer one. We observe that the MA schemes have a slightly better performance than the IEEE standard ARQ; however the increase in the performance is not significant because the majority combining algorithms are repetition code which is a 1-error correcting code. Furthermore, it is well known that the codes with large length have a better error-correcting capability. Hence, these schemes cannot achieve significant improvement in the performance since they are utilizing repetition codes with length three and five.

As mentioned in Chapter 2, the concept of Hybrid ARQ (HARQ) schemes have been developed in information theory and coding theory which employs various codes including Reed Solomon and LDPC for error correction [48]- [53]. To compare the performance of PEEC with HARQ-I and HARQ-II, we introduce parameter Δ. In particular, Δ specifies the proportion of the redundancy information in the packet. For instance, for the packet with size n, in the HARQ-I, the sender transmits $\frac{n}{\Delta}$ redundancy and $\frac{n(\Delta-1)}{\Delta}$ data symbols. Similarly, $\frac{n}{\Delta}$ specifies the amount of additional redundancy that is allocated by HARQ-II for a particular packet. As illustrated in Fig. 4.7, the value of Δ has a direct impact on the overall throughput of the HARQ schemes. Using a small Δ will result in the transmission of few redundancy bits which provide insufficient information for distortion recovery and so FEC becomes ineffective in improving the performance. On the hand,

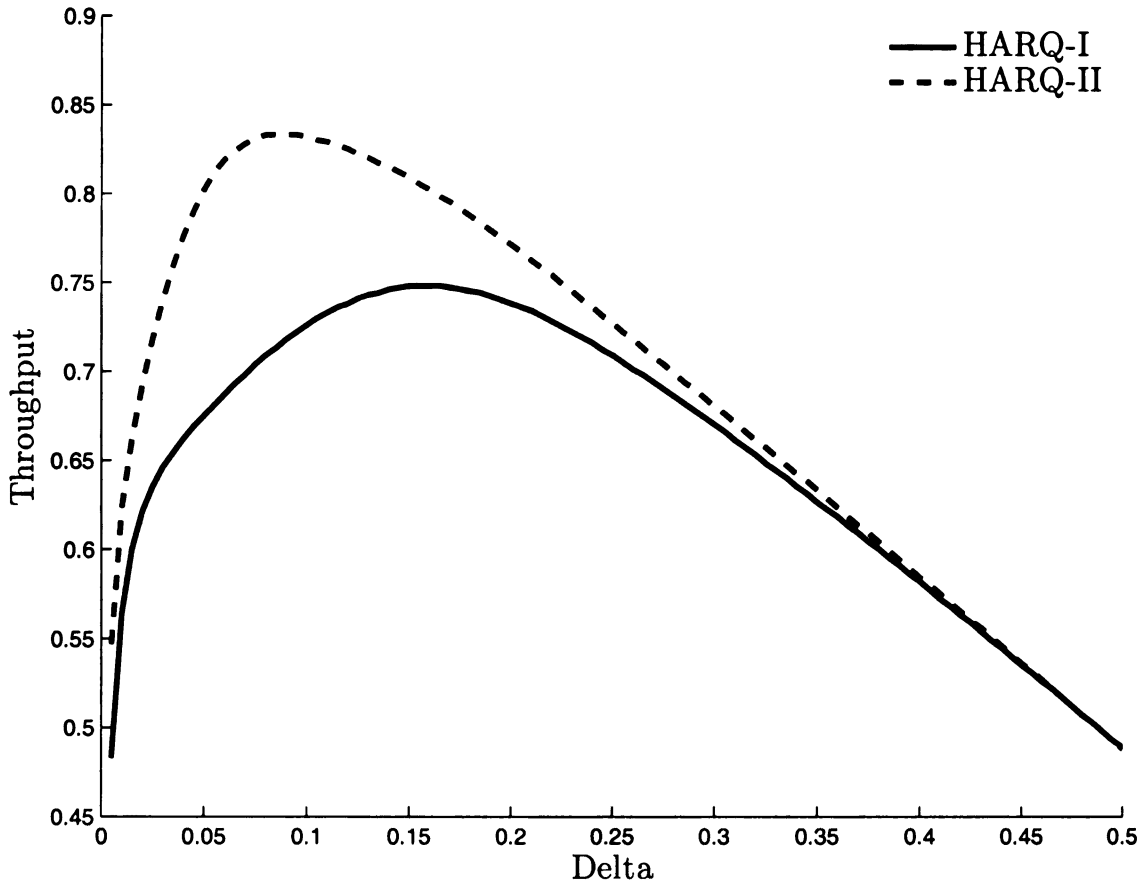
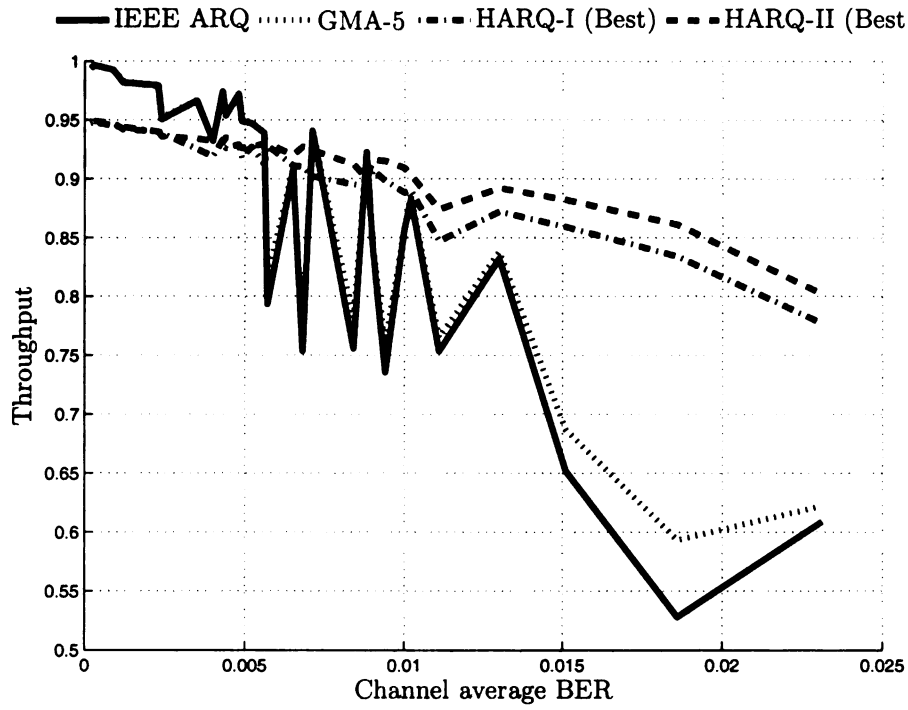


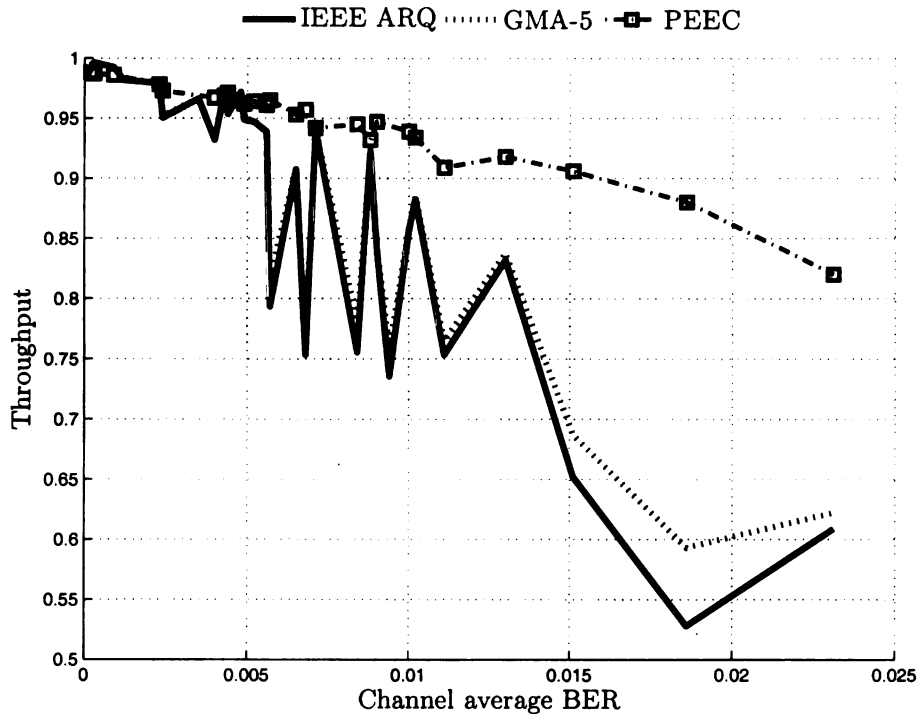
Figure 4.7: The Impact of Δ value in throughput of HARQ schemes for a channel with PHY rate 11Mbps and transmission rate 1024kbps.

using a large value of Δ increases the likelihood of successful decoding at the receiver while decreases the amount of data symbols that are transmitted in every packet; therefore the overall throughput will decay. We refer to the maximum achievable throughput of HARQ schemes with respect to Δ as the *best* performance of HARQ schemes.

We apply the IEEE standard ARQ, majority combining methods and HARQ techniques as well as the PEEC on all channel traces. For each trace, we evaluate the throughput of the IEEE standard ARQ and GMA-5 and we also find the best performances of the HARQ schemes with respect to Δ. For these experiments, the MTL is six; for HARQ schemes and PEEC $\alpha = 0.3$, the buffer size is 10 (i.e., $m = 10$) and there is only one decoder operates at the receiver (i.e., $s = 1$). Fig. 4.8 illustrates these performances with respect the average BERs of our channel traces. Specifically,

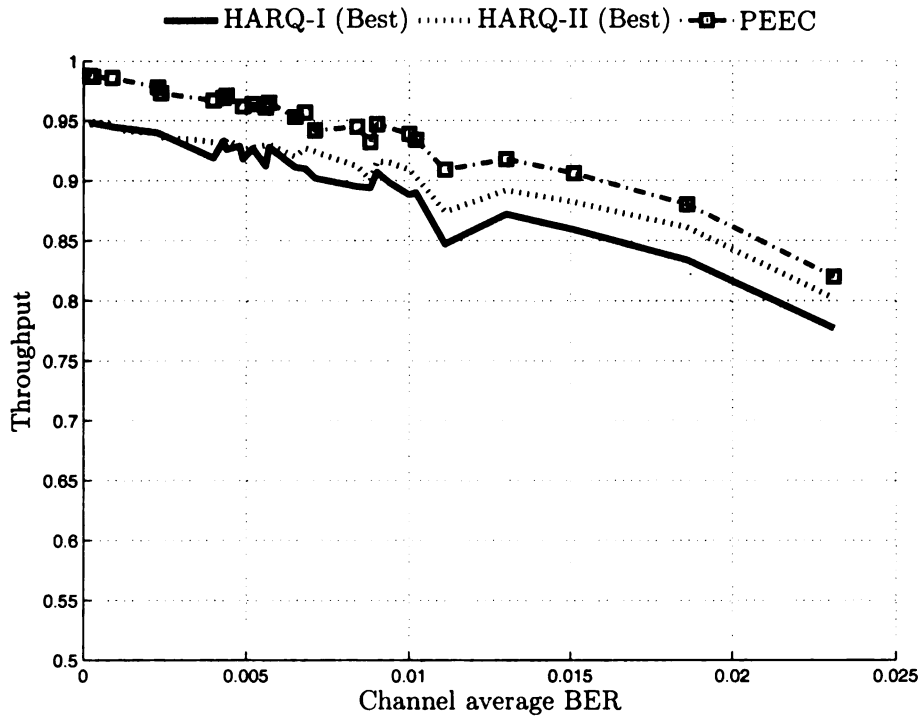


(a)

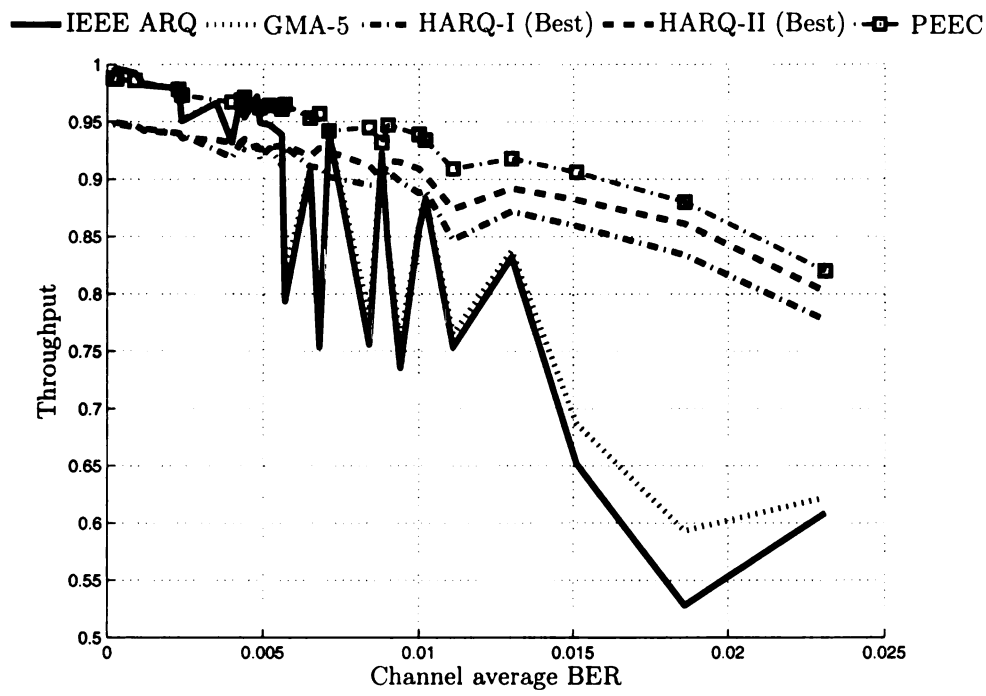


(b)

Figure 4.8: The throughput of different error control protocols over the various channel traces.



(c)



(d)

Figure 4.8 continued.

Figure 4.8a compares the performances of the IEEE standard ARQ and GMA-5 with those of the HARQ schemes. In general, we observe that the HARQ-II outperforms HARQ-I for all traces. In addition, as the previous experiments suggested, the GMA-5 has a slightly better performance than IEEE ARQ especially when BER is large. However, we observe that IEEE ARQ performs better than HARQ schemes when the BER is small (i.e., below 0.005) because these schemes transmit a fixed amount of redundancy regardless of the channel conditions. As a result, for channels with low BERs these schemes suffer overcompensation. As the BER increases, the performance of the IEEE ARQ decreases but at the same time the performance of HARQ schemes increases. Therefore, they appear close to each other in Fig. 4.8a when BER is ranging from 0.005 to 0.01. For BER larger than 0.01, the HARQ schemes outperform the IEEE ARQ scheme. Notice that for some traces with the average BER in this range $(0.005, 0.01)$, a sudden fluctuations in the performances of the IEEE ARQ and GMA-5 is detected; this is discussed in the sequel.

Fig. 4.8b shows PEEC performance with that of IEEE802.11 ARQ and GMA-5. Similar to the HARQ schemes, PEEC outperforms the IEEE802.11 ARQ scheme for channels with relatively high average BERs, but unlike the HARQ techniques for small BERs, PEEC performs very close to IEEE802.11 ARQ. This result shows that the adaptive redundancy allocation in the PEEC protocol can successfully prevent overcompensation for the channels with low BERs. In Fig. 4.8c, the performance of PEEC is compared with the performances of the HARQ schemes for all trace channels. It shows that the PEEC protocol has even a better throughput than the *best* throughput of the HARQ schemes regardless of the channel conditions. The performances of all schemes are illustrated in Fig. 4.8d.

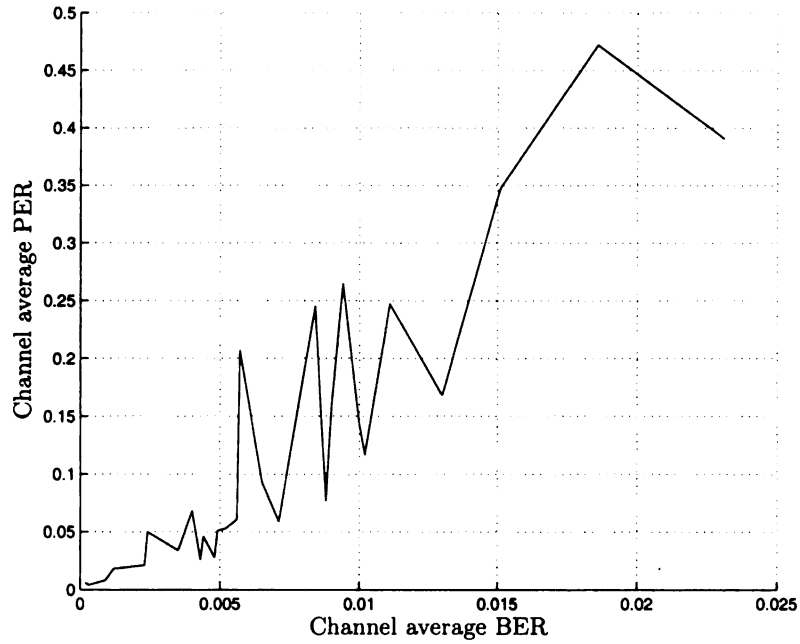
In our performance analysis, we observe that the throughput of the IEEE ARQ is fluctuating for the channels with BERs ranging from 0.005 to 0.01. Recall that the IEEE ARQ uses the retransmission mechanism instead of forward error correction. Therefore, the overall throughput of this scheme depends on the average number of receptions of distorted packets rather than the

average number of distorted bits in a packet. That is, the throughput of the IEEE ARQ scheme depends on the packet error rate of the channel (PER) rather than its bit error rate (BER). So, the IEEE throughput fluctuations appear when the PER fluctuates as the BER increase.

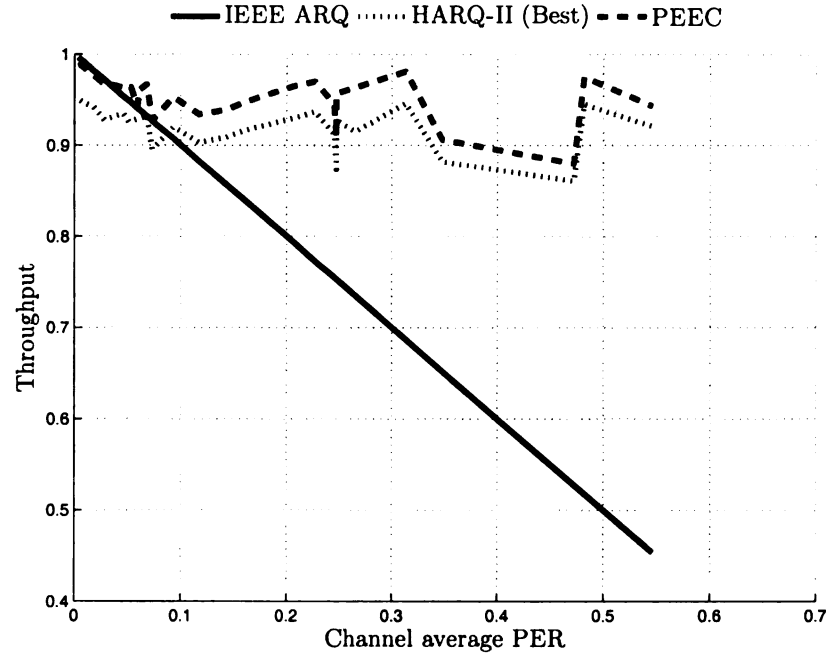
Figure 4.9a depicts the average PERs of our channel traces with respect to their average BERs. We observe that in general for large (small) BERs, the PER is large (small). However, although theoretically we expect that the PER monotonically increases with respect to BER; for real traces this growth in not monotonic. Figure 4.9a shows noticeable fluctuations of the PER values when BER is in the range of $(0.005, 0.01)$ which explains the sudden changes that appear in the IEEE ARQ throughput in this range. Figure 4.9b shows the throughput of IEEE ARQ, HARQ-II and the PEEC protocols with respect to the average PER value of each channel trace. As expected, the IEEE ARQ performance decreases as the PER increases; however, since FEC is utilized by HARQ and PEEC schemes, the variations in the PER values have a little impact on the overall performances of these schemes.

For the experimental results illustrated in Fig. 4.8, it is assumed that the error-correcting capability of a code is $\alpha = 0.3$. To investigate whether the performance analysis given for these results is always valid and they are not limited to a particular code, we evaluate the throughput of the PEEC protocol for α values ranging from 0.1 to 0.5 over various channel traces.

In Fig. 4.10a, we observe that even for $\alpha = 0.1$ the PEEC outperforms the IEEE standard ARQ for the channel traces with large BER and at the same time, it performs close to IEEE standard ARQ when the BER is less than 0.005. Notice that a code with an average error-correcting capability of $\alpha = 0.1$ requires 10 parity bits to correct a single error which is a code with a very low complexity. So, this result suggests that by using a very simple decoder, one can achieve more than 10% improvement in the overall throughput with respect to the IEEE standard ARQ. On the other hand, Fig. 4.10 shows that the throughput is above 90% even for most of the channel traces when a decoder uses a perfect code. Furthermore, it is shown in Fig. 4.10b that the performance



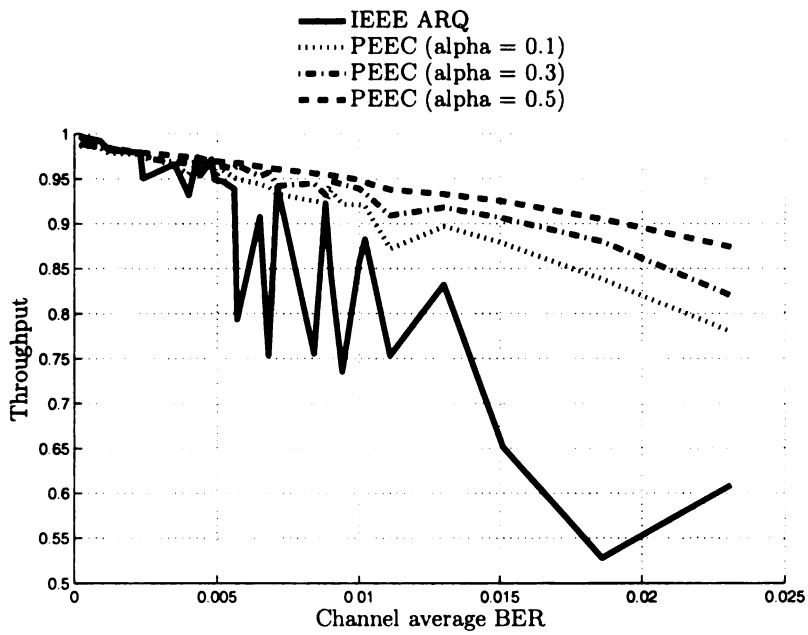
(a) The average PER of the channel traces with respect to their average BER values



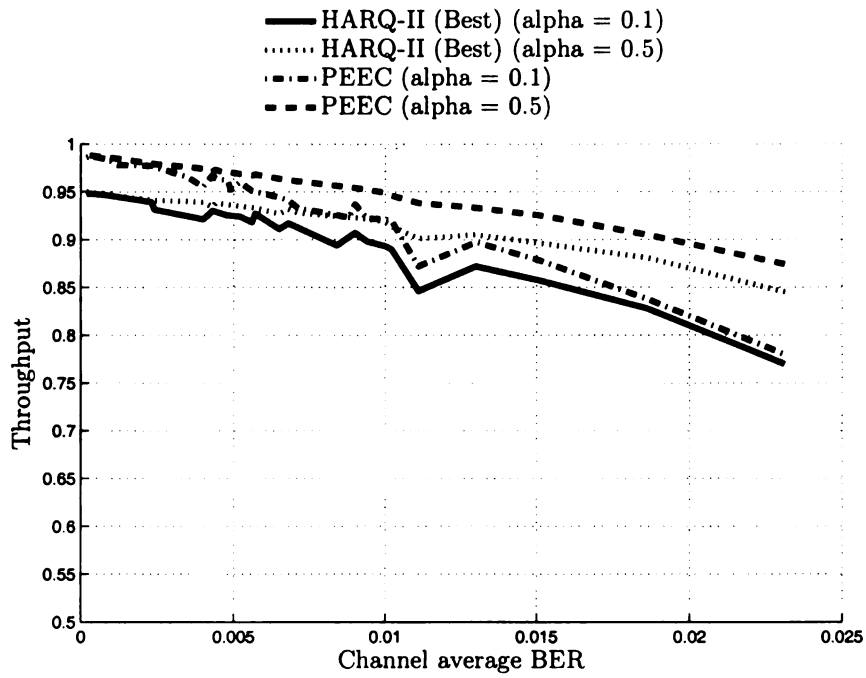
(b) The throughput of the ARQ schemes with respect to the PER variations

Figure 4.9 The throughput of error control schemes with respect to variation in channel PER.

of the PEEC is above the best performance of the HARQ-II scheme regardless of the value of α.



(a)



(b)

Figure 4.10 The change in the PEEC throughput with respect to α over the various channel traces.

4.4.3 Implementation of PEEC using A-LDPC

For the experimental results illustrated in Fig. 4.8, it is assumed that the error-correcting capability of a code is $\alpha = 0.3$. To investigate whether the performance analysis given for these results is

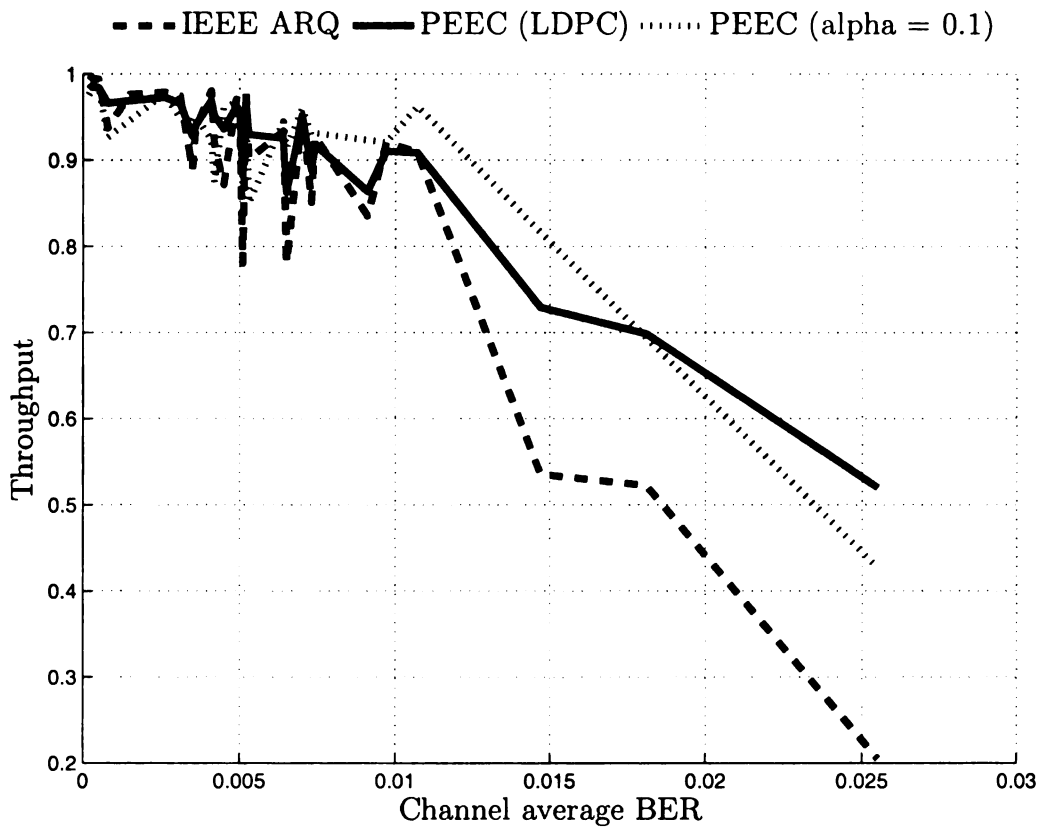


Figure 4.11 The throughput of the PEEC protocol using the A-LDPC decoder over various channel traces.

always valid, we evaluate the throughput of the PEEC protocol using a relatively simple Adaptive Low-Density-Parity-Check code (A-LDPC). The experimental setup is as follow: we let a sender always transmits a zero codeword. We use our channel traces to distort the transmitted codeword by flipping distorted bits. The receiver uses A-LDPC to decode the received word and checks whether the decoded word is a zero codeword. Upon decoding failure, a receiver stores the received word in its buffer and requests for additional redundancy. We use the LDPC source code provided in [104] for our experiment. Note that we use a soft decision decoding using an iterative belief propagation method which requires a knowledge of channel BER. So, we use our estimate for channel BER for every transmission interval described in Section 4.3 to decode each packet. For this experiment, the maximum iteration is 100; the variable side has degree three and the check side degree is approximately regular.

Fig. 4.11 shows the overall performance could improve at a minimum of 25% with respect to the performance of the IEEE802.11 standard ARQ scheme over channels with higher BER. Also, PEEC performs close to IEEE standard ARQ when the BER is less than 0.005. To estimate the average error-correcting capability of this decoder, we consider all the packets that are successfully decoded; for each packet, we compute the fraction of the number of errors by the total amount of parity symbols embedded in that packet. We find that the estimated error-correcting capability of this decoder is around $\hat{\alpha} = 0.1$. In Fig. 4.11, we also show the throughput of PEEC when a decoder is utilizing a code with error-correcting capability $\alpha = 0.1$. We observe that the performance of the model captures the throughput obtained by employing A-LDPC. This result verifies the accuracy of our analysis and suggests that by using a code with error-correcting capability close to perfect codes, we can achieve higher throughput.

4.5 Channel Coding Rate Analysis

An important objective of error-correcting schemes is to increase the likelihood of successful recovery of the original packet by utilizing a particular decoding procedure to recover a corrupted packet. For instance, IEEE802.11ARQ uses retransmissions as its decoding procedure while HARQ schemes and PEEC use Forward Error Correction (FEC) codes. Each error-correcting method introduces a channel coding rate measured by

$$R = \frac{\text{\# of data symbols}}{\text{\# transmitted symbols}} \tag{4.10}$$

which measures the rate of *new* data transmitted over the channel.

According to the channel coding theorem, a particular decoding scheme has to operate below the capacity to guarantee reliable transmission. That is

$$R < C$$

where C is the channel capacity. Ideally, a perfect error-combating scheme should operate with a channel coding rate very close to the capacity while achieving a likelihood of decoding failure close to zero. However, in practice error-correcting codes have to operate considerably below capacity to achieve zero decoding failure probability. In this section, we derive the likelihood of successful decoding and the channel coding rate of the contemporary error-correcting methods and PEEC.

4.5.1 IEEE802.11 ARQ

IEEE802.11 ARQ mechanism requests a retransmission for the copy of the corrupted received packet. This procedure is repeated until an uncorrupted copy is received or the Maximum Transmission Limit (MTL) is reached. To find the probability of decoding failure of IEEE802.11 ARQ, we let d represent the tolerable delay of every transmitted packet at the receiver side. According to the IEEE802.11 ARQ mechanism, this delay should be at most MTL. Let ς_i represent the likelihood that the transmitted packet is erroneous when the channel is in state S_i. Thus,

$$\varsigma_i = 1 - (1 - \epsilon_i)^n. \tag{4.11}$$

Under IEEE802.11 ARQ, a particular packet is successfully delivered if it is errorless in the first transmission or in any of consecutive $d - 1$ retransmissions. Therefore, the process of recovering the original packet under IEEE802.11 ARQ has the geometric distribution with parameter ς_i. Consequently, the likelihood of successful decoding under IEEE802.11 ARQ after d transmissions when the channel is in state S_i is,

$$P_{ieee}(i, d) = \sum_{t=1}^{d} \varsigma_i^{t-1}(1 - \varsigma_i). \tag{4.12}$$

The channel is in state S_i with the steady state probability of π_i, correspondingly, the probability

of successful decoding (on average) is:

$$P_{ieee}(d) = \sum_{i=1}^{N} \pi_i P_{ieee}(i,d). \tag{4.13}$$

Under IEEE802.11 ARQ, a sender transmits a *new* packet in the first transmission and performs $d-1$ retransmissions until an errorless packet is received. Since each packet has length n, the channel coding rate is

$$R_1(d) = \frac{n}{nd} = \frac{1}{d}. \tag{4.14}$$

4.5.2 Enhanced ARQ

Under Enhanced ARQ mechanisms such as majority combining (MA-k) and generalized majority combining (GMA-k) methods, a majority combination of the original packet and its corrupted copies is performed on the top of the retransmission mechanism when the original packet is not retrieved after $k-1$ retransmissions. In the following sections, we compute the probability of successful decoding (recovering the original packet) of each of these schemes. In general, this probability is governed by the number of packets that are combined k and the maximum retransmission limit d.

Majority Combining with k Copies

In the case of MA-k, bit-by-bit comparisons of k copies of a packet are performed, if a retransmission mechanism fails to recover the original packet after $k-1$ transmissions. Assuming the channel is in state S_i, equation 4.12 suggests that a retransmission mechanism fails to recover a packet after $k-1$ transmissions with probability of

$$1 - \sum_{t=1}^{k-1} \varsigma_i^{t-1}(1-\varsigma_i)^t. \tag{4.15}$$

In $i^{th}, i \geq k$ transmission, a MA-k method successfully retrieves a packet when an errorless copy is received or a majority combining method succeeds. Having a channel in state S_i, the

probability of receiving an error-free packet is $(1-\varsigma_i)$. Further, under MA-k, a successful recovery is achieved if any $m = \lceil \frac{k}{2} \rceil$ combinations of k copies are identical. Let random variable Z represent the number of bits of a particular location in k packets that have identical values. Since each bit can be distorted with probability of $1 - \epsilon_i$, Z has the binomial probability mass function with parameters $(1 - \epsilon_i, k)$. Therefore, the probability that the majority combining algorithm successfully decodes a bit (at least m bits have identical value) can be computed as follows:

$$P_i = \sum_{x=m}^{k} \binom{k}{x} (1 - \epsilon_i)^x \epsilon_i^{(k-x)}. \tag{4.16}$$

Each packet has the length of n bits, therefore using the majority combining algorithm, all n bits are decoded successfully with the probability of P_i^n when the channel is in state S_i. Consequently, the likelihood of successful delivery in $i^{th}, i \geq k$ retransmission is

$$g_i = (1 - \varsigma_i) + \varsigma_j P_i. \tag{4.17}$$

In general, the enhanced ARQ mechanism which uses MA-k recovers the original packet after $d, d \geq k$ transmissions if (1) the ARQ mechanism succeeds in the first $(k-1)$ transmissions and (2) the majority combining method succeeds for the rest of the possible transmissions. Therefore, assuming the channel is in state S_i, using equations (4.15) and (4.17), the probability of successful decoding under MA-k method is

$$P_{ma}(i, k, d) = P_{ieee}(i, k-1) + (1 - P_{ieee}(i, k-1))\Lambda(i, k, d) \tag{4.18}$$

where

$$\Lambda(i, k, d) = \sum_{t=k}^{d} (1 - g_i)^{t-k} g_j. \tag{4.19}$$

The channel is in state S_i with steady state probability of π_i, therefore the probability of successful decoding enhanced ARQ method using MA-k (on average) is

$$P_{ma}(k, d) = \sum_{i=1}^{N} \pi_i P_{ma}(i, k, d). \tag{4.20}$$

Generalized Majority Combining with k Copies

In case of generalized majority combining scheme (GMA-k), in $i^{th}, i > k$ retransmission, a GMA-k successfully recovers a packet when an error-free copy of a packet is received or any combination of MA-l ($l < k$) succeeds. For instance GMA-5 should apply all cases of the MA-3 method on 5 copies so for $k = 5$, $l = 3$. Equation (4.16) suggests that each MA-l operation successfully recovers the original packet with the probability of P_i when the channel is in state S_i. We let random variable W represent the number of MA-l cases that succeed. Having k packets at the receiver, the number of possible MA-l operations is $L = \binom{k}{l}$. Therefore, W has the binomial distribution with parameters (P_i, L). Consequently, the likelihood that GMA-k algorithm successful recovers the original packet (at least one of the L MA-l operations succeed) when the channel is in state S_i can be computed as follows,

$$P_W(i,k) = \sum_{x=1}^{L} \binom{L}{x} P_i^x (1-P_i)^{L-x}. \tag{4.21}$$

Therefore, the probability of successful recovery in i^{th} retransmission is

$$w_i(k) = (1-\varsigma_i) + \varsigma_j P_W(i,k). \tag{4.22}$$

In general, enhanced ARQ mechanism which employs GMA-k successfully recovers the original packet when (1) retransmission mechanism succeeds in the first $l-1$ transmissions, (2) MA-l method succeeds in $j, l \leq j \leq k-1$ transmissions and (3) GMA-k succeeds in $j, k \leq j \leq d$ transmissions, to recover a packet. Therefore the probability of successful decoding when the channel is in state S_i is computed as follows

$$\begin{aligned} P_{gma}(i,l,k,d) &= P_{ieee}(i,l-1) \\ &+ (1-P_{ieee}(i,l-1))\Lambda(i,l,k-1) \\ &+ (1-\Lambda(i,l,k-1))\Gamma(i,k,d) \end{aligned} \tag{4.23}$$

where

$$\Gamma(i,k,d) = \sum_{t=k}^{d} (1 - w_i(k))^{t-k} w_i(k) \tag{4.24}$$

measures the successful probability of GMA algorithm. Accordingly, the probability of successful decoding of enhanced ARQ mechanism using GMA-k (on average) is

$$P_{gma}(k,d) = \sum_{i=1}^{N} \pi_i P_{gma}(i,k,d). \tag{4.25}$$

$MA-k$ and $GMA-k$ have the same channel coding rate as IEEE802.11 in Equation (4.14).

4.5.3 Hybrid ARQ (HARQ)

The successful decoding probabilities computed so far are for decoding methods based on the concept of pure retransmission. However, the Hybrid ARQ (HARQ) schemes utilize FEC codes for error recovery. Let α represent an expected error-correcting capability of a particular decoder. Specifically, if x represents the amount of parity bits embedded in a particular packet, then an α-decoder can correct up to $\alpha \times x$ corrupted bits in that packet. To calculate the likelihood of successful recovery for HARQ-I method, we let random variable D_i represent the distortion of a transmitted packet when a channel is in state S_i. Since in state S_i the channel is a BSC with crossover BER ϵ_i, then D_i has a binomial distribution with parameters n and ϵ_i.

The distortion level D_i is random and unknown to the receiver, the notion of partial recovery is unrealistic in error correction. That is, the receiver can either correct all errors in a packet and acknowledges successful decoding or can just assume that no recovery is achieved. So the level of distortion in a packet *after decoding*, denoted by U_i, is

$$U_i = \begin{cases} 0 & D_i \leq \alpha x \\ D_i & \text{otherwise} \end{cases} \tag{4.26}$$

Equation (4.26) shows that the distribution of U_i is equivalent to the distribution of D_i *truncated* on αx. So, the overall probability of successful decoding (SD) is

$$SD(i,x) = P(D_i \leq \alpha x) = \binom{n}{\alpha x} (1-\epsilon_i)^{\alpha x} \epsilon_i^{n-\alpha x}. \tag{4.27}$$

In the j^{th} retransmission, a successful delivery occurs if a retransmission succeeds or a packet is decoded. So the probability of successful decoding in j^{th} retransmission $g_4(j)$ is

$$g_4(j) = (1 - \varsigma_i) + \varsigma_i SD(j, x_j). \tag{4.28}$$

Therefore the successful decoding likelihood in packet recovery under HARQ-I after d retransmissions is

$$f_4(d) = \sum_{i=1}^{d} (1 - g_4(i))^{i-1}(1 - g_4(i)). \tag{4.29}$$

The channel coding rate for HARQ-I depends on the amount of redundancy carried with a packet in the first transmission x_1,

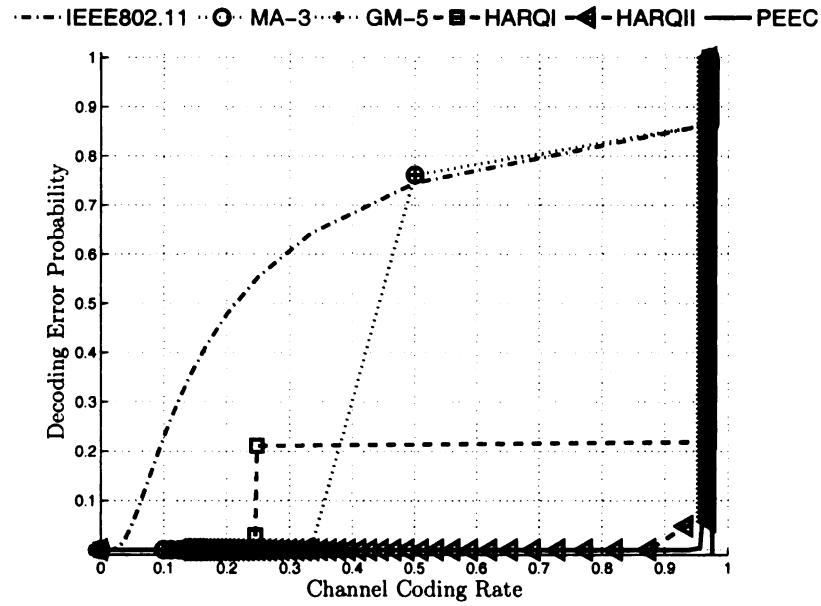
$$R_4(x_1, d) = \frac{n - x_1}{nd} \tag{4.30}$$

The derivations of the failure decoding probabilities of HARQ-II and PEEC are similar to HARQ-I. Because the later protocols utilize additional parity bits, the failure likelihood decreases dramatically as additional parity bits are employed. The channel coding rate for HARQ-II is

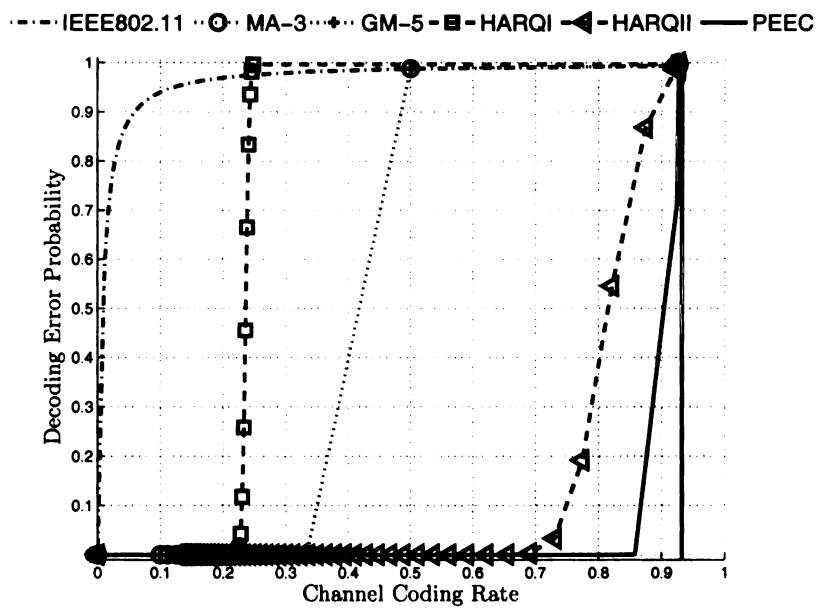
$$R_5(x_1, \mathbf{Y}, d) = \frac{n - x_1 - \left(\sum_{i=2}^{d} y_i\right)}{n + \left(\sum_{i=2}^{d} y_i\right)} \tag{4.31}$$

where $\mathbf{Y} = (y_2, \cdots, y_d)$ represents the amount of additional parity bits.

Using the trained channel Markov model of a particular channel trace, we compute the probability of decoding failure of each of the error-correcting scheme based on different channel coding rates. Fig. 4.12 shows this performance over four different Markov channels, with the average probabilities of 0.0025, 0.009, 0.018 and 0.03. We observe that the IEEE802.11 ARQ scheme has a relatively poor performance because regardless of channel condition, the probability of decoding failure is very sensitive to the increase of the channel coding rate. MA-3 and GMA-5 have a relatively better performance than IEEE802.11 ARQ and even HARQ-I, but they have to operate way below channel capacity to achieve zero decoding failure. As it is illustrated in Fig. 4.12,



(a) Average BER: 0.0025

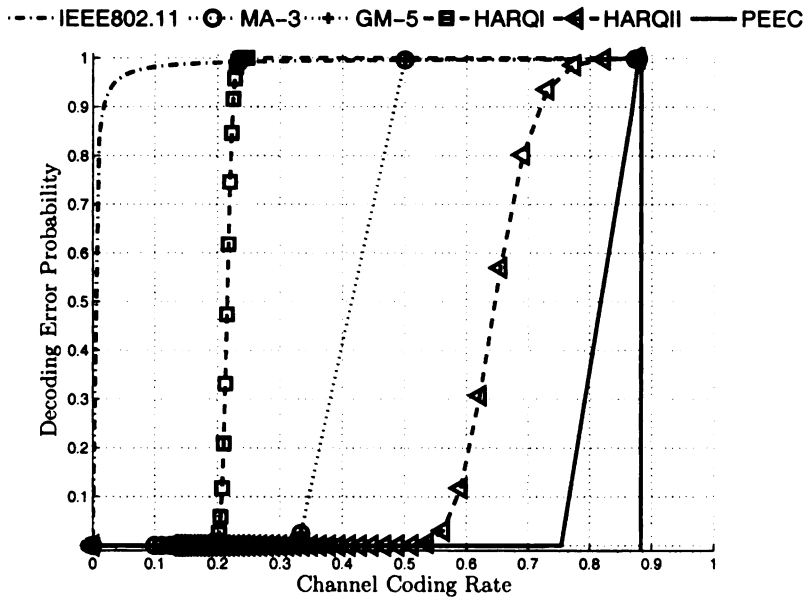


(b) Average BER: 0.0090

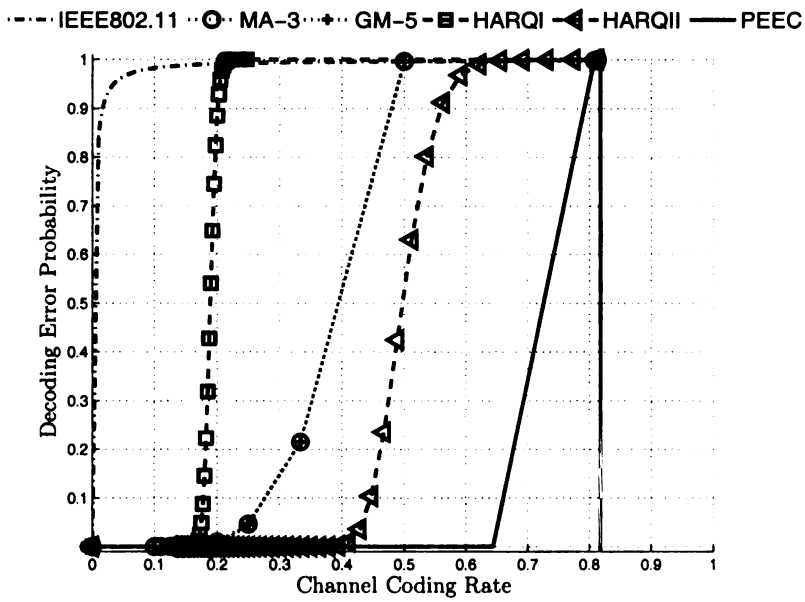
Figure 4.12 The variation of decoding failure with respect to variation channel coding rate over different Markovian Channels. Note that the vertical line represents the channel capacity of a given channel.

HARQ-II and PEEC performance are superior to other methods. Meanwhile, as we observe PEEC outperform HARQ-II as the channel conditions worsen.

Fig. 4.13 illustrates an overall performance of all schemes over various channels. Specifically,



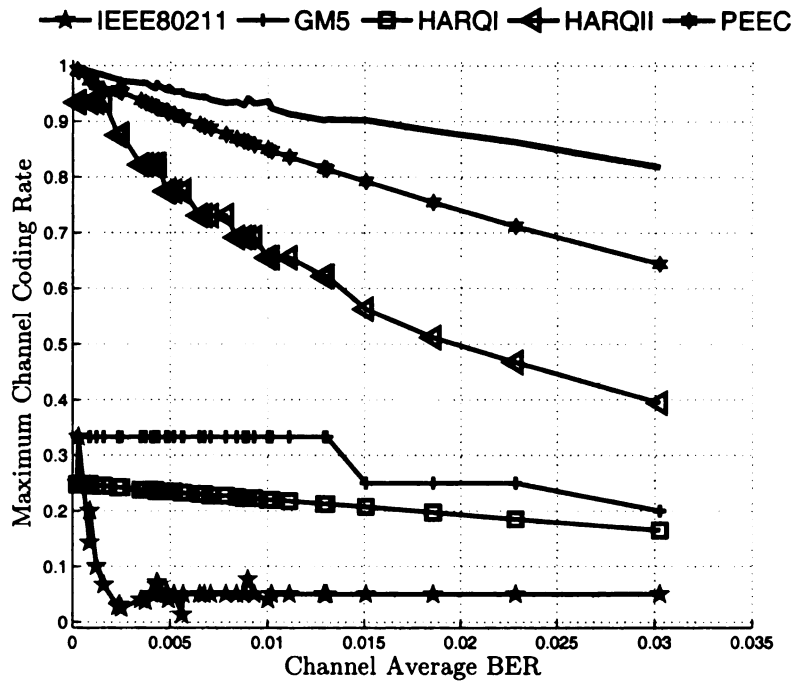
(c) Average BER: 0.0180



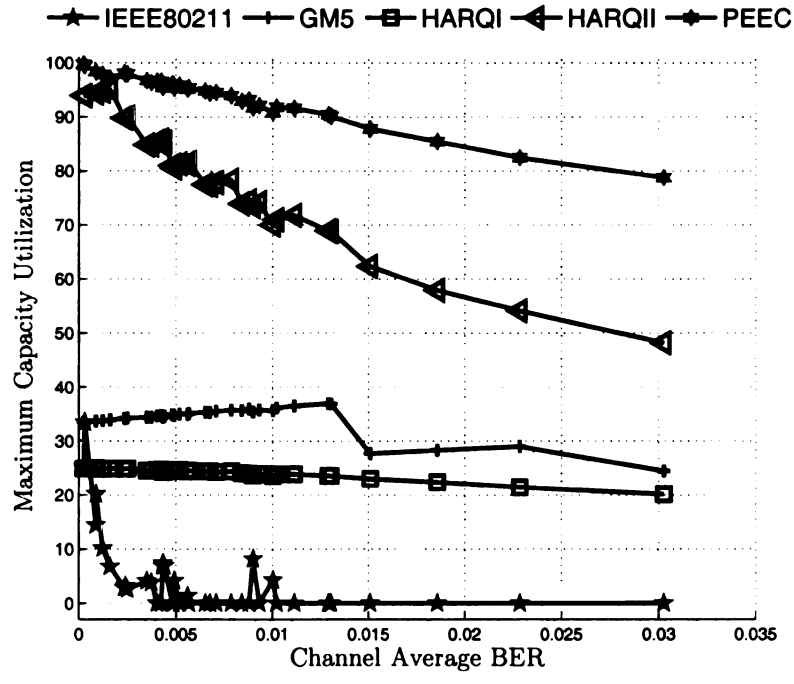
(d) Average BER: 0.0300

Figure 4.12 continued.

Fig. 4.13a shows the maximum channel coding rate of each scheme with respect to the channel average BER. The maximum channel coding rate measures the highest achievable rate such that the likelihood of decoding error is zero. The upper bound capacity of each channel is also depicted in Fig. 4.13a. It is clear that PEEC and HARQ-II achieve higher rates regardless of the channel



(a) Maximum Channel Coding Rate



(b) Channel Capacity Utilization

Figure 4.13 The maximum channel coding rate and percentage of channel utilization over different Markovian Channels. Note that the solid line in (a) represents the upper bound of the channel capacity.

condition. In Fig. 4.13b, we observe that PEEC utilizes more than 90% of channel capacity over channels with average BER less than 0.015. On the other hand, over the channels with BER more than 0.02 where all other schemes can only utilize less than 50% of the capacity, PEEC operates above 80% of the capacity.

4.6 Discussion

In this chapter, we studied the problem of reliable transmission over wireless links. In particular, a contention free wireless communication model was analyzed where a receiver stores corrupted packets for future error recovery. We developed suitable models for each component of this communication scheme to introduce an error-combating scheme at the link layer, which we refer to as Packet Embedded Error Control (PEEC) protocol. In addition to theoretically analyzing PEEC, the performance of the proposed scheme was extensively analyzed over real channel traces collected on 802.11b WLANs. We compared PEEC performance (as measured by throughput) with the performance of (a) the IEEE802.11 standard ARQ protocol, (b) enhanced ARQ schemes such as majority combining mechanisms and (c) the well known hybrid ARQ/FEC (HARQ) schemes. Our analysis and experimental simulations showed that PEEC outperforms all three competing protocols over a wide range of actual channel traces. Finally, the implementation of PEEC using an Adaptive Low-Density-Parity-Check (A-LDPC) decoder was presented. In the next chapter, we extend the PEEC protocol to provide both reliability and flow control for realtime video traffic.

CHAPTER 5

DC-PEEC: Delay Constraint Error Control Protocol For Realtime Video Communication

Realtime multimedia applications require a communication mechanism that provides reliable information delivery while ensuring (1) effective bandwidth utilization to deliver high quality video contents, and (2) low-latency to satisfy realtime delay constraint. These requirements make it essential to design an error control protocol that provides flexible flow control mechanism (with respect to cost, quality, and latency) in conjunction with reliability in data delivery. Such requirements are even more critical under wireless environments since wireless channels are subject to information loss due to the error-prone nature of wireless links and their susceptibility to a variety of distortions caused by fading, interference, and mobility.

The de facto IEEE802.11 link-layer error-control protocol [29] uses retransmissions to provide reliability for wireless environments. Although this technique ensures reliability "over-a-long-run", it does not provide flow control mechanism with respect to latency. This design strategy has led to a great deal of inefficiency in throughput which introduces frequent information loss in

the system. The impact of such information loss could be dramatic on multimedia applications because any damage to the compressed video stream may lead to undesirable visual distortions at the decoder [94]. Inefficient channel bandwidth utilization is quite conspicuous in realtime video streaming since it often results in frame freezing, skipping and/or playback jitter leading to an unsatisfactory user experience.

Recent works in wireless multimedia communications have proposed cross-layer mechanisms to overcome performance limitations imposed by conventional protocols [94] [95]. Examples of these protocols are the Hybrid Erasure-Error Protocols (HEEPs), well established hybrid ARQ (HARQ) schemes which make use of incremental channel codes to achieve reliable transmission over wireless channels [43–45]. Although, these approaches can achieve reliability with fewer packet retransmissions, but these and other conventional 802.11 ARQ based link-layer schemes do not address issues raised by various realtime video demands such as quality and delay constraints.

In this chapter, we built on the PEEC protocol described in the previous chapter and introduce a novel error control protocol for wireless link-layer that targets both reliability and traffic flow control for realtime video communication. Delay Constraint Packet Embedded Error Control (DC-PEEC) protocol employs channel codes (using robust and well-defined code rates) in each packet to ensure that video packets are delivered reliably within an end-to-end delay deadline of realtime video communication. The proposed effort begins by outlining a novel analytic framework to capture video traffic flow at the wireless link-layer. In particular, we develop a queuing model that captures realtime video traffic behavior under reliability and end-to-end delay constraint at the wireless link-layer. Specifically, we model the link-layer buffer as an M/G/1 queuing system with random-sized batch arrivals of video information having a single server representing the link-layer protocol with a general service-time distribution. Using this model, we find an operational code rate for DC-PEEC which guarantees video traffic flow with tolerable latency while utilizing the channel bandwidth effectively. Our contributions can be summarized as follows:

- We develop a novel video communication model to analytically determine the expected latency of realtime video traffic at the wireless link-layer.

- We propose the DC-PEEC protocol for point-to-point link-layer wireless communication which is layer oblivious and provides reliability and flow control for realtime video communication in an optimal and joint manner.

5.1 Realtime Video Communication Model

A fundamental objective in realtime video communication is the delivery of compressed video in a timely fashion to ensure continuous decoding and presentation. To achieve this objective, it is important to capture the behavior of realtime video traffic more rigorously. To that end, we adopt a generic realtime video communication model illustrated in Fig. 5.1. This model captures a dual-level view of video traffic for wireless communication.

The upper-level view (i.e., application layer) shows an *ideal* encoder-decoder buffer model for video communication analyzed in many multimedia literatures [96]. Under this model, uncompressed video pictures first enter the compression engine of the (sender's application layer) encoder at a particular frame rate. The compressed pictures exit the video encoder and enter the encoder egress buffer (at the application layer). Current H.264/AVC standard for realtime video (e.g., video conferencing) uses *baseline profile* to encode video stream [93]. Each video frame is encoded to a compressed Group of Pictures (GOPs) comprising of I-slices followed by P-slices. Similarly at the decoder side, the compressed pictures exit the decoder ingress buffer and enter the decoding engine. Let T_{enc} and T_{dec} represent video encoder and decoder buffers delays respectively. Under this upper-level view the end-to-end delay is governed only by the total delay encountered in both encoder and decoder buffers (encoding and decoding take zero time) and is

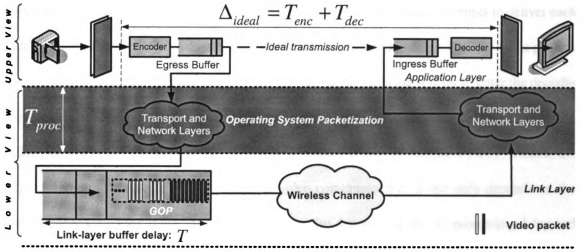


Figure 5.1 Realtime video communication model.

constant. Therefore,

$$\Delta_{ideal} = T_{enc} + T_{dec}.$$

However, in general the compressed video data also encounters different protocol and network delays before it arrives at the decoder buffer. This is captured at the lower-level view shown in Fig. 5.1.

Compressed pictures exiting the encoder buffer are processed by the Operating System (OS) and converted to transport-layer segments. These segments are then converted to network-layer datagrams with appropriate headers added. In this packetization phase, compressed pictures are divided into small video packets. Specifically, the Network Abstraction Layer Units (NALU) of H.264 standard is utilized to encode the video stream. Traditionally each NALU is embedded in MPEG-2/RTP-IP/H.32x/UDP packets. We choose each NALU packet (at the application layer) to be embedded within a single UDP packet (at the transport layer) along with an incremental index for packet sorting at the decoder. Under this setup, the video stream is partitioned into equal and fixed size video slices which is set to be at most the size of Maximum Transmission Unit (MTU) including the UDP packet header size and index used for sorting. We represent OS processing delay at the sender side by T_{tx}. The OS then delivers video packets to wireless link-layer where

they enter the link-layer buffer. Next, the link-layer error control protocol attempts to deliver each video packet to the destination over a lossy wireless channel with the delay T_{ll}. The video packets successfully delivered to the destination are passed up to the application layer to enter the decoder ingress buffer. The processing delay of delivering each video packet from the link-layer to the video decoder buffer at the application layer at the receiver side is denoted by T_{rx}. Hence, the total end-to-end delay of a realtime wireless video communication Δ not only depends on the encoder/decoder buffer at the application layer but also depends on the OS processing delay and the delay of wireless link-layer protocol:

$$\Delta = \Delta_{ideal} + T_{tx} + T_{rx} + T_{ll}.$$

The process of delivering video pictures is constrained to an end-to-end delay $\Delta_{wireless}$. In other words, $\Delta_{wireless}$ is the *total tolerable* delay representing realtime video communication deadline: $\Delta \leq \Delta_{wireless}$. Hence, those video pictures that miss this deadline are unusable for video decoding. This leads to degradation in video quality. Therefore, for realtime video communication we require the following inequality to be satisfied:

$$\Delta_{ideal} + T_{tx} + T_{rx} + T_{ll} \leq \Delta_{wireless}. \tag{5.1}$$

Let T_{proc} represent the maximum processing delay of the OS at the sender and receiver sides and is constant (i.e., $T_{tx} + T_{rx} \leq T_{proc}$). By substituting T_{proc} in Equation (5.1), we have:

$$\Delta_{ideal} + T_{proc} + T_{ll} \leq \Delta_{wireless} \tag{5.2}$$

$$T_{ll} \leq \Delta_{wireless} - \Delta_{ideal} - T_{proc} \tag{5.3}$$

$$T_{ll} \leq T. \tag{5.4}$$

where $T = \Delta_{wireless} - \Delta_{ideal} - T_{proc}$ is the maximum *tolerable* delay at the wireless link-layer (for realtime video communication). Therefore, satisfying the $\Delta_{wireless}$ constraint depends directly on delivering the video packets to the link-layer destination within the link-layer *tolerable*

delay T. To that end, we analyze the video packet traffic behavior at the link-layer and model the link-layer buffer to analytically obtain an average (expected) delay of each video packet at the link-layer (before it is successfully delivered to the destination).

Recall that, compressed video pictures at encoder egress buffer are partitioned into different GOPs. OS packetization process generates video packets from each GOP. Here, based on the GOP content, a random number of video packets are generated for each GOP. Let OS deliver the video packets associated to every GOP to the link-layer at a rate λ. This rate governs the GOP arrival process at the link-layer buffer. Hence, the arrival process of the link-layer buffer is a Poisson process having rate λ. Thus, the link-layer buffer is a queue with customers arriving in random-sized batch where each batch represents one GOP and each customer (in the batch) is one video packet. On the other hand, the link-layer protocol attempts to reliably transmit each video packet in the buffer over a wireless channel. Thus, the algorithm employed at the link-layer protocol dictates the service time distribution of the buffer. Consequently, the link-layer buffer can be modeled as an $M/G/1$ queuing system with random-sized batch arrivals having a single server representing the link-layer protocol with a general service time distribution G.

Let us denote by $\alpha_j, j \geq 1$, the probability that an arbitrary GOP consists of j video packets; and N denotes a random variable representing the size of a GOP (measured in terms of the number of video packets); hence, $P\{N = j\} = \alpha_j$. Therefore, the average arrival rate of entering a video packet in the link-layer buffer is $\lambda_a = \lambda E[N]$. Consequently, the (average) delay for delivering a single video packet to the link-layer destination V becomes [71]

$$V = \lambda E[N] \left\lceil E[G]W_Q + \frac{E[G^2]}{2} \right\rceil, \tag{5.5}$$

where G and W_Q represent, respectively, the service time and the latency of each video packet (the time that a given video packet spends waiting in the link-layer buffer). On the other hand, the latency of a given video packet in the queue is equal to the delay for delivering that packet

to the destination and the latency due to other video packets in the same batch associated with a particular GOP W_{GOP}. Thus

$$W_Q = V + E[W_{GOP}]. \tag{5.6}$$

However, the expected latency in the link-layer buffer due to other video packets is:

$$E[W_{GOP}] = \sum_j E[W_{GOP}|\text{GOP Size}]\frac{j\alpha_j}{E[N]} \tag{5.7}$$

$$= \frac{E[G](E[N^2] - E[N])}{2E[N]}. \tag{5.8}$$

Using mathematical deductions, from Equations (5.5), (5.6), and (5.8) we obtain

$$W_Q = \frac{\frac{E[G](E[N^2]-E[N])}{2E[N]} + \frac{\lambda E[N]E[G^2]}{2}}{1 - \lambda E[N]E[G]}. \tag{5.9}$$

In this work, we use constant-quality encoding to have uniform video quality for all video frames. Consequently, the distribution of the random variable N representing the GOP size is determined by the nature of the video content being encoded, video encoder type, and the desired playback quality. Specifically, the H.264/AVC standard codec uses the quantization parameter (QP) for each macroblock in each video frame to achieve the desired playback quality for that video frame. Under this setting, QP dictates the number of bits allocated to each video frame leading to variable bit rate (VBR) video encoding. This in turn determines the number of video packets in each GOP and consequently the distribution of the random variable N. Using 15 different video sequences[1] with different QP values (e.g, 20 to 36) suggests that the Gamma distribution can be well fitted to the *empirical* distribution of N. For instance, Fig. 5.2 shows that the quantile function of fitted Gamma distribution accurately captures the quantile values of the *empirical* distribution of GOP size captured using QP 20. Therefore, we model the random variable N using a

[1] We use the following video sequences (CIF): akiya, bowing, coastguard, container, deadline, foreman, hall_monitor, husky, mad900, mobile, paris, sign_irene, silent, stefan, students. These sequences can be found at: `http://media.xiph.org/video/derf/index.html?rev=7865`

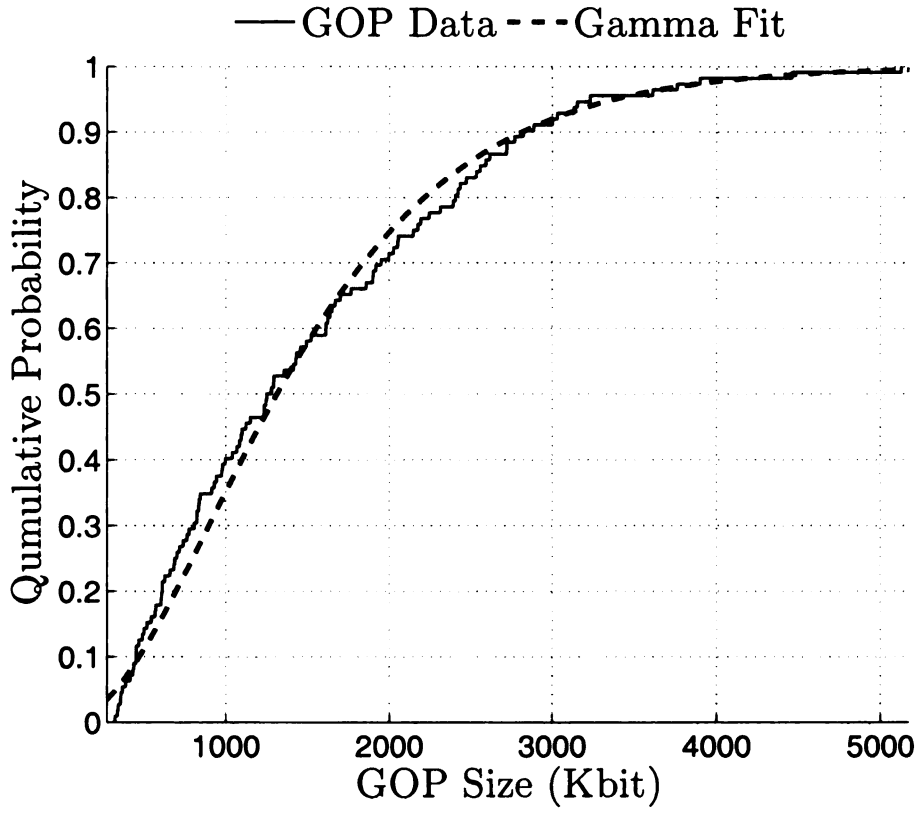


Figure 5.2 The Gamma cumulative probability function fitted on GOP data distribution of 15 video sequences encoded with QP 20.

Gamma distribution with density function

$$f_N(x) = \frac{\theta e^{-\theta x}(\theta x)^{\upsilon-1}}{(\upsilon-1)!}. \tag{5.10}$$

Accordingly, $E[N] = \frac{\upsilon}{\theta}$ and $E[N^2] = \frac{\upsilon}{\theta^2}$.

Consequently, the average time that a given video packet spends at the link-layer buffer before it is successfully delivered to the destination is

$$W = W_Q + E[G] = \frac{E[G](\frac{1}{\theta}+1) + \frac{\lambda\upsilon}{\theta}(E^2[G] - Var[G])}{2 - 2\frac{\lambda\upsilon}{\theta}E[G]}. \tag{5.11}$$

To deploy the above video packet-level queuing model within the context of delay-constrained video, we adopt the following assumption. The realtime constraint of the proposed video communication model demands that the latency of delivering each video packet to the destination does not exceed the link-layer *tolerable* delay deadline T. It is important to note that adhering to this

video packet-level delay constraint satisfies the traditional video-frame level delay constraint. As we pointed out earlier, we adopt a packet-level delay constraint to be consistent with the packet-level queueing model. The average waiting time of each video packet computed in Equation (5.11) should be less than T (i.e., $W \leq T$). Consequently, the link-layer protocol should provide reliable delivery of new video packets to the decoder while ensuring low latency. Toward that end, in the next section, we present the Delay Constraint Packet Embedded Error Control (DC-PEEC) link-layer protocol which employs novel rate-adaptive channel codes to minimize the latency in Equation (5.11) while adhering to the realtime video communication delay constraints.

5.2 Delay Constraint PEEC Protocol

In the previous chapter, we have developed a novel reliable MAC layer that employs the same level of feedback used in 802.11. However, targeting true realtime wireless video will require much-further and much-needed, novel developments and integration of communication solutions. DC-PEEC framework is designed (based on the realtime video communication model developed in the last section) to provide *reliable* and *delay-constraint* information delivery at the link-layer. We first describe the operational communication model of DC-PEEC. Next, we model the service time distribution in Equation (5.11) based on the DC-PEEC functionality. Finally, we describe two important components of the DC-PEEC protocol (namely *Channel State Estimation* and *Redundancy Allocation*) which are essential to maintain continuous and low-latency realtime traffic flow while ensuring maximal reliability and throughput.

5.2.1 Communication Model

In DC-PEEC operational communication model a *transmission interval* τ_i is expressed as the duration in which a transmitter sends the i^{th} message (packet) M_i and receives its corresponding

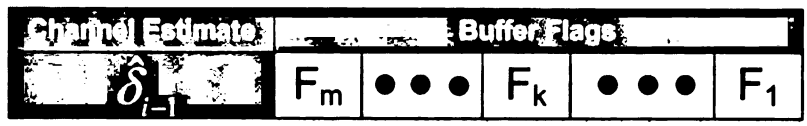


Figure 5.3 The acknowledgment format of the DC-PEEC link-layer protocol.

acknowledgment ACK_i. A transmitter sends a new message after the reception of an acknowledgment.

Sender Side

During τ_i, a sender transmits a message which is represented by the tuple $M_i = [C_i(k_i, x_i), \mathbf{y}_i]$ where k_i represents the number of bits in a video packets which are not being retransmitted. In each τ_i, a transmitter encodes k_i with parity symbols x_i creating a codeword $C_i(k_i, x_i)$. We refer to these parity symbols as type-X parity. The receiver utilizes x_i to decode C_i. Upon successful decoding, C_i is extracted and k_i data bits corresponding to a GOP are passed up to the higher layer. In case of decoding failure, the receiver stores C_i in its buffer and issues a request for more parity symbols. The transmitter also sends additional (type-Y) parity symbols denoted by $\mathbf{y}_i$. The receiver utilizes $\mathbf{y}_i$ symbols to recover old corrupted codewords accumulated in its buffer (e.g., $C_j, j = 1, \cdots, i-1$).

Receiver Side

We assume that the receiver has a finite buffer which can accommodate up to m corrupted messages waiting for recovery. If a newly corrupted packet finds all rooms in the buffer occupied, it does not enter the buffer and is dropped. The status of the receiver is reported to the transmitter via certain flags in an acknowledgment message (ACK) which are called *buffer flags*. Fig. 5.3 illustrates the ACK format. There are m buffer flags are encapsulated in ACK. Let $F_i[k], k = 1, \cdots, m$ represent buffer flags in ACK_i. Each buffer flag is associated with a particular room in the buffer and represents the status of that room. That is, if the k^{th} room is occupied then $F_i[k] = 1$ (as

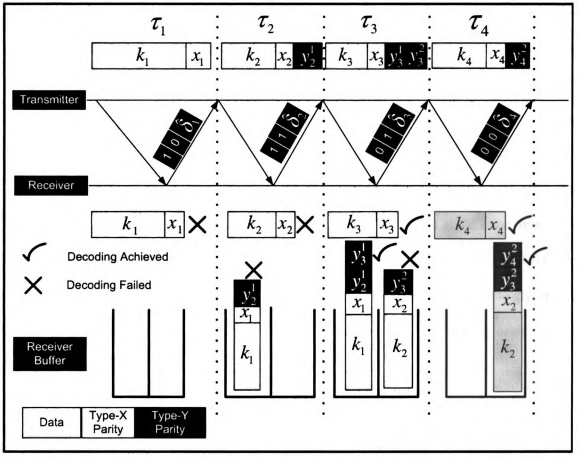


Figure 5.4 An example of DC-PEEC operational communication model consists of four transmission interval.

illustrated in Fig. 5.4). In addition, the receiver estimate of channel condition $\hat{\delta}_i$ in τ_i is also encapsulated in acknowledgment message. Later, we describe channel estimation process by the receiver. The DC-PEEC feedback scheme requires m additional bits for buffer flags and one additional byte for channel estimation feedback to the current IEEE802.11 standard frame control field. In our analysis, we choose $m = 5$.

An example of DC-PEEC operational communication model is illustrated in Fig. 5.4. A short communication consisting of four transmission intervals and buffer capacity of two at the receiver is shown. During the first transmission interval τ_1, a message $M_1 = [C_1(k_1, x_1), \mathrm{y}_1]$ is sent. There are no type-Y parity symbols in M_1, because there is no prior corrupted message in the receiver buffer, so $\mathrm{y}_1 = 0$. A receiver that fails to decode C_1, stores C_1 in its buffer and sends

an acknowledgment $ACK_1 = (1, 0, \hat{\delta}_1)$. In τ_2, the transmitter sends $M_2 = [C_2(k_2, x_2), \mathbf{y}_2 = \{y_2^1\}]$. The receiver uses x_2 to decode C_2 and employs type-Y parity symbols y_2^1 (y_i^j denote additional parity for $C_j, j < i$ transmitted in τ_i) in addition to x_1 to decode C_1. The receiver acknowledges $ACK_2 = (1, 1, \hat{\delta}_2)$, indicating decoding failure of C_2 and $C_1(k_1, x_1 + y_2^1)$. As a result, in τ_3, the sender sends $M_3 = [C_3(k_3, x_3), \mathbf{y}_3 = \{y_3^1, y_3^2\}]$. In τ_3, the receiver successfully decodes C_3 using x_3 and $C_1(k_1, x_1 + y_2^1 + y_3^1)$. Because decoding of C_1 was successful, the receiver sets the buffer flag of the first room to zero (i.e., $F_3[1] = 0$) but at the same time since the receiver is waiting for type-Y parity symbols to perform decoding on C_2, the buffer flag for the second room is set to one (i.e., $F_3[2] = 1$); so $ACK_3 = (0, 1, \hat{\delta}_3)$. Accordingly, in τ_4, the sender transmits $M_4 = [C_4(k_4, x_4), \mathbf{y}_4 = \{y_4^2\}]$. The receiver decodes C_4 using x_4 and $C_2 = (k_2, x_2 + y_3^2 + y_4^2)$ successfully; so $ACK_4 = (0, 0, \hat{\delta}_4)$.

5.2.2 Service Time Distribution under DC-PEEC

DC-PEEC encodes each video packet data bits k_i with x_i type-X parity bits creating a codeword $C_i(k_i, x_i)$ in transmission interval τ_i. Let the wireless channel be in state S_i, each symbol in C_i is distorted independently from the other symbols with probability of ϵ_i. Thus, the distortion of each symbol has a Bernoulli distribution [71] with parameter ϵ_i. As a result, the error introduced in C_i with the length of $|C_i| = k_i + x_i$, can be represented by the random variable E_i which has a binomial distribution $E_i \approx Bi(|C_i|, \epsilon_i)$.

The receiver attempts to retrieve k_i video data bits by utilizing x_i parity bits embedded in C_i. Depending on the decoding algorithm, the receiver can correct certain level of error proportional to the number of parity symbols embedded in the message. DC-PEEC uses an iterative belief propagation LDPC decoder which requires an estimate of channel condition for decoding. Therefore, if the channel estimate is $\hat{\epsilon}_i$, for x_i parity symbols, the receiver is capable of correcting up to $\alpha(\hat{\epsilon}_i) \times x_i$ errors out of $|C_i|$ symbols in the message. Here $\alpha(\hat{\epsilon}_i)$ measures the expected

error-correcting capability of the LDPC soft decision decoder using channel estimate $\hat{\epsilon}_i$. Thus, the probability of successfully recovering video data bits and passing them up to the video decoder is,

$$P(E_i \leq \alpha \times x_i) = F_{E_i}(\alpha(\hat{\epsilon}_i)x_i) \tag{5.12}$$

where $F_{E_i}(.)$ is a cumulative density function of E_i.

Under DC-PEEC, in every transmission a *new* video data bits are decoded by the probability of $F_{E_i}(\alpha(\hat{\epsilon}_i)x_i)$. Upon decoding failure, the packet is buffered, and decoding attempts are performed in the consecutive transmissions by utilizing type-Y parity symbols. Therefore the probability of successful recovery of a particular packet increases as more type-Y parity symbols are transmitted. Consequently, the error correction process of every packet under DC-PEEC has a nonhomogenous geometric distribution [99] with parameter $p_t^i = F_{E_i}(\alpha(\hat{\epsilon}_i)z_t)$ which measures the likelihood of successful decoding (when the channel is in state S_i) on t decoding trial using total parity bits (type-X and type-Y) of z_t. Consequently, the service time distribution G of the link-layer buffer (under DC-PEEC) in the proposed realtime video communication model has the following density function

$$f_G^i(t) = P(\text{Successful recovery on } t \text{ trial}) = p_t^i \prod_{k=1}^{t-1}(1 - p_k^i). \tag{5.13}$$

Using the density function in Equation (5.13), we calculate (numerically) the first and second moments (i.e., $E[G]$ and $E[G^2]$) of service time distribution in Equation (5.11). As a result, we can determine an average latency of each video packet at the link-layer under DC-PEEC algorithm. Later, we utilize this latency to find an optimal redundancy allocation scenario for DC-PEEC.

5.2.3 Channel State Estimation

Maintaining a low latency traffic flow and achieving maximum throughput and reliability under the DC-PEEC communication model will require accurate channel estimation and prediction in

realtime. To that end, we utilize physical-layer and link-layer side-information (provided in current 802.11 implementations). It is important to note that accurate estimation and prediction of the channel condition has a critical impact on the performance of the proposed protocol. This is due to the fact that DC-PEEC employs LDPC codes for decoding link-layer packets, and LDPC codes use a soft decision decoding (an iterative belief propagation method) which requires a knowledge of channel bit error rate (BER). Therefore, it is essential to identify practically observable variables, which can be used for reasonably robust channel state estimation. DC-PEEC uses Signal to Silence ratio (SSR) as side information in every transmission [21].

The Markovian channel model described in Chapter 3 implies that in every transmission the channel is in a particular state represented by a BSC with a unique BER. The objective is to train the Markovian channel model to achieve accurate estimations of BER values for each state of the model. The training process is in an online fashion in the sense that DC-PEEC adjusts its parameters as more and more packets arrived during a session. Specifically, upon a reception of a packet in a transmission τ_i, the receiver obtains the SSR and estimated BER values of packet preamble. We let ssr_i and $\hat{D}_i = g(ssr_i)$ represent the SSR value of packet M_i and its corresponding estimated BER respectively. A receiver creates a one-to-one mapping between each SSR value and each state of a Markov chain [i.e., $(ssr_1 \equiv S_1), \cdots, (ssr_N \equiv S_N)$]. It also keeps record of the observed BER values associated with each SSR, denoted by $(\Delta_i, i = 1, \cdots, N)$. Notice that the number of states of the channel model is dictated by the total number of unique SSR values observed by the receiver. The receiver training process is as follow:

1. Obtain ssr_i and $\hat{D}_i$ of the received packet in τ_i.

2. Find a state where $S_i \equiv ssr_i$.

3. Add $\hat{D}_i$ to the list of observed BER values associated with $(ssr_i \equiv S_i)$: $\Delta_i = \{\Delta_i, \hat{D}_i\}$.

4. Adjust the BER estimation associated with state S_i by taking the average value of the up-

dated Δ_i

$$\hat{\delta}_i = \frac{\sum_{k=1}^{|\Delta_i|} \hat{D}_k \in \Delta_i}{|\Delta_i|}$$

In every transmission interval, the receiver adjusts the parameters of the channel model and sends its estimate of the current channel condition in an acknowledgment message.

5.2.4 Redundancy Allocation

In τ_i, DC-PEEC uses channel estimate of the previous transmission interval $\hat{\delta}_{i-1}$ along with *buffer* flags in ACK_{i-1} to allocate data and parity symbols for M_i. Specifically, DC-PEEC uses $\hat{\delta}_{i-1}$ as its estimate of the channel BER in current transmission interval τ_i; so $\hat{\epsilon}_i = \hat{\delta}_{i-1}$.

The realtime video communication model requires the delivery of each video packet at the link-layer within the link-layer *tolerable* delay T. Therefore DC-PEEC should provide reliable and rapid delivery of video information to meet this deadline. Meaning, the proposed scheme should allocate parity bits to each video packet such that the average time that a video packet spends at the link-layer (before it is delivered to the video decoder) is minimized. In summary the objective is to find an optimal allocation of parity bits in every transmission to

1. ensure the reliability video data bits transmission over the channel while utilizing channel bandwidth efficiently, and

2. reduce the video packet latency at the link-layer obtained in Equation (5.11) to meet the link-layer *tolerable* delay T.

To satisfy the first objective, DC-PEEC should allocate minimum number of parity symbols which guarantee reliable packet delivery. DC-PEEC uses $\hat{\epsilon}_i$ to determine type-X parity symbols x_i. Let r_{x_i} represent the rate of type-X redundancy in the message. That is, r_{x_i} represents the minimum number of symbols required to correct each data symbol transmitted over the BSC channel with BER $\hat{\epsilon}_i$. For this BSC, there is an uncertainty about the correctness of each transmitted

symbols imposed by $\hat{\epsilon}_i$. Conceptually, we can resolve this uncertainty by sending additional information to the receiver which shows whether the channel caused an error. This is equivalent to sending a redundancy that informs the receiver about the BSC status with the transmission of every symbol. According to the Shannon's channel coding theorem [55] this redundancy must have a minimum number of symbols equal to the entropy of a BSC channel with crossover parameter $\hat{\epsilon}_i$. Hence, for transmission of k_i video data bits, DC-PEEC should allocate *at least* $k_i \times H(\hat{\epsilon}_i)$ parity symbols in $C_i(k_i, x_i)$. Thus we have

$$x_i = \beta_i k_i H(\hat{\epsilon}_i) = \beta k_i \left(\hat{\epsilon}_i \log_2(\frac{1}{\hat{\epsilon}_i}) + (1 - \hat{\epsilon}_i) \log_2(\frac{1}{1 - \hat{\epsilon}_i}) \right), \tag{5.14}$$

where β_i is an adjustable parameter which should be tuned to satisfy the delay constraint (second objective) when the channel is in state S_i.

Let W_i represent the video packet latency at the link-layer buffer calculated in Equation (5.11) when the channel is in state S_i. By assuming that a channel permits the transmission of maximum n_i bits in τ_i, the length of a packet should not exceed n_i. To ensure that the second objective is met (i.e., $W_i \leq T$), we need to find an optimal value of β_i in Equation (5.14). This leads us to the following optimization problem

$$\begin{aligned} &\min \beta_i \quad \text{subject to:} \\ &W_i \leq T, x_i = \beta_i k_i H(\hat{\epsilon}_i) \\ &k_i + x_i \leq n_i, \beta_i \geq 1 \end{aligned} \tag{5.15}$$

By solving (5.15), DC-PEEC calculates β_i^* (minimum value for $\beta_i, \beta_i \geq 1$) such that the parity bits x_i allocated to a new video packet k_i reduces the average delay W_i (when the channel is in state S_i) to meet the deadline T. Consequently, for a video packet with length k_i, DC-PEEC allocates $x_i = k_i \beta_i^* H(\hat{\epsilon}_i)$ type-X parity symbols and creates a codeword $C_i(k_i, x_i)$.

According to the DC-PEEC communication model each buffer flag in the acknowledgment message ($F_{i-1}[j]$ $j = 1, \cdots, m$) indicates the status of a particular room in the receiver. If

$F_{i-1}[j] = 0$, no parity symbol is necessary. However, $F_{i-1}[j] = 1$ indicates that room j contains a particular codeword C_k transmitted in some $\tau_k, k < i$ which requires additional redundancy symbols for recovery. Accordingly, C_k with length n_k requires at most $T_i^k = n_k \beta_i^* H(\hat{\epsilon}_i)$ parity symbols for error recovery. However, for C_k, $x_k + \sum_{l=k+1}^{i-1} y_l^k$ parity symbols are already transmitted in the previous transmission intervals $\tau_k, \cdots, \tau_{i-1}$. Thus, type-Y parity symbols necessary to transmit in τ_i for C_k is $y_i^k = T_i^k - x_k + \sum_{l=k+1}^{i-1} y_l^k$.

The transmitter constructs a message $M_i = C_i(k_i, x_i, \mathbf{y}_i)$ according to the message distribution specified by DC-PEEC. This distribution is adaptively computed in every transmission interval based on the channel and the receiver's buffer conditions. It guarantees reliability and tolerable latency for realtime video communication. Therefore, the overall video quality increases when the DC-PEEC protocol is utilized. To ascertain the capability of DC-PEEC, in the next section, we demonstrate the performance gain of realtime video communication using the proposed protocol on real channel traces with various error conditions.

5.3 Experiment

In this section, we evaluate the performance of realtime video communication over real wireless channel traces collected on an 802.11b WLAN as described in Section 3. In particular, we analyze the impact of the variation in *quality* and *scene change* of a particular video stream transmitted over wireless channels with varying error conditions. We evaluate the performances for DC-PEEC, IEEE802.11 ARQ and HARQ protocols. For the following experiments, all three protocols DC-PEEC, HARQ, IEEE802.11 ARQ are implemented with OMNET++ network simulator [105]. We use an Adaptive LDPC (A-LDPC) codes [104] for channel coding operations in DC-PEEC, Reed-Solomon codes [67] for HARQ, and `INET Framework` IEEE802.11 module in OMNET++ for implementation of ARQ mechanism. The performance measure is *Y-PSNR*

which is a function of the mean square error (MSE) between the values of the original and decoded Y frame pixels,

$$Y_{PSNR} = 10log_{10}\left[\frac{255^2}{||\mathbf{y_{org}} - \mathbf{y_{dec}}||_2^2}\right].$$

In case of, frame loss, we use the variance of the pixels to calculate the frame PSNR as follows:

$$Y_{PSNR} = 10log_{10}\left[\frac{255^2}{||\mathbf{y_{org}} - \overline{\mathbf{y_{org}}}||_2^2}\right].$$

where $\overline{\mathbf{y_{org}}}$ represents the mean of the values of the original frame pixels and $||.||_2$ is the $\mathcal{L}^2$-norm. We have conducted the following experiments for all the 15 video sequences listed in Section 5.1, however for brevity only results for *Stefan CIF* and *Foreman CIF* are presented.

5.3.1 Quality Selection

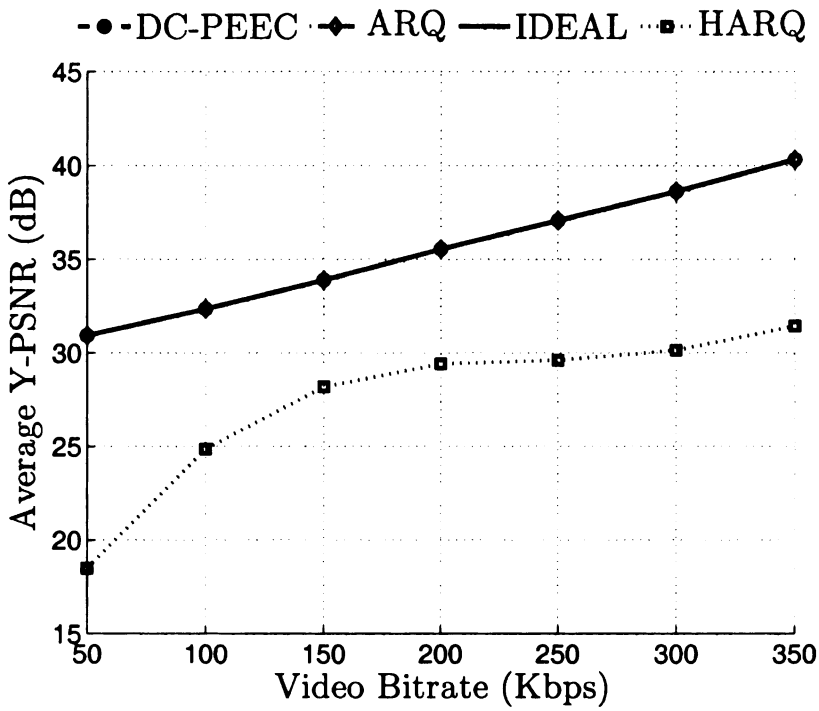
Often in realtime multimedia application (such as video streaming and video conferencing), a user can choose the video playback quality based on available bandwidth and latency of Internet connection. As mentioned before, we configure the H.264/AVC JM14.0 codec to achieve uniform playback quality by setting the quantization parameter (QP) to a specific value for each frame. Under this setting, QP dictates the number of bits allocated to each video frame leading to variable bit rate (VBR) video encoding. Accordingly, the sender should deliver video data at or above particular rate which is specified by the *video bitrate*. The value of *video bitrate* determines the video decoder tolerable delay. Those packets which miss their deadlines are unusable by the decoder, leading to degradation in video quality.

We setup the simulations based on the realtime video model depicted in Fig 5.1. We use H.264/AVC JM14.0 codec [106] to serve as video encoder/decoder. The simulation setup is as follows: A video stream is encoded based on a specific QP value at the sender-side in the application layer. The encoded video streams (video packets) are buffered at the sender's link-layer to be transmitted over the wireless channel. The DC-PEEC protocol is simulated with the network sim-

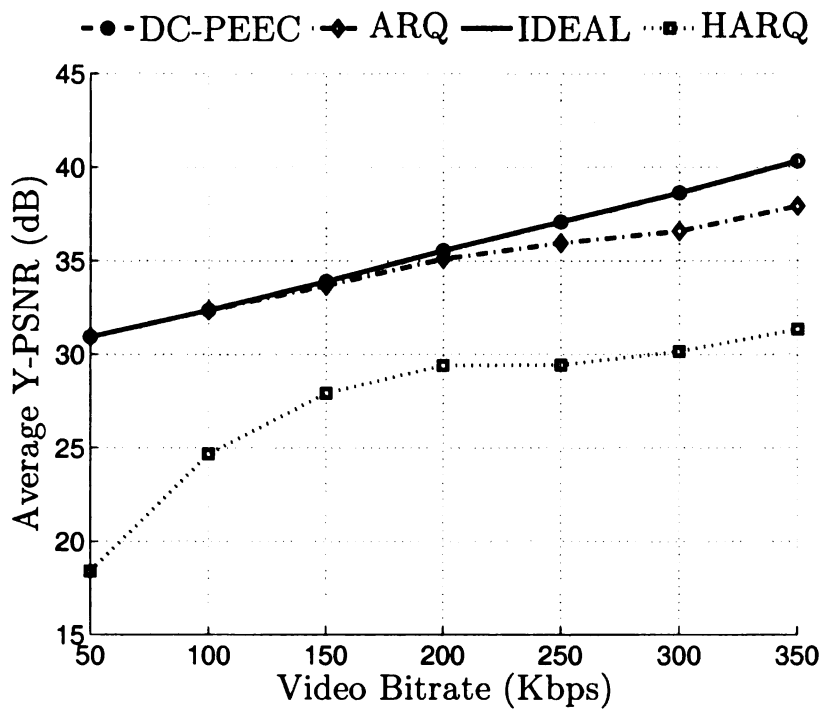
ulator OMNET++ software [105]. DC-PEEC encodes each video packet using LDPC codes [104] and transmits the encoded packet over a wireless channel. Each transmitted packet tx is distorted based on the channel traces tr. Specifically, an XOR operation is performed between the trace packet and DC-PEEC packet: $\mathbf{rx} = \mathbf{tx} \oplus \mathbf{tr}$.

The corrupted packet **rx** is decoded using LDPC. The LDPC decoder uses BER estimates determined by a channel model trained using previously received packets. If the packet is not decoded successfully, it is stored in receiver's link-layer buffer and additional redundancy is requested according to DC-PEEC. Otherwise, the decoded video data is passed up to the (video decoder at) receiver's application layer. The simulation is terminated when all the video slices are transmitted by the sender. We repeat the simulation for different *video bitrates* as governed by different QP values ranging from 20 to 36. We increase QP by two to obtain nine different *video bitrates*. We transmitted the video bitstreams over 12 different channel traces with different BERs. Therefore, for each video sequence, a total of 168 simulations were conducted. We repeat these simulations to compute the performances of IEEE802.11 ARQ and HARQ protocols using the above settings.

Fig. 5.5 shows the PSNR values for *Stefan CIF 30fps* sequence transmitted by different link-layer protocols over various channel traces. In this figure, the solid line represents the *best* PSNR value that can be achieved for each *video bitrate*. We observe that DC-PEEC and IEEE802.11 ARQ protocols perform well over channels with low BERs in comparison with HARQ. This is due to low probability of packet corruption which mitigates the need to perform retransmissions (in case of ARQ) or allocate many parity bits (in case of PEEC). However, since HARQ is not channel adaptive [44], it allocates a fixed amount of redundancy regardless of channel conditions. As a result HARQ does not send enough video data bits on time (causing frequent frame losses) and the degradation in PSNR value. However, as the BER increases, we observe that the performance of IEEE802.11 ARQ decreases dramatically while DC-PEEC manages to achieve significantly better quality. In particular, we observe 5-10 dB PSNR gains over channels with BER greater than 0.005

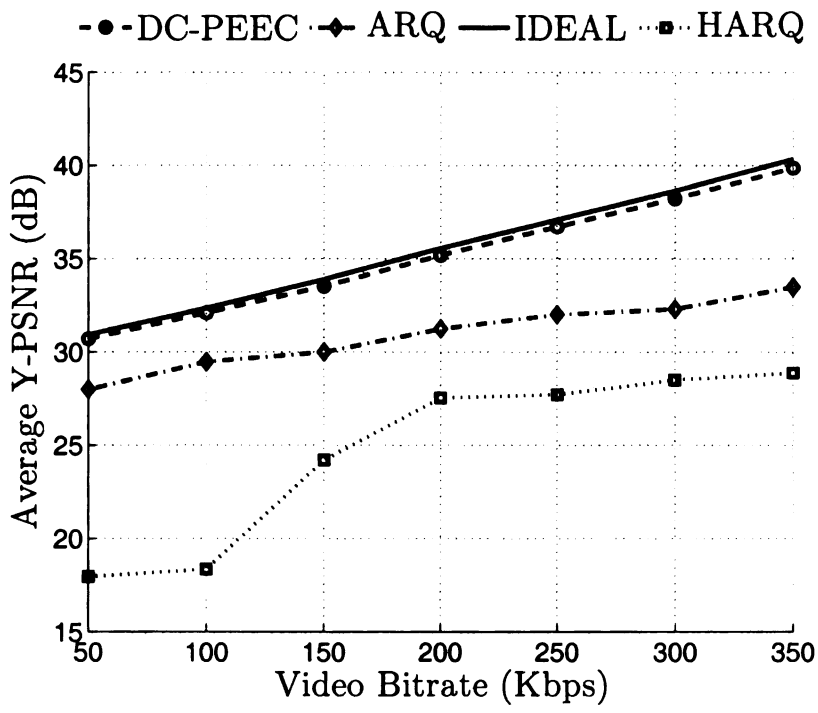


(a) BER: 0.0001

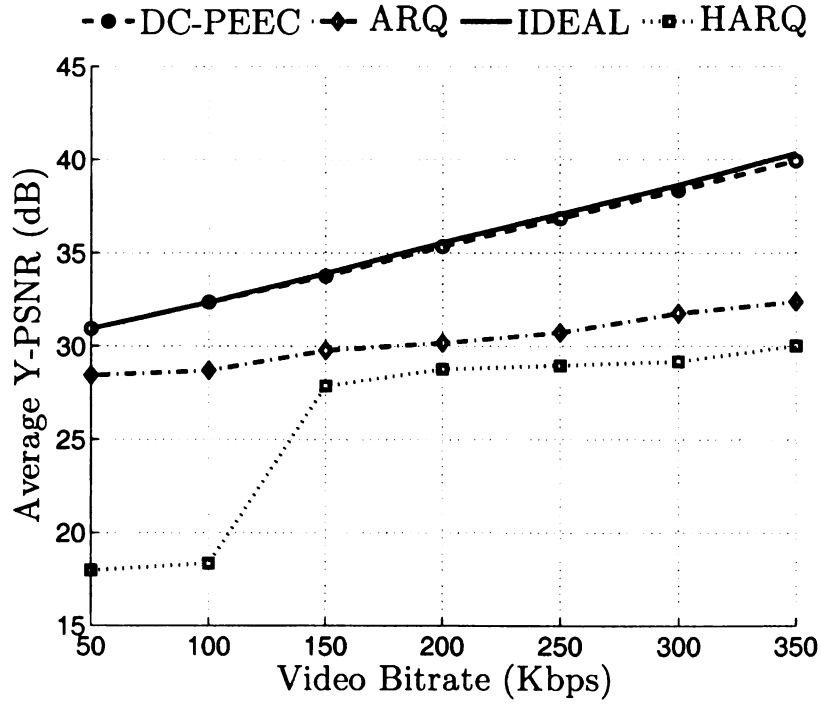


(b) BER: 0.001

Figure 5.5 Average $Y - PSNR$ of "stefan" sequence with respect to variation of video bitrate over different channel traces. The solid line in each figure represents maximum achievable video quality for each video bitrate.

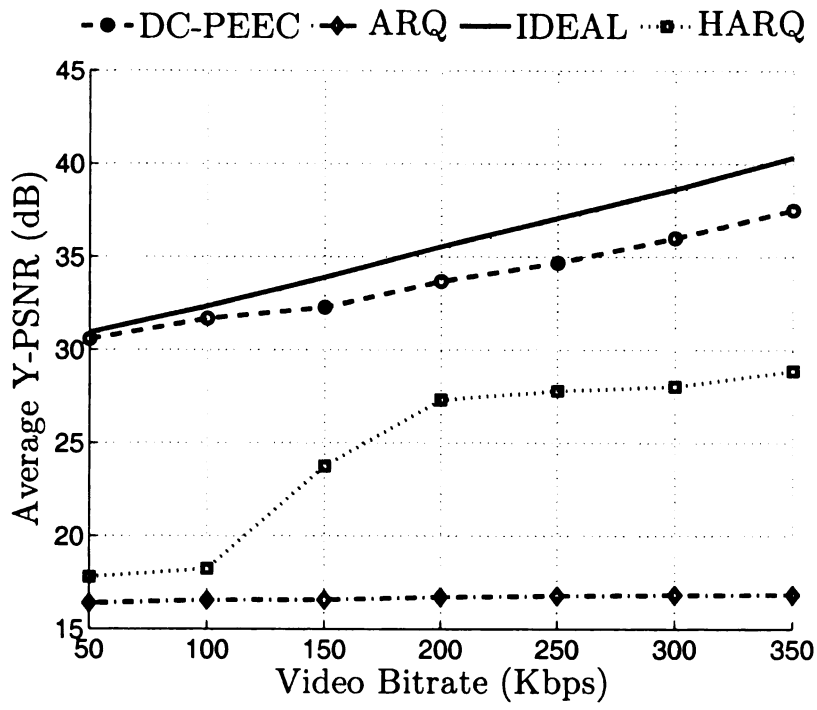


(c) BER: 0.005

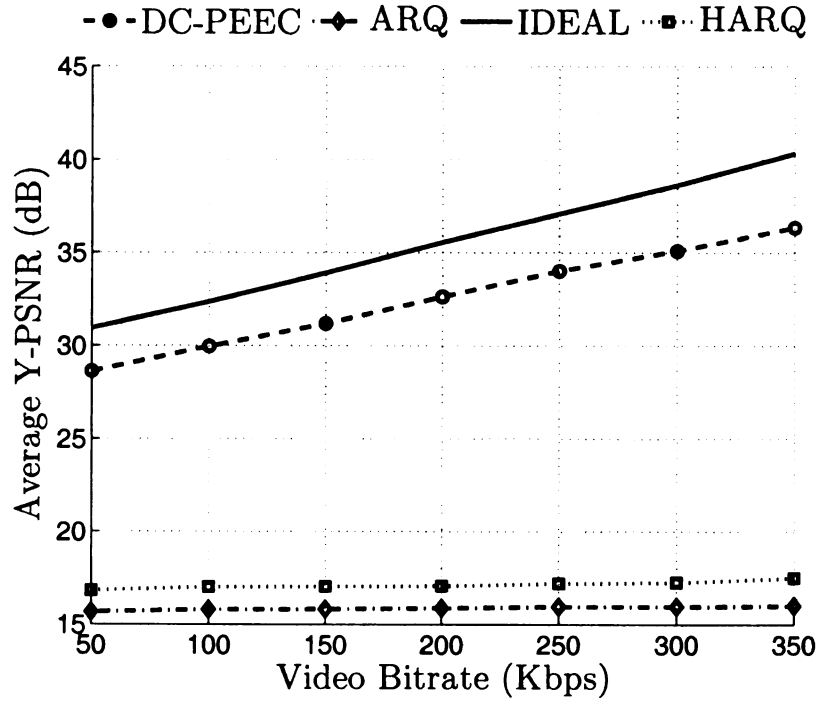


(d) BER: 0.01

Figure 5.5 continued.



(e) BER: 0.015



(f) BER: 0.03

Figure 5.5 continued.

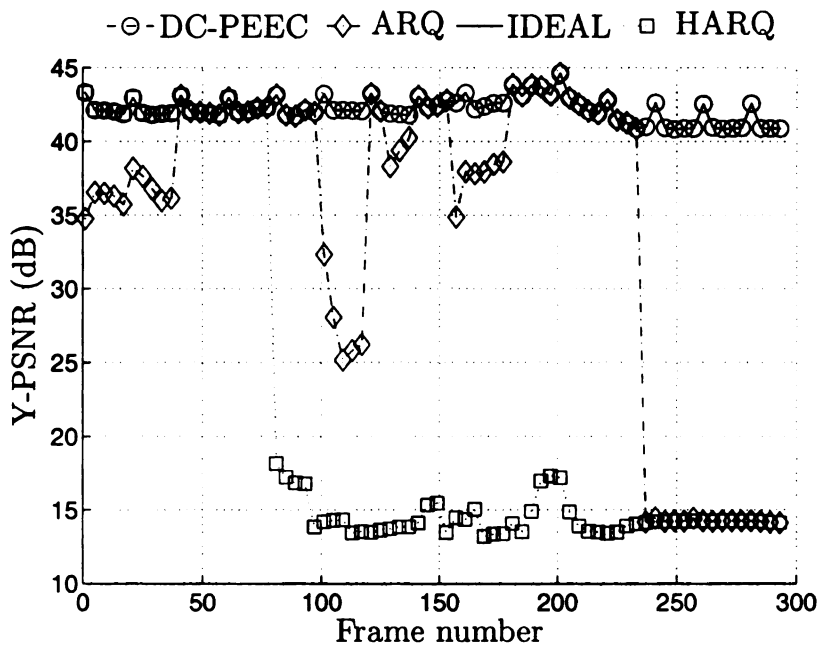
for various video bitrate. This is due to channel adaptive nature of DC-PEEC which allocates parity bits based on the end-to-end delay constraint. As a result, DC-PEEC sends sufficient video data bits before the deadline T expires (while ensuring data reliability).

5.3.2 Scene Change

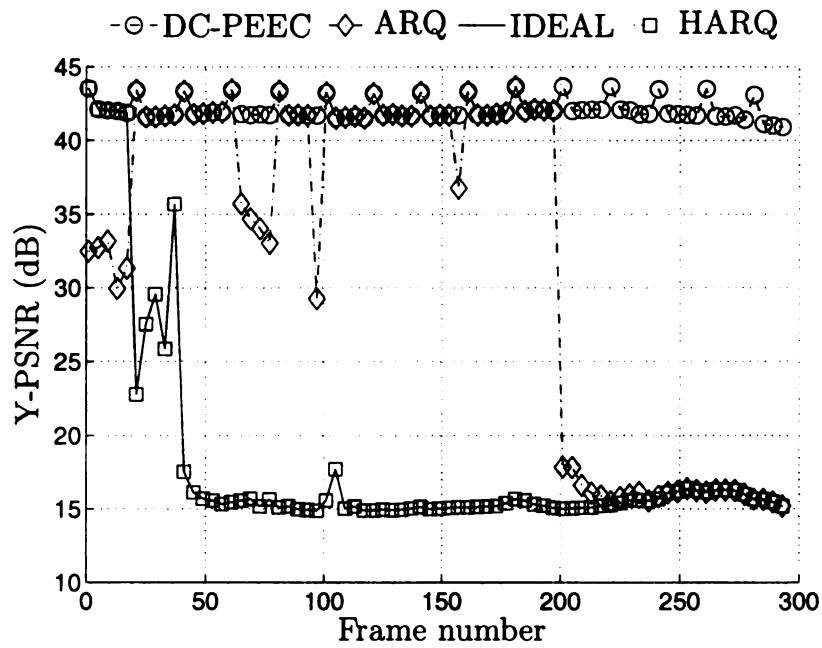
The simulation results presented in the last section illustrates the average playback video quality. However, for realtime multimedia applications, a high *average* PSNR value is not necessarily an indication of smooth realtime video playback. For instance, a single frame skipping, frame loss or jitter in video conferencing is sufficient to deteriorate the system performance from user viewpoint. Therefore it is vital that *all* video frames are delivered and decoded on time. This objective is more difficult to achieve for video sequences with scene change where each encoded GOP has significantly different bitrate. Hence, it is important for the link-layer protocol to deliver video data bits rapidly such that all video frames are decoded with high quality regardless of the variations in channel conditions.

The *average* PSNR value is shown in Fig. 5.5b for *Stefan CIF* encoded with QP 20 (video bitrate of 350Kbps) transmitted over a wireless channel with BER 0.001. It suggests that both IEEE802.11 ARQ and DC-PEEC should have comparably similar high performance (PSNR is above 35dB) for *Stefan CIF* sequence. Fig. 5.6 shows PSNR values of individual decoded video frames for *Stefan* and *Foreman CIF* for this simulation. Interestingly, Fig. 5.6b shows that although IEEE802.11 ARQ achieves high PSNR value for most of the frames, it cannot deliver all of them (e.g., frame #100) on time and therefore we observe fluctuations in frame quality.

To illustrate the impact of these fluctuations on the playback quality of realtime multimedia applications, Fig. 5.7 provides a sample of five consecutive frames captured from *Stefan CIF* sequence. In this Figure, the frame: distortion, freeze and loss are easily identifiable under IEEE802.11 ARQ and HARQ. This stems from the fact that these protocols are not adaptive to a



(a) Foreman CIF 30 fps



(b) Stefan CIF 30 fps

Figure 5.6 The $Y - PSNR$ values of sequence frames encoded with QP 20 and transmitted over channel with BER 0.001.

sudden change in channel condition. Although the wireless channel has overall low BER (0.001), it still has some temporal regions with severe error conditions which lead to the fluctuations in frame quality.

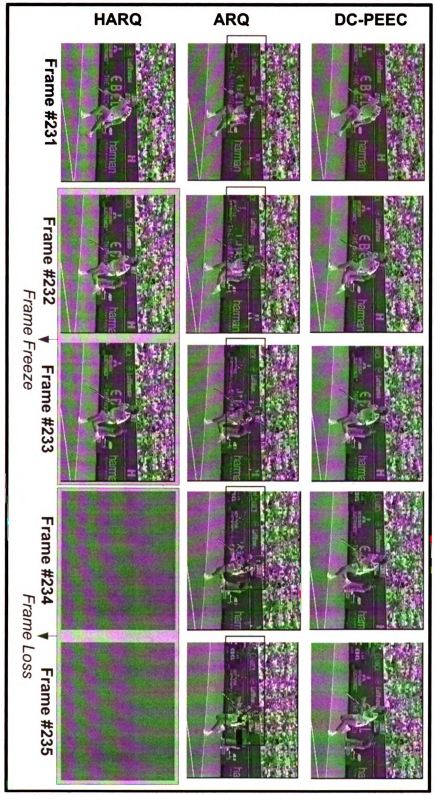
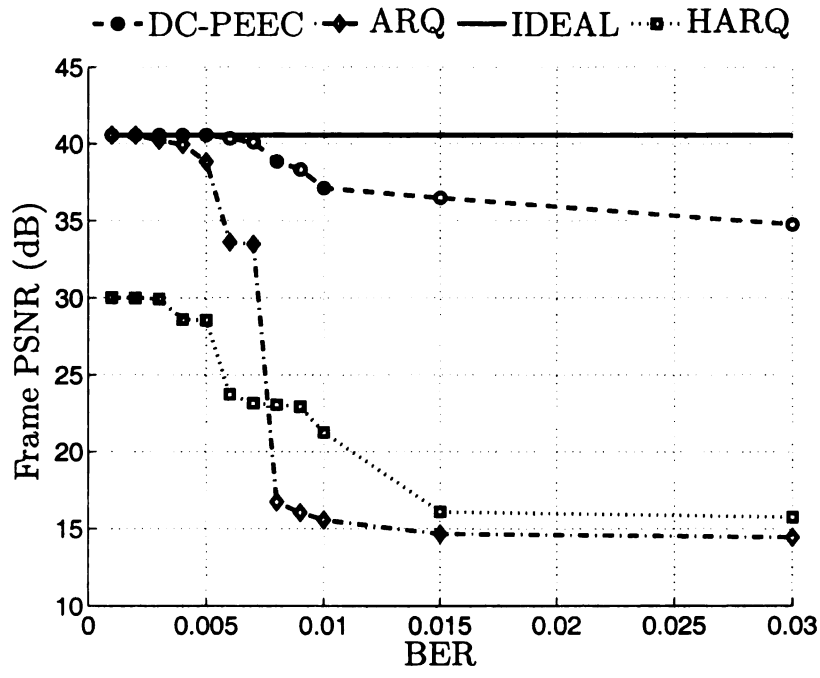
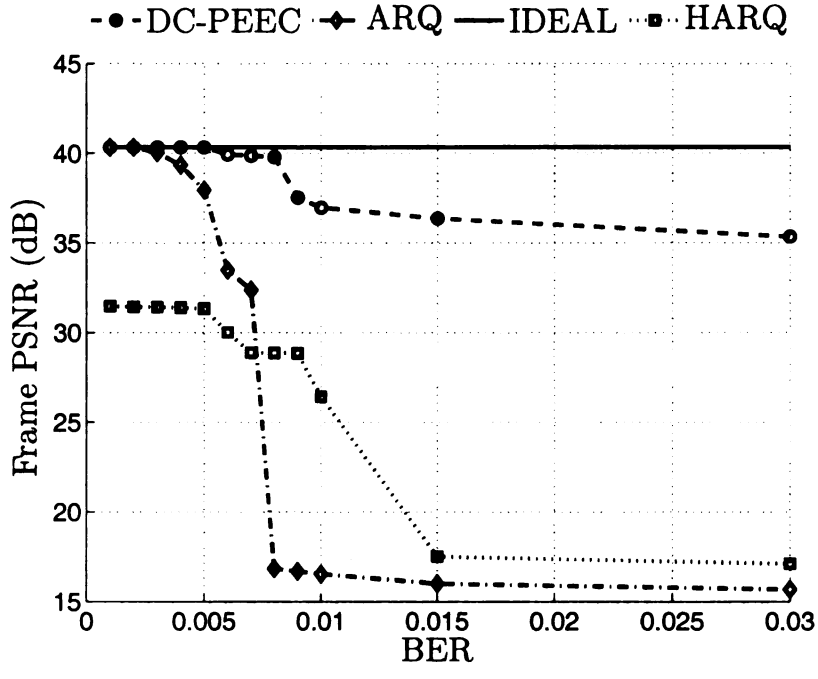


Figure 5.7 A quality comparison of five consecutive frame captures of "Stefan CIF" delivered by different error control protocols.



(a) "Foreman CIF"



(b) "Stefan CIF"

Figure 5.8 The impact of BER variations in frame quality (encoded with QP 20) under different error control protocols

Fig. 5.8 illustrates the impact of BER variations in frame quality. As we expected, for low values of BER (BER less than 0.005) IEEE802.11 ARQ and DC-PEEC performs close to ideal

frame quality. However, as BER increases the performance of IEEE802.11 dramatically decays and falls below HARQ at the same time DC-PEEC maintains the high visual quality. For transmissions where BER is more than 0.015, DC-PEEC achieves 10-15dB PSNR gain over HARQ and IEEE802.11. This result clearly demonstrates the efficacy of DC-PEEC in allocating parity bits for video transmissions to achieve reliability condition and meet end-to-end delay constraint.

5.4 Discussion

In this chapter, we investigated the problem of reliable and delay sensitive wireless video communication. We proposed a queuing system which captures the behavior of video traffic flow at the wireless link-layer. Using this model, we introduced DC-PEEC, an error control protocol for wireless link-layer which deploys various channel codes to ensure the delivery of video packets in a timely fashion while efficiently utilizing channel bandwidth. In particular, DC-PEEC uses the receiver feedback to estimate the channel condition and deploys optimal channel code rates using realtime video communication model to construct each video packet. The performance of the proposed scheme was analyzed by simulating realtime video communication over real channel traces collected on 802.11b WLANs using H.264/JVT JM14.0 video codec. The experimental results demonstrated performance gains of 5-10dB for different realtime video scenarios. In the next chapter, we pursue a paradigm shift where both reliability and stability are ensured at the underlying link-layer. We investigate the feasibility of designing stable and reliable link layer over point-to-point (single-hop) 802.11 channels; and more importantly we study the feasibility of achieving significantly improved throughput by using this type of link-layer.

CHAPTER 6

ACE: A Reliable and Stable Wireless Link Layer

Reliable communication over wireless channels is very challenging since wireless links are error-prone and susceptible to noise imposed by fading, interference and mobility. Additionally, wireless networks need to accommodate diverse traffic types with various requirements for rate, reliability and delay. The wide variety in traffic rate requirements leads to a large variance in the traffic volume injected into the network which often leads to throughput instability. This is exacerbated due to errors introduced in the wireless network. Leading link-layer protocols focus primarily on reliability and ignore the stability aspect of wireless communication; relying on (or arguably shifting the problem to) higher layers to provide stable flow control for both realtime and non-realtime traffic. It is our belief that one of the important goals for any link layer protocol is to provide system stability by ensuring that higher layers are neither starved for information packets, nor is there a glut of packets leading to buffer overflows. More specifically, the current link-layer paradigms aim at providing reliability in hop-to-hop communication without regard to traffic demand. One outcome of failures at the link-layer, resulting from errors in the wireless network, is that the network layer assumes there are broken links in the current packet route. This

leads to nonessential determination of new packet routes by the routing agent [76]. Similarly, wireless errors are interpreted as congestion by the transport-layer resulting in an unnecessary drop in transmission rate [82, 84].

Current IEEE802.11 ARQ protocol [29], which focuses only on reliability, attempt to recover from losses using retransmissions. Corrupted packets are discarded without regard to the number and location of the errors. This methodology is designed to ensure "reliability in the long run", i.e. the packet would eventually be recovered. However this approach suffers from degradation of throughput rate and overall system instability. This is easy to see, since even a single bit error in consecutive packets leads to packet drops and therefore discarding of a large number of correct data bits. As a result, the network utilization deteriorates steadily and rapidly with increasing channel Bit Error Rate (BER). In an alternative approach, Hybrid ARQ (HARQ) protocols are proposed, which make use of incremental channel codes to achieve reliable transmission over wireless channels using fewer packet transmissions [43, 44]. However, both the ARQ and HARQ based approaches follow the conventional paradigm and do not address throughput stability issues raised by varying traffic demand and intensity. This design strategy has led to a great deal of inefficiency in throughput and to other major technical issues and challenges at higher layers. A well-known example is the TCP over-wireless performance degradation phenomenon, which led to major research efforts and numerous studies [84] in attempt to mitigate the shortcoming of the lower layers.

In this chapter, we propose a paradigm shift where both reliability and stability are ensured using an Automatic Code Embedding (ACE) wireless link-layer protocol. The proposed wireless ACE link-layer (a) employs a theoretically-sound framework and a corresponding strategy for embedding channel codes, using robust and well-defined code rates, in each packet; and (b) selects the code rates in an optimal and constrained manner to ensure reliability, stability, and maximum throughput. We believe that this work is the first to present a theoretical framework for

analyzing and designing a wireless link-layer protocol that targets system stability in conjunction with reliable communication. We begin by outlining a novel joint analytic framework to predict system behavior under ACE. Specifically, we first obtain an upper bound on operational code embedding rate that ensures reliability. Next, we develop a queuing model that captures system behavior under stability condition. In particular, we describe the link-layer failures as an on-off source model using a two-state Continuous Time Markov chain (CTMC) model. We deploy fluid approximations to analytically characterize the buffer growth. By utilizing these models, we find a lower bound on operational code embedding rate which guarantees stable operation while utilizing the channel bandwidth effectively. An important conclusion of the above analysis is that various traffic demands (in terms of reliability and stability requirements) can be met using a packet-by-packet code embedding rate constraint that is independent of traffic type. This leads to simplistic, traffic-independent and elegant design rules for the ACE protocol, while providing reliability and stability in an optimal and joint manner.

The ACE protocol is a point-to-point wireless communication protocol where the receiver stores corrupted packets in its buffer for further recovery. The channel conditions are estimated using simple feedback mechanism. ACE utilizes receiver BER estimate and buffer flags encapsulated in the ACK message to determine the composition of the next packet to be transmitted by the sender. We demonstrate experimentally that ACE provides both rapid and reliable wireless data transmission under varying channel conditions. Our contributions can be summarized as follows:

- We present two distinct analytical frameworks to determine optimal code embedding rates which ensure system reliability and stability. We further show that these conditions are met using a packet-by-packet code embedding rate that is independent of traffic type.

- We propose the ACE protocol for point-to-point link-layer wireless communication which is layer oblivious and provides reliability and stability in an optimal and joint manner.

6.1 Preliminaries

In this section we determine the likelihood of successful transmission by introducing *distortion* model for ACE. This model along with the channel model described in Chapter 3 provide essential tools in finding an optimal code embedding rate constraint for the ACE protocol. Note that in the following discussion, the terms "message", "packet" and "codeword" are used interchangeably.

6.1.1 Distortion Model

The *distortion model* measures the distortion level of a received packet and computes the likelihood of successful recovery of the packet under embedded channel coding. To develop this model, we let $C_i(k_i, x_i)$ represent the transmitted codeword in τ_i where k_i is the number of data symbols which are encoded with x_i parity symbols. Letting the wireless channel be in state S_i, each symbol in C_i is distorted independently from the other symbols with probability of ϵ_i. Thus, the distortion of each symbol has a Bernoulli distribution with parameter ϵ_i. As a result, the error introduced in C_i with the length of $|C_i| = k_i + x_i$, can be represented by the random variable E_i which has a binomial distribution a can be written as $E_i \approx Bi(|C_i|, \epsilon_i)$. In practice, $|C_i|$ is a relatively large number, and ϵ_i is very small. So, we can approximate E_i with a Poisson distribution with rate $\lambda_{E_i} = |C_i|\epsilon_i$.

The receiver attempts to retrieve k_i data symbols by utilizing x_i parity symbols embedded in C_i. Depending on the decoding algorithm, the receiver can correct certain level of error proportional to the number of parity symbols embedded in the message. Specifically, for x_i parity symbols, the receiver is capable of correcting up to $\alpha \times x_i$ errors out of $|C_i|$ symbols in the message. Here α measures the expected error-correcting capability of a particular decoder. For example, the error-correcting capability of Reed-Solomon codes is half as many as redundant symbols (i.e., $\alpha = 0.5$) [67].

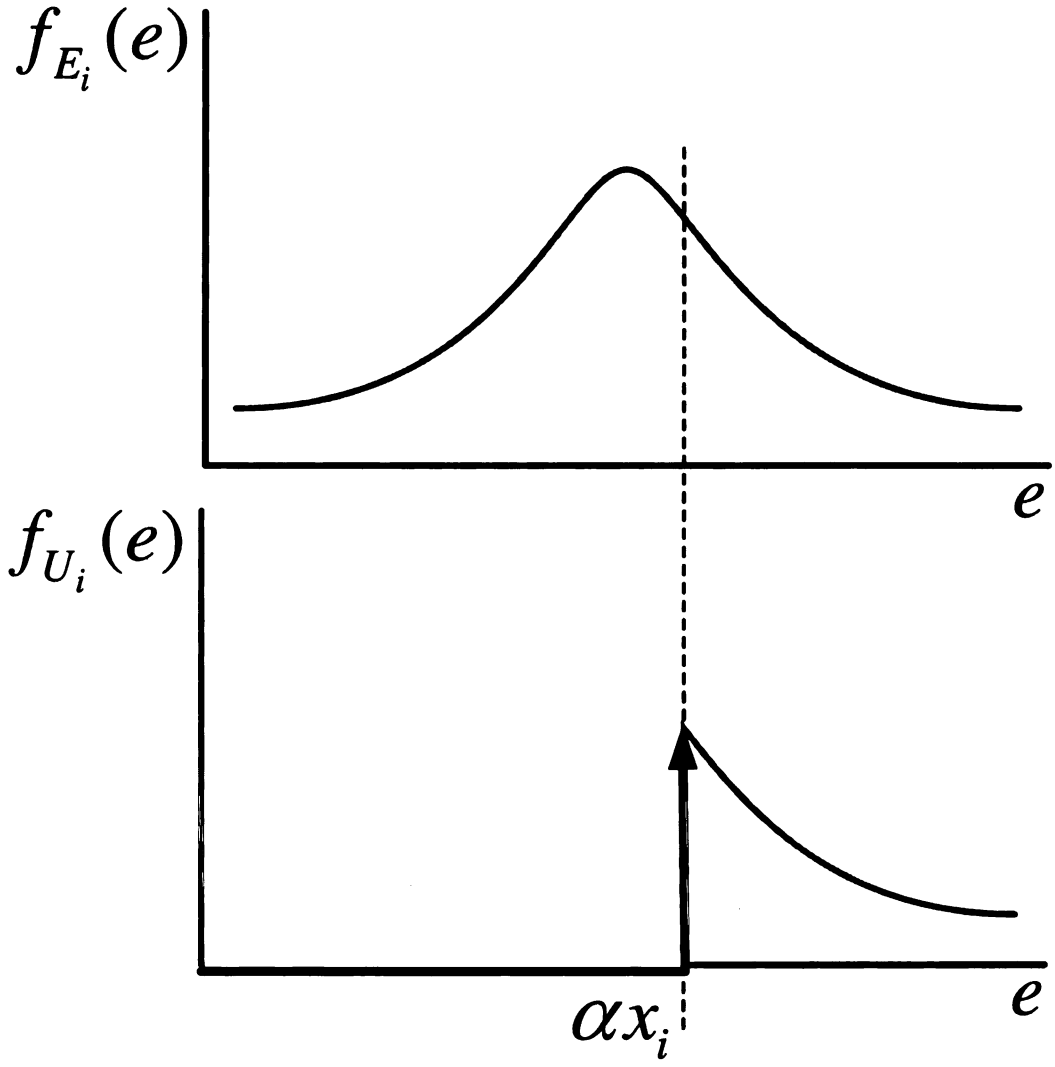


Figure 6.1 The density of error *after decoding* is truncated on αx_i.

The distortion level E_i is random and unknown to the receiver and therefore the notion of partial recovery is unrealistic in error correction. Meaning, the receiver can either correct all errors in C_i declare successful decoding or just assumes that no recovery is achieved. So the level of distortion in C_i *after decoding*, denoted by U_i, is

$$U_i = g(E_i, x_i) = \begin{cases} 0 & E_i \leq \alpha x_i \\ E_i & \text{otherwise} \end{cases} \tag{6.1}$$

Equation (6.1) shows that the distribution of U_i is equivalent to the distribution of E_i *truncated* on αx_i (see Fig. 6.1). Therefore, U_i has the probability density function

$$f_{U_i}(u) = \begin{cases} F_{E_i}(\alpha x_i) & u = 0 \\ f_{E_i}(u) & u > \alpha x_i \end{cases} \tag{6.2}$$

where $F_{E_i}(u)$ is a cumulative density function of E_i. Correspondingly, the probability of successful decoding of n_i is equivalent to the probability that $U_i = 0$. That is,

$$P(U_i = 0) = F_{E_i}(\alpha x_i) = \sum_{d=0}^{\lfloor \alpha x_i \rfloor} e^{-\lambda_{e_i}} \frac{\lambda_{E_i}^d}{(d)!}. \tag{6.3}$$

This density determines the likelihood of successfully error recovery using α-error correcting codes. In the following section, we use this likelihood to determine an optimal code embedding rate necessary for reliable and stable operations in wireless communication.

6.2 ACE Code Embedding Rate

This section describes two distinct analytical frameworks that determine the upper and lower bounds on operational code embedding rates under Automatic Code Embedding (ACE) wireless link-layer protocol. The first analytic framework determines the upper bound on operational code rate that ensures reliability. The second framework develops a queuing model that captures stable system behavior and identifies the lower bound on operational code rate under stability condition. The operational code embedding rate measures the fraction of data symbols that are embedded in a particular codeword. For instance a codeword $C_i(k_i, x_i)$ is generated based on the code rate $R_i = \frac{k_i}{k_i + x_i}$.

6.2.1 Code Rate: Reliability

One of the main objectives in wireless communication is reliable data transmission. We define reliability as follows:

Definition 3. *System is reliable when information is transmitted with no or diminishing error over a wireless channel.*

Recall that during τ_i the channel is in state S_i and every transmitted symbol is altered by the error probability ϵ_i. The sender uses the channel $k_i + x_i$ times to transmit a codeword $C_i(k_i, x_i)$ encoded with parity symbols x_i. The amount of error introduced in the received codeword (on average) is $(k_i + x_i)\epsilon_i$. According to the distortion model developed in Section 6.1.1, a decoder can correct up to αx_i errors in the codeword. Thus, to successfully deliver k_i data symbols over a wireless channel, the following inequality has to be satisfied:

$$(k_i + x_i)\epsilon_i \leq \alpha x_i. \ \ \epsilon_i \in F_N$$

In addition, if a channel permits the transmission of n_i symbols in τ_i, then the length of a codeword should not exceed n_i. Based on these requirements, we have the following optimization problem:

$$\begin{aligned} &\max k_i \quad \text{subject to:} \\ &k_i\epsilon_i + x_i(\epsilon_i - \alpha) \leq 0 \text{ and } k_i + x_i \leq n_i. \end{aligned} \tag{6.4}$$

This leads us to find an upper bound on the operational code embedding rate that ensures reliability which is given by the following Lemma:

Lemma 1. *The operational code embedding rate that ensures reliability in wireless transmission over a channel in state S_i is bounded above by*

$$R_i \leq 1 - \frac{\epsilon_i}{\alpha}. \ \ \epsilon_i \ll \alpha$$

where α is error-correcting capability of a decoder.

Proof. See appendix. □

6.2.2 Code Rate: Stability

Most Internet applications are subjected to various requirements for reliability and delay. For example, the quality degradation is significant in audio/video conferencing if data packets are not delivered in a timely fashion. The wide variety in traffic rate requirements leads to a large

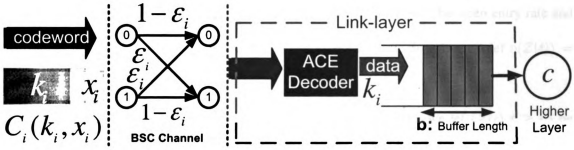


Figure 6.2 System model for stability analysis in wireless Communication.

variance in the traffic-volume injected into the network. This often leads to throughput instability. We define stability as follows:

Definition 4. *System is stable when higher layers are neither starved for information packets nor is there a glut of packets leading to buffer overflow.*

Figure 6.2 shows a queueing model for the system. The *consumption rate* c of the buffer represents the rate at which higher layers remove data symbols from the buffer. One of the important stability requirement is that the buffer has to be non-empty to avoid execution stalls. This property is satisfied when data arrival rate is high enough to satisfy the data consumption rate. Execution stalls refer to a condition where higher layers cannot continue execution because there is no data symbol available in the buffer, leading to system instability.

The fluctuations of the buffer growth can be captured by computing limiting distributions of the buffer length using a general model of fluid entering and leaving a single buffer. The input and output rates of the buffer depend on the external environment: let $Z(t)$ be the state of the external environment and $B(t)$ be the amount of fluid in the buffer at time t. In our framework, $Z(t)$ indicates the decoding outcome in the link-layer (later, we model $Z(t)$ as a two-state CTMC) and $B(t)$ is the number of data symbols in the buffer at time t. The dynamics of the buffer length is captured by a fluid process $B = \{B(t), t \geq 0\}$ (driven by $Z = \{Z(t), t \geq 0\}$ process) given by:

$$\frac{dB(t)}{dt} = \eta(Z(t)) \tag{6.5}$$

where $\eta(Z(t))$ is called the *drift function* which measures the difference between entry rate and exit rate at state $Z(t)$. To ensure that B process does not become negative, we let $\eta(Z(t)) = \max(\eta(Z(t)), 0)$ when $B(t) = 0$.

To calculate the limiting distributions of $B(t)$ with respect to $Z(t)$, we let $\{Z(t), t \geq 0\}$ be an irreducible CTMC on state space $S = \{1, 2, \cdots, M\}$ with generator matrix $Q = [q_{ij}]$. Correspondingly, $(B, Z) = \{(B(t), Z(t)), t \geq 0\}$ is a bivariate markov process with the limiting distribution defined as:

$$F(b, j) = \lim_{t \to +\infty} P(B(t) \leq b, Z(t) \leq j). \ 1 \leq j \leq M \tag{6.6}$$

By defining:

$$\begin{aligned} F(b) &= [F(b, 1), F(b, 2), \cdots, F(b, M)] \\ D &= diag[\eta(1), \eta(2), \cdots, \eta(M)]. \end{aligned}$$

where D is the drift matrix, Mitra in [97] shows that $F(b)$ satisfies the differential equation of a form

$$\frac{dF(b)}{dx} D = F(b)Q. \tag{6.7}$$

By solving the differential equation in (6.7), we can obtain the limiting distributions in equation (6.6).

The limiting distribution determines the likelihood of buffer length variations in every state of $Z(t)$. In our framework, we deploy these distributions to determine the lower bound on operational code embedding rate which ensures stability regardless of the state of $Z(t)$.

From the higher-layer viewpoint, $Z(t)$ the decoding process in the link-layer, can be expressed as an *on-off* fluid source model. Such a source stays *on* for an $exp(\omega)$ and stays *off* for an $exp(\beta)$ amount of time. It generates fluid at rate k_i when it is *on* and does not produce any fluid when it is *off*. Accordingly, a successful decoding at the link-layer indicates that the source is *on*. Using

the distortion model, with the channel in state S_i, the amount of time that the source is *on* is determined as

$$\omega_i = Pr(\text{Successful Decoding in } \tau_i) = F_{E_i}(\alpha x_i). \tag{6.8}$$

Correspondingly, the amount of time that the source is *off* is $\beta_i = 1 - \omega_i$.

Using the *on-off* source model, the environment process $Z = \{Z(t), t \geq 0\}$ in equation (6.5) is modeled as a two-state CTMC on state space $S = \{1 = \text{on}, 2 = \text{off}\}$ with the following generator matrix

$$Q = \begin{pmatrix} -\omega_i & \omega_i \\ \beta_i & -\beta_i \end{pmatrix}.$$

Recall that when the source in *on*, k_i data symbols enters the buffer while data symbols are removed from the buffer at the constant rate c regardless of the state of the source. Therefore, the drift matrix is given by

$$D = \begin{pmatrix} k_i - c & 0 \\ 0 & -c \end{pmatrix}$$

Using the matrices Q and D, we can solve the differential equation in (6.7). The final solution is given by

$$F(b,1) = \beta_i(1 - e^{\lambda b}) \tag{6.9}$$

$$F(b,2) = \omega_i - \frac{\beta_i(k_i - c)}{c} e^{\lambda b}, \tag{6.10}$$

where $\lambda = \frac{\beta_i}{c} - \frac{\omega_i}{k_i - c}$.

Based on the above derivations, $k_i\omega_i$ represent the expected number of data symbols injected into the buffer when the channel is in state S_i. Since the buffer has finite capacity, the following condition has to be satisfied to prevent buffer overflow

$$k_i\omega_i \leq c. \tag{6.11}$$

Using this model, we determine the lower bound on operational code embedding rate under ACE that satisfies the stability condition. This is given by the following Lemma:

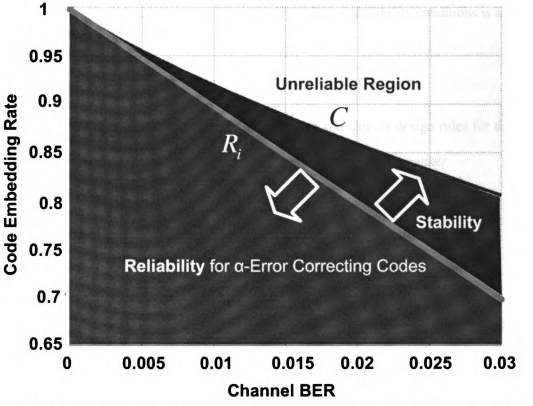


Figure 6.3 Operational code embedding rate domain with respect to reliability and stability.

Lemma 2. *The operational code embedding rate that ensures stability in wireless transmission over a channel in state S_i has a lower bound*

$$R_i \geq 1 - \frac{\epsilon_i}{\alpha}, \quad \epsilon_i \ll \alpha$$

Proof. See appendix. □

Lemma 2 suggests that for the channel in state S_i, stability condition is guaranteed for a variety of traffic demands using operational code embedding rate of at least $R_i = 1 - \frac{\epsilon_i}{\alpha}$. Meanwhile, Lemma 1 suggests that this rate is the upper bound of operational code rate which achieves reliability. As illustrated in Fig. 6.3, the domain of operational code rate is partitioned into two subdomains which intersect at $R_i = 1 - \frac{\epsilon_i}{\alpha}$. Further, this domain is bounded by the channel coding theorem [55] which requires the operational code embedding rate to operate below channel capacity of equation (3.11) ($R_i < C$). An important conclusion of this analysis is that an opti-

mal solution for code embedding rate that ensures reliability and stability conditions is a unique solution:

$$R_i^* = 1 - \frac{\epsilon_i}{\alpha}. \quad \epsilon_i \ll \alpha \tag{6.12}$$

This conclusion leads to simplistic, traffic independent and elegant design rules for the ACE protocol, while providing reliability and stability in an optimal and joint manner.

6.3 Automatic Code Embedding

In this section, we present the architecture design and implementation of ACE which uses the optimal code embedding rate deduced in the previous section for redundancy allocation and deploys channel side information to assess an accurate estimate of the channel condition in every transmission. To that end, we first describe a point-to-point communication model under which ACE is operating. Next, we present detailed functionality of ACE, specifically channel state estimation and redundancy allocation.

In this section, we describe ACE operational communication model. Here a *transmission interval* τ_i is expressed as the duration in which a transmitter sends the i^{th} message (packet) M_i and receives its corresponding acknowledgment ACK_i. A transmitter sends a new message after the reception of an acknowledgment.

Sender Side

During τ_i, a sender transmits a message which is represented by the tuple $M_i = (C_i(k_i, x_i), \mathbf{y}_i)$ where k_i represents the number of data symbols which are not being retransmitted. In each τ_i, a transmitter encodes k_i with parity symbols x_i creating a codeword $C_i(k_i, x_i)$. We refer to these parity symbols as type-I parity. The receiver utilizes x_i to decode C_i. Upon successful decoding, C_i is extracted and k_i data symbols are passed up to the higher layer. The error correction fails

when the decoding operation fails as indicated in FCS. In that case, the receiver stores C_i in its buffer and issues a request for more parity symbols. The transmitter also sends additional (type-II) parity symbols denoted by $\mathbf{y}_i$. The receiver utilizes $\mathbf{y}_i$ symbols to recover old corrupted codewords accumulated in its buffer (e.g., $C_j, j = 1, \cdots, i-1$).

Receiver Side

We assume that the receiver has a finite buffer which can accommodate up to m corrupted messages waiting for recovery. If a newly corrupted packet finds all rooms in the buffer occupied, it does not enter the buffer and is dropped. The status of the receiver is reported to the transmitter via certain flags in an acknowledgment message which are called *buffer flags*. Specifically, m flags are encapsulated in every acknowledgement. Let $F_i[k], k = 1, \cdots, m$ represent buffer flags in ACK_i. Each buffer flag is associated with a particular room in the buffer and represents the status of that room. That is, if the k^{th} room is occupied then $F_i[k] = 1$ (as illustrated in Fig. 6.4). In addition, the receiver estimate of channel condition $\hat{\delta}_i$ in τ_i is also encapsulated in acknowledgment message. Later, we describe channel estimation process by the receiver.

An example of ACE operational communication model is illustrated in Fig. 4.1. A short communication consisting of four transmission intervals and buffer capacity of two at the receiver is shown. During the first transmission interval τ_1, a message $M_1 = (C_1(k_1, x_1), \mathbf{y}_1)$ is sent. There are no type-II parity symbols in M_1, because there is no prior corrupted message in the receiver buffer, so $\mathbf{y}_1 = 0$. A receiver that fails to decode C_1, stores C_1 in its buffer and sends an acknowledgment $ACK_1 = (1, 0, \hat{\delta}_1)$. In τ_2, the transmitter sends $M_2 = (C_2(k_2, x_2), \mathbf{y}_2 = \{y_2^1\})$. The receiver uses x_2 to decode C_2 and employs type-II parity symbols y_2^1 (y_i^j denote additional parity for $C_j, j < i$ transmitted in τ_i) in addition to x_1 to decode C_1. The receiver acknowledges $ACK_2 = (1, 1, \hat{\delta}_2)$, indicating decoding failure of C_2 and $C_1(k_1, x_1 + y_2^1)$. As a result, in τ_3, the sender sends $M_3 = (C_3(k_3, x_3), \mathbf{y}_3 = \{y_3^1, y_3^2\})$. In τ_3, the receiver successfully decodes

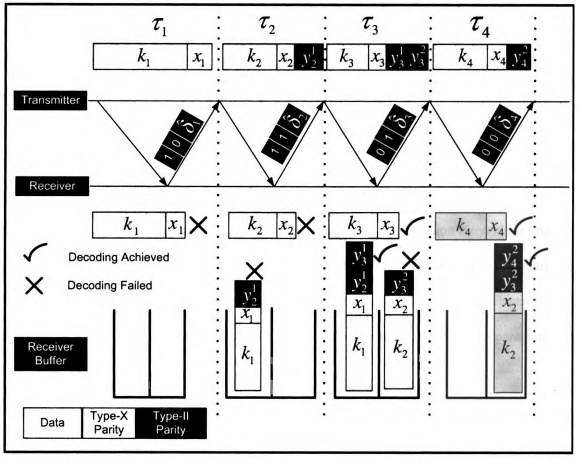


Figure 6.4 An example of ACE operational communication model consists of four transmission interval.

C_3 using x_3 and $C_1(k_1, x_1 + y_2^1 + y_3^1)$. Because decoding of C_1 was successful, the receiver sets the buffer flag of the first room to zero (i.e., $F_3[1] = 0$) but at the same time since the receiver is waiting for type-II parity symbols to perform decoding on C_2, the buffer flag for the second room is set to one (i.e., $F_3[2] = 1$); so $ACK_3 = (0, 1, \hat{\delta}_3)$. Accordingly, in τ_4, the sender transmits $M_4 = (C_4(k_4, x_4), \mathbf{y}_4 = \{y_4^2\})$. The receiver decodes C_4 using x_4 and $C_2 = (k_2, x_2 + y_3^2 + y_4^2)$ successfully; so $ACK_4 = (0, 0, \hat{\delta}_4)$.

The model described above represents ACE operational communication mechanism. In the following section, we describe the functionality of ACE protocol in estimating and adjusting the amount of necessary redundancy in every transmission under the delay constraint imposed by the application to achieve high performance.

6.3.1 ACE Protocol

ACE utilizes the receiver channel estimate and *buffer* flags to assess the status of the channel condition and the receiver buffer. Depending on this assessment, ACE determines the composition of the next message to be transmitted by the sender.

Channel State Estimation

Recall that the Markovian channel model implies that in every transmission interval the channel is in a particular state represented by a BSC with a unique BER. The objective is to train the Markovian channel model to achieve an accurate estimations of BER values for each state of the model. The training process is in an online fashion in the sense that ACE adjusts its parameters as more and more packets arrived during a session.

There are many ways to predict channel BER. One example is to use readily available information in the received packet. ACE uses Signal to Silence ratio (SSR) as side information in every transmission interval. Specifically, upon a reception of a packet in transmission interval τ_i, the receiver obtains the SSR and estimated BER values of packet preamble. We let ssr_i and $\hat{D}_i = g(ssr_i)$ represent the SSR value of packet M_i and its corresponding estimated BER respectively. A receiver creates a one-to-one mapping between each SSR value and each state of a Markov chain [i.e., $(ssr_1 \equiv S_1), \cdots, (ssr_N \equiv S_N)$]. It also keeps record of the observed BER values associated with each SSR, denoted by $(\Delta_i, i = 1, \cdots, N)$. Notice that the number of states of the channel model is dictated by the total number of unique SSR values observed by the receiver. The receiver training process is as follow:

1. Obtain ssr_i and $\hat{D}_i$ of the received packet in τ_i.

2. Find a state where $S_i \equiv ssr_i$.

3. Add $\hat{D}_i$ to the list of observed BER values associated with $(ssr_i \equiv S_i)$: $\Delta_i = \{\Delta_i, \hat{D}_i\}$.

4. Adjust the BER estimation associated with state S_i by taking the average value of the updated Δ_i

$$\hat{\delta}_i = \frac{\sum_{k=1}^{|\Delta_i|} \hat{D}_k \in \Delta_i}{|\Delta_i|}$$

In every transmission interval, the receiver adjusts the parameters of the channel model and sends its estimate of the current channel condition in an acknowledgment message.

Redundancy Allocation

In τ_i, ACE uses channel estimate of the previous transmission interval $\hat{\delta}_{i-1}$ along with *buffer* flags in ACK_{i-1} to allocate data and parity symbols for M_i. Specifically, ACE uses $\hat{\delta}_{i-1}$ as its estimate of the channel BER in current transmission interval τ_i; so $\hat{\epsilon}_i = \hat{\delta}_{i-1}$.

According to the communication model each buffer flag in the acknowledgment message ($F_{i-1}[j]$ $j = 1, \cdots, m$) indicates the status of a particular room in the receiver. ACE first allocates the amount of type-II parity symbols (y_i) necessary to transmit based on the buffer flags. If $F_{i-1}[j] = 0$, no parity symbol is necessary. However, $F_{i-1}[j] = 1$ indicates that room j contains a particular codeword C_k transmitted in some $\tau_k, k < i$ which requires additional redundancy symbols for recovery. According to optimal code embedding rate obtained in Section 6.2, C_k with length n_k requires a at most $T_i^k = \frac{n_k \hat{\epsilon}_i}{\alpha}$ parity symbols for error recovery. However, for C_k, $x_k + \sum_{l=k+1}^{i-1} y_l^k$ parity symbols are already transmitted in the previous transmission intervals $\tau_k, \cdots, \tau_{i-1}$. Thus, type-II parity symbols necessary to transmit in τ_i for C_k is $y_i^k = T_i^k - x_k + \sum_{l=k+1}^{i-1} y_l^k$.

ACE protocol requires that each message must have a fixed length n. After allocating type-II parity symbols for corrupted codewords, ACE has $n_i = n - \sum_k y_i^k$ symbols to transmit a new codeword $C_i(k_i, x_i)$. The amount of parity symbols necessary for encoding C_i is $x_i = \frac{n_i \hat{\epsilon}_i}{\alpha}$, and therefore the amount of *new* data in C_i is $k_i = n_i - x_i$.

The transmitter constructs a message M_i according to the message distribution specified by ACE. This distribution is adaptively computed in every transmission interval based on the channel and the receiver buffer conditions while it guarantees to satisfy the stability and reliability conditions. Therefore, the overall performance increases when the ACE protocol is utilized. In the next section, we conduct extensive performance evaluations of ACE through simulations under varying operating requirements.

6.4 Performance Evaluation

In this section, we present extensive performance evaluation of the ACE protocol using real channel traces collected on an 802.11b WLAN. First, we compare the performance ACE protocol as opposed to the conventional IEEE802.11 ARQ protocol on *realtime* and *non-realtime* traffic. Second, we analyze the impact of deploying ACE in the link-layer on the traditional TCP throughput. Finally, we illustrate the quality of real-time video communication in terms of PSNR gain under ACE and IEEE802.11 ARQ.

6.4.1 Realtime Traffic

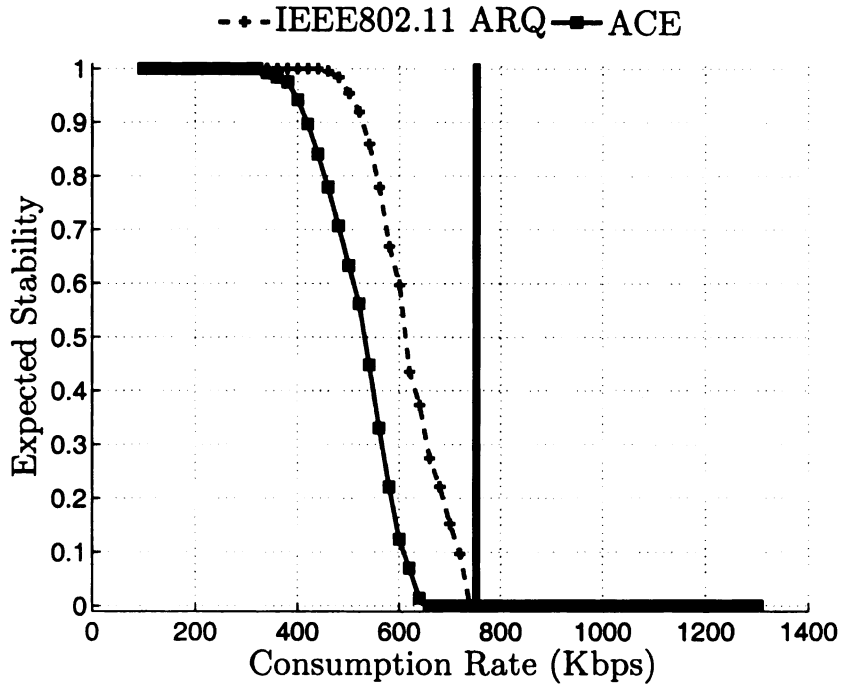
An important aspect of realtime traffic delivery is to prevent instability (by delivering information in timely fashion) while ensuring the required reliability. To determine the impact of deploying ACE protocol on stability for *realtime* traffic, we introduce the following parameters:

- *Consumption rate:* This parameter determines minimum number of data symbols required per second to ensure stability for realtime traffic. This rate is determined by the delay constraint imposed on the realtime traffic. During a particular session, the rate of correct information delivery must satisfy the consumption rate to guarantee stability.

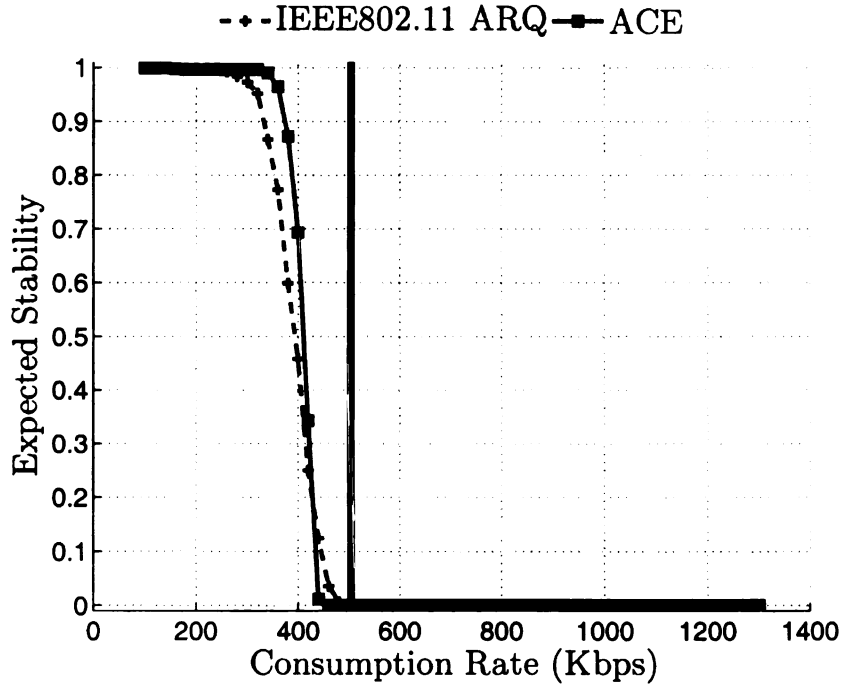
- *Expected stability:* This parameter measures the time period during a particular session that the traffic delivery is stable (see definition of stability in section 6.2.2).

For lower *consumption rate*, it is more likely that the traffic will be delivered to the upper layers on time and therefore avoid instability. However, as the *consumption rate* increases the likelihood that the upper-layers do not receive the packets in a timely fashion increases. Ultimately, if the *consumption rate* exceeds the sender transmission rate, then the traffic never reaches the destination on time and therefore the system becomes constantly unstable. The *expected stability* declines significantly as the *consumption rate* reaches the sender transmission rate.

To measure impact of the ACE protocol on *expected stability*, we use network simulator OMNET++ [105] to incorporate the ACE protocol in the current IEEE802.11 MAC layer. Specifically, we utilize the `INET Framework` package and modified the link layer (`Ieee802.11Nic` module) to add the ACE protocol. We also incorporated A-LDPC [104] into the software for embedded coding operations. The simulation setup is as follow: we let a sender transmits a codeword. We use our channel traces to distort the transmitted codeword by flipping distorted bits. The receiver uses A-LDPC to decode the received word and checks whether the decoded word is a valid codeword. Upon decoding failure, a receiver stores the received word in its buffer and requests for additional redundancy. We use the LDPC source code provided in [104] for our experiment. Note that we use a soft decision decoding using an iterative belief propagation method which requires a knowledge of channel BER. So, ACE uses its estimate for channel BER for every transmission interval described in Section 6.3.1 to decode each packet. For this experiment, the maximum iteration is 100; the variable side has degree three and the check side degree is approximately regular. Upon successful decoding of a packet, the header of the packet is extracted and the payload is sent to the upper-layers. To simulate the delivery of realtime traffic, the *consumption rate* is set at a range of 100 to 1300Kbps.



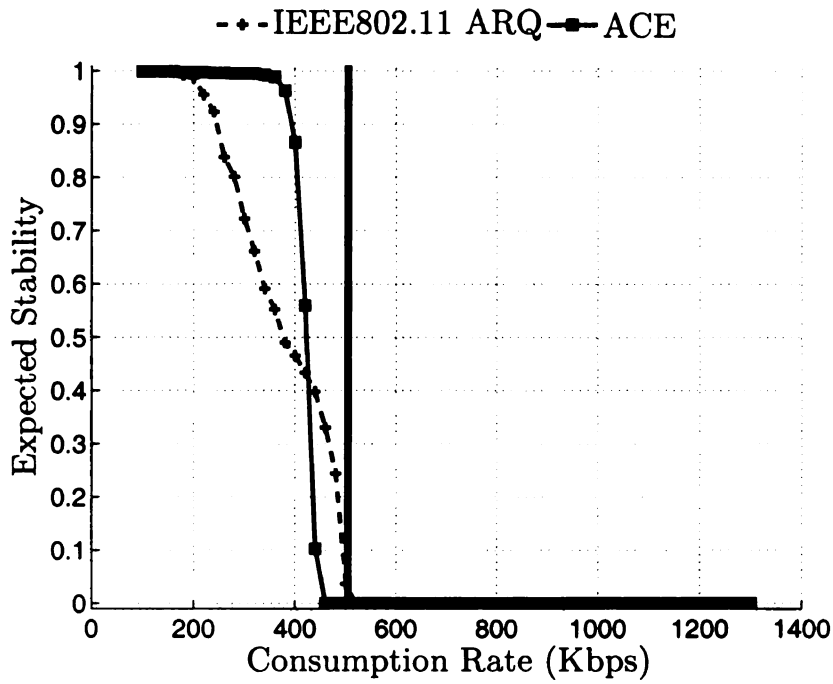
(a) BER: 0.001



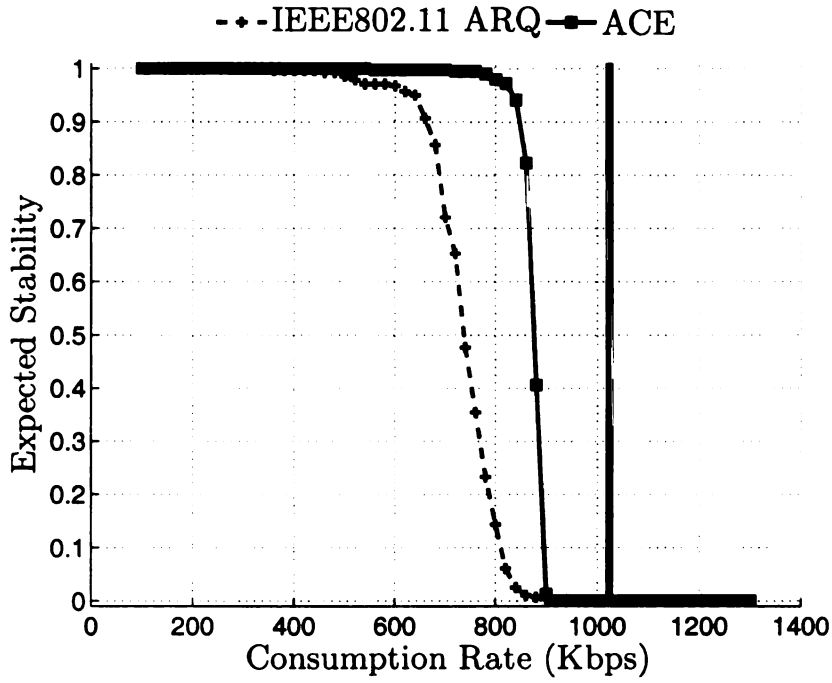
(b) BER: 0.005

Figure 6.5 Expected stability of realtime traffic with respect to variation of consumption rate over different channel traces. The vertical line in each figure represents the sender transmission rate.

Fig. 6.5 illustrates the variation of *expected stability* over various channel traces. We observe that the IEEE802.11 ARQ performs slightly better that ACE over channels with low BERs (less



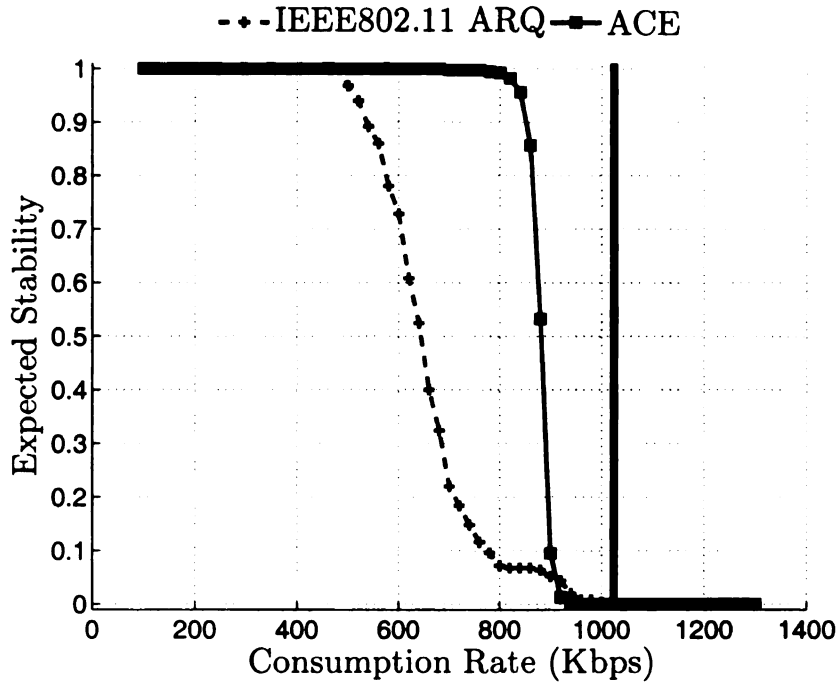
(c) BER: 0.007



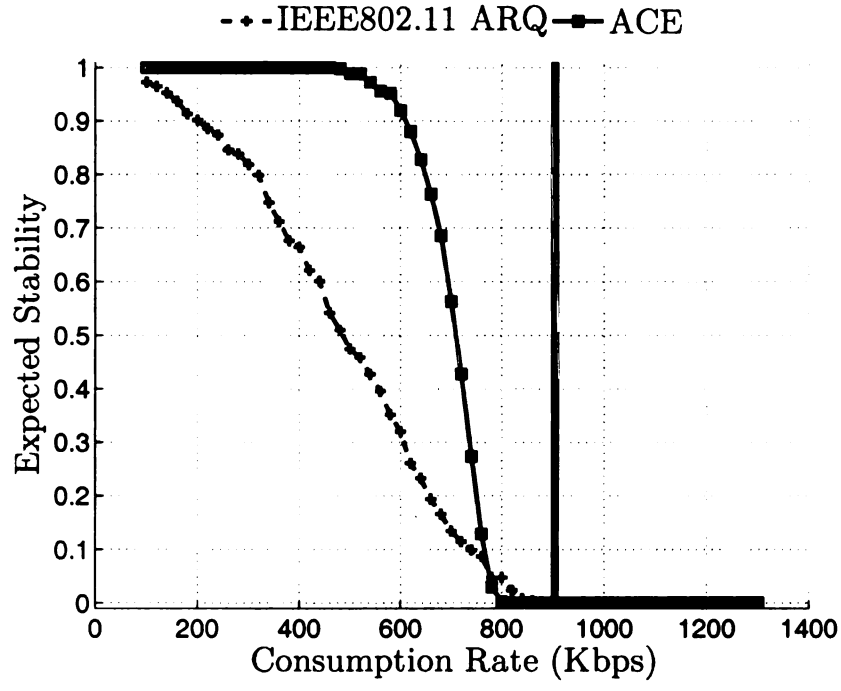
(d) BER: 0.009

Figure 6.5 continued.

than 0.002). This result is expected since for channels with low BER the likelihood of corruption is very low and a simplistic ARQ mechanism is sufficient for error recovery. Over channels



(e) BER: 0.018



(f) BER: 0.020

Figure 6.5 continued.

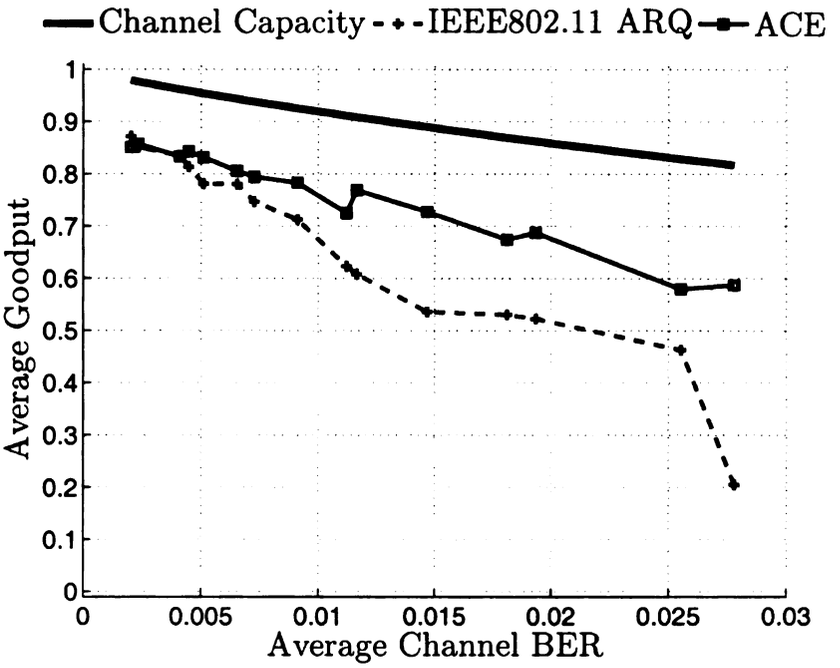
with BER in a range of 0.002 to 0.007, both protocols produce similar performances, however ACE outperforms IEEE802.11 ARQ for higher consumption rates. When the channel BER ex-

ceeds 0.007, the ACE performance is significantly better than IEEE802.11 ARQ performance. For instance, ACE guarantees 100% expected stability for consumption rates up to 800Kbps over channel with BER 0.009 while the traffic delivery is 80% *unstable* under IEEE802.11 ARQ at this rate. Further, over noisy channels with BER more than 0.18, we observe a drastic drop in expected stability under IEEE802.11 ARQ as the consumption rate exceeds 300Kbps meanwhile ACE maintains stability for source rates as high as 700Kbps. In this figure, the vertical line represent the sender transmission rate. We observe that expected stability drops to zero (constant instability) when the consumption rate exceeds the sender transmission rate.

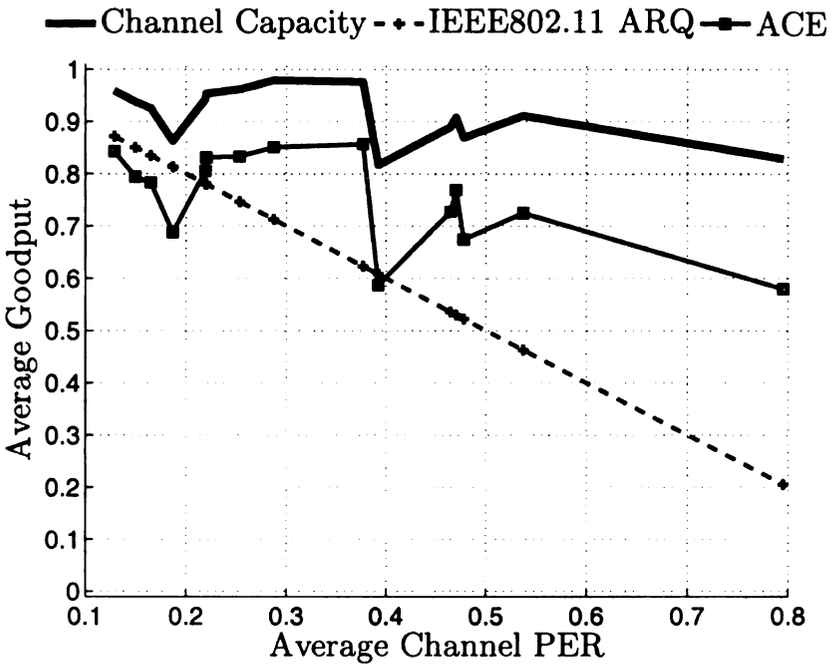
6.4.2 Non-Realtime Traffic

The main objective in *non-realtime* traffic delivery is to maximize the bandwidth utilization of the wireless medium "per channel use" while providing essential reliability. We compare the performances of ACE and IEEE802.11 ARQ under different channel conditions. The performance is in terms of *average goodput* which measures the average number of *new data* symbols that are delivered *correctly* to the destination per channel use. So, average goodput closer to one is an indication that the protocol utilizes the channel more efficiently.

Fig 6.6 illustrates the average goodput achieved by ACE and IEEE802.11ARQ over variety of channel conditions. In Fig. 6.6a we observe that IEEE802.11 ARQ and ACE performances are almost identical when the BER is small (i.e., below 0.005). As the BER increases the decline in average goodput becomes more rapid for IEEE802.11 ARQ than ACE. For example, over traces with BER ranging from 0.015 to 0.02, IEEE802.11 ARQ performance is around 50% while ACE hovers around the 70% mark. We also observe that average goodput under IEEE802.11 ARQ declines dramatically as channel Packet Error Rate (PER) increases as seen in Fig. 6.6b. Overall, this result shows 10% to 30% improvement in average goodput under ACE for non-realtime traffic communication. Also, it is observed than ACE in general operates closer to channel capacity than



(a) Channel condition (Average BER)



(b) Channel condition (Average PER)

Figure 6.6 The average goodput of ACE and IEEE802.11 ARQ over various channel conditions. Note that channel capacity in each figure represents the maximum amount of achievable goodput without errors.

IEEE802.11 ARQ.

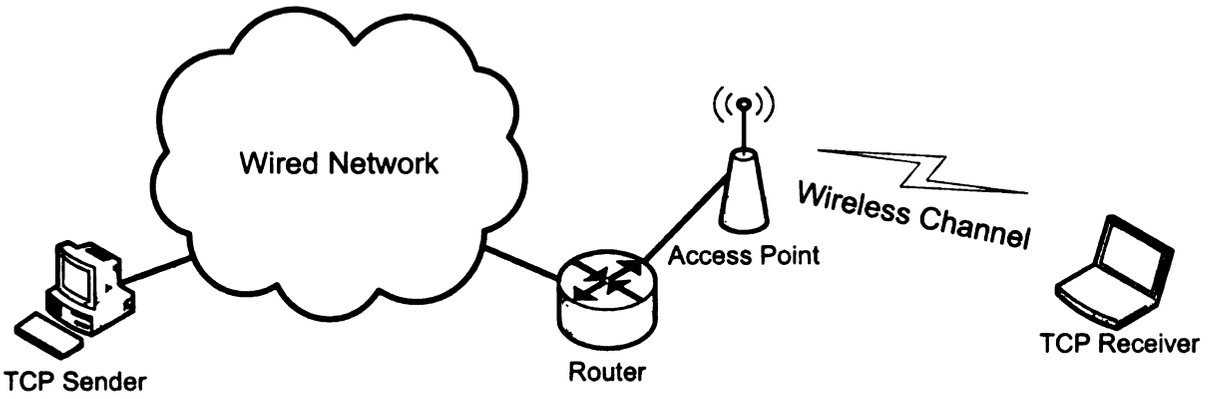


Figure 6.7 Heterogenous network model.

6.5 Throughput analysis of TCP

The Jacobson's adaptive window flow control algorithm is common for most TCP variations [83]: let $W(t_k)$ be TCP sender congestion window width at time instant t_k and $W_{th}(t_k)$ be a slow-start threshold. TCP sender operates at *slow start* phase if $W(t_k) < W_{th}(t_k)$. In this phase, each ACK causes $W(t_k)$ to be incremented by one. When $W(t_k)$ exceeds slow-start threshold, the TCP sender enters *congestion avoidance* phase where each ACK increments $W(t_k)$ by $1/W(t_k)$. TCP sender exits the congestion avoidance when the timeout occurs. In this case, the congestion window is set to one and $W_{th}(t_k^+)$ is set to $\lceil W(t_k)/2 \rceil$.

This simple algorithm is well established for congestion control in wired network. However, due to lossy environment of wireless links, TCP congestion control suffers from performance degradation since packet losses in wireless links are interpreted as congestion by the TCP agent. Because leading link-layer protocols focus primarily on reliability and ignore the stability aspect of wireless communication, numerous studies have led to vast variety of TCP congestion control algorithms [84] in attempt to fix TCP over-wireless performance degradation phenomenon. We argue that since ACE is designed to guarantee stability as well as reliability at the link-layer, the traditional TCP congestion control algorithm should perform relatively well over wireless link despite the lossy environment.

6.5.1 Network Model

We consider a heterogeneous network model consisting of wired and wireless sections depicted in Fig. 6.7. A TCP sender located within a wired section of the network is connected to a TCP receiver placed in a wireless section. An access point (AP) connected to the wired section, receives transmitted packets and sends them over a contention-free wireless channel. The wired network comprises multiple links connected through different routers. In our analysis, we model such network as a single link with average capacity of c packet-per-second and a bottleneck router buffer with finite capacity. A particular packet traversing the wired section is stored in the router bottleneck buffer as well as AP buffer before it enters the wireless section.

In our network model, a packet loss occurs under the following scenarios: (1) *Congestion-based loss*: A transmitted packet is dropped at the wired section due to buffer overflow of the bottleneck router. (2) *AP-based loss*: A transmitted packet which successfully crossed the wired section never enters the wireless channel and is rather dropped at AP. This kind of loss is due to the *instability* of the link-layer which causes AP buffer overflow. (3) *distortion-based loss*: A broadcasted packet over the wireless channel gets corrupted due to wireless noise. This packet is not retrieved due to link-layer *unreliability* and is reported as lost to TCP agent.

6.5.2 System Model under ACE

In this section, we perform extensive analysis on our network scenario to determine steady state loss observed at the TCP sender. We assume that ACE is an underlying error combating scheme in wireless MAC layer.

According to our network model, a particular packet traverses multiple queues before it is sent over the wireless channel. These queues represent the bottleneck router buffer at the wired section and also the AP buffer. In addition, under ACE, if a decoding of corrupted packet fails, it is

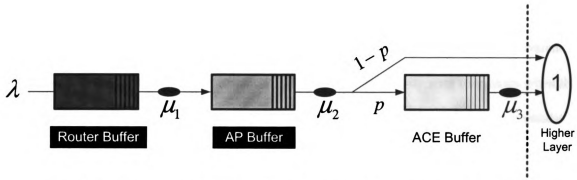


Figure 6.8 A system of queuing network for the heterogenous network under ACE protocol.

stored at the receiver buffer for future recovery. In this system, since all three buffers have limited capacity, a packet loss occurs if any of these buffers are full. From the TCP sender perspective, the cause of packet loss is unknown. That is, TCP sender is unable to identify the location of a loss in the network. In addition, the behavior of one queue impacts the behavior of other queues. For instance if the AP buffer is full, packets in the router buffer have to wait before they are transmitted to the AP. So, the assumption that the loss process in these queues are independent is rather naive and one has to analyze the joint behavior of these queues to capture more accurate estimate of loss behavior. To that end, we model our network scenario using a system of queueing network depicted in Fig. 6.8. In this system, $Q_i, \quad i = 1, 2, 3$ with a limited capacity of $B_i, \quad i = 1, 2, 3$ represent the router, AP and ACE buffers respectively.

To analyze the behavior of this system, we let $X_i(t)$ denote the number of packets arrive at the $Q_i, \quad i = 1, 2, 3$ at time t. Then, $\mathbf{X}(t) = [X_1(t), X_2(t), X_3(t)]$ is a Markov Chain whose transition rates are $\lambda, \mu_1, p\mu_2, q\mu_2$ and μ_3 where $q\mu_2$ and μ_3 measure the departure rate of a packet from the system. Note that p measures the likelihood that the received packet is not decoded successfully using type-I parity bits; $q = 1 - p$. To determine the steady state loss probability in this system, we derive the stationary distribution of the process. Let

$$\pi_{m,n,l} = \lim_{t \to +\infty} Pr\{X_1(t) = m, X_2(t) = n, X_3(t) = l\}$$

denote the stationary distribution of $\mathbf{X}(t)$. By examining particular states of the Markov chain,

we drive the balance equations for the process stationary distribution. The intuition behind these derivations is that in a steady state analysis, the outgoing rate from particular state is equivalent to the incoming rate to that state. For instance, Fig. 6.9 illustrates steady state transitions when the system in in state (m, B_2, B_3). It shows the system status where the router buffer accommodated m packets and the AP and ACE buffers are full and drop any incoming packets. The system enters to this state under the following scenarios: (1) there was $m-1$ packets in Q_1 and a new packet arrives, (2) there was $m+1$ and B_2-1 packets in Q_1 and Q_2, and a packet is transmitted from router to AP. On the other hand, a system exits state (m, B_2, B_3) if: (1) any incoming packet arrives so there are $m+1$ packets in Q_1, (2) a receiver successfully decodes a transmitted packet from the AP, (3) ACE successfully decodes a corrupted packet using type-II parity bits. By applying the above scenarios, the steady state equation for

$$\pi_{m,B_1,B_2} = \lim_{t\to+\infty} Pr\{X_1(t) = m, X_2(t) = B_1, X_3(t) = B_3\}$$

is

$$(\lambda+q\mu_2+\mu_3)\pi_{m,B_2,B_3} = \lambda\pi_{m-1,B_2,B_3}+\mu_1\pi_{m+1,B_2-1,B_3}. \quad 1 \le m \le B_1-1 \quad (6.13)$$

By applying this methodology, we derive the balance equations for all possible states of the queueing system. The complete list of these equations are given in Appendix. To determine $\pi_{m,n,l}$, we solve the balance equations of steady state distribution as a system of linear equations. Let $\Pi = (\pi_{1,1,1}, \cdots, \pi_{m,n,l}, \cdots, \pi_{B_1,B_2,B_3})$ represent a vector of unknown variables and A is a $L \times L$ coefficient matrix with $L = B_1 \times B_2 \times B_3$. Then, the steady state equations can be presented as

$$\mathbf{A}\Pi = \vec{0}. \quad (6.14)$$

Using Gaussian-Jordan method, we are able to determine Π and therefore $\pi_{m,n,l}$.

The steady state probability distribution enables us to perform thorough analysis of the proposed queueing system. However, this probability distribution itself is a function of transition rates of

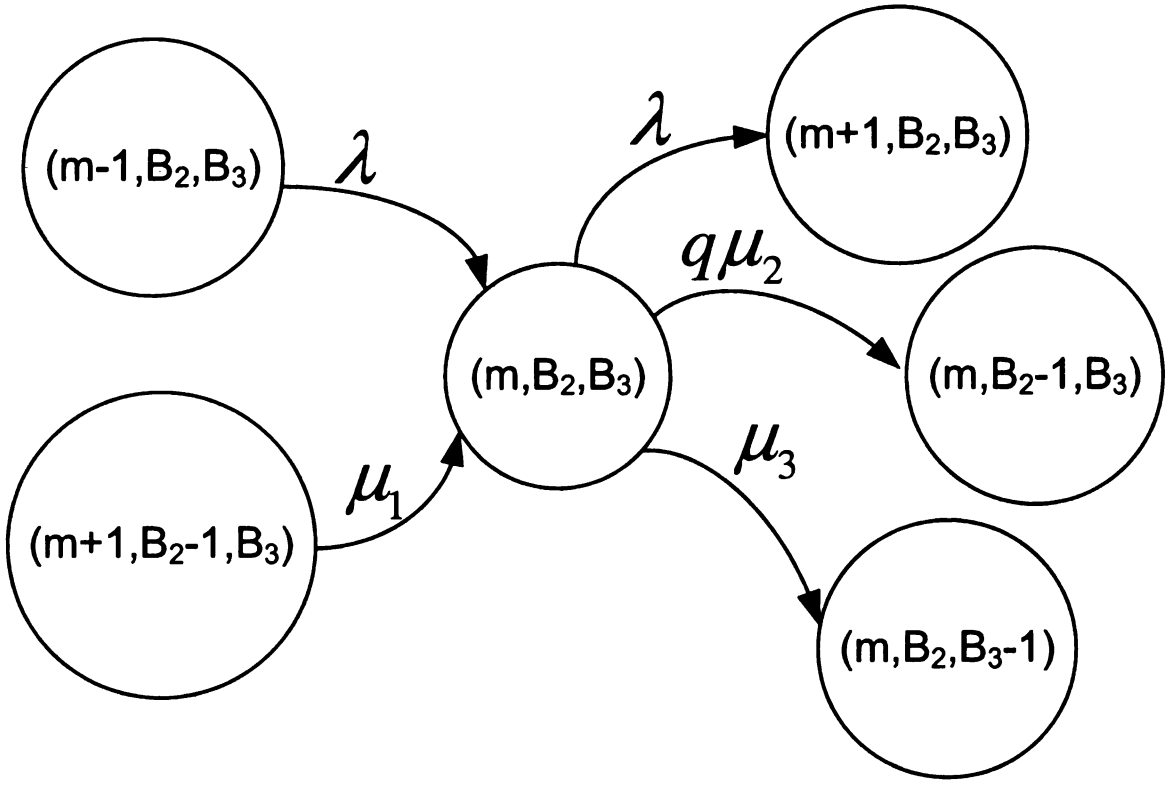


Figure 6.9 The state diagram of system dynamics where $|Q_1| = m, |Q_2| = B_2, |Q_3| = B_3$.

the process. Accordingly, it is necessary to compute the transition rates of a Markov Chain. In the following section we determine the arrival and service rates of the queueing system namely λ and $\mu_i, i = 1, 2, 3$ as well as parameter p.

Arrival and Service rates

In our analysis, we model a wired section of our network scenario as a single wired link with average capacity of c packet-per-second and a bottleneck router buffer with size B_1. Under this model, the arrival rate of the system is equivalent to the transmission rate of a TCP sender $\chi(t) = W(t)/T$ where $W(t)$ defines the maximum number of unacknowledged data outstanding in the network at the sampling time instant t and T represents the round-trip time (RTT) which refers to the amount of time that elapses between the instant that the sender transmits a packet and the instant at which it receives the acknowledgment (ACK) for that packet. We assume that the ACK's

are cumulative and error-free. Thus,

$$\lambda = \chi(t) \tag{6.15}$$

$$\mu_1 = c. \tag{6.16}$$

To compute the service rate of the AP buffer μ_2, it is important to consider the underlying error control scheme utilized by AP. Under ACE, in every transmission interval τ_i, AP transmits a new packet. Further, additional parity bits are transmitted with a new packet upon of receiver request. Thereby, in steady state, the AP buffer transmits a new packet at the every transmission interval. Thus

$$\mu_2 = E[\tau]. \tag{6.17}$$

The proportion of packets that enters the ACE buffer Q_3 is characterized by parameter p. Any particular packet transmitted over the wireless channel enters Q_3 when its decoding with type-I parity symbol fails. Therefore, finding p is equivalent to computing the expected rate of decoding failure using type-I parity bits. To that end, we let D_i represent the distortion distribution of a particular packet transmitted when a channel is in state S_i with BER ϵ_i and X_i represent the distribution of type-I parity bits. Then, the event $A_i = \{D_i > \alpha X_i\}$ indicates the decoding failure of a packet using type-I parity bits where α is an error-correcting capability of a decoder. By applying the statistical models developed in Chapter 6, we have

$$Pr(A_i) = 1 - e^{-n(\epsilon_i + p_{X_i})} \sum_{x=0}^{n} \sum_{r=0}^{\lfloor \alpha x \rfloor} \frac{n^{r+x} \epsilon_i^r p_{X_i}^x}{r!x!}. \tag{6.18}$$

According to the Markovian channel model introduced in Chapter 3, the channel is in state S_i with the steady state probability of π_i, therefore the overall probability of decoding failure is as following

$$p = E_i[A] = \sum_{i=1}^{N+1} \pi_i Pr(A_i). \tag{6.19}$$

Any particular packet leaves the ACE buffer Q_3 when its decoding using its original type-I parity with additional type-II parity bits is performed successfully. So, in the steady state analysis, the service rate of ACE buffer is equivalent to the expected likelihood of successful decoding using type-I and type-II parity bits. Let Y_i represent the type-II parity distribution of a packet when the channel is in state S_i. Then, the event $T_i = \{D_i \leq \alpha(X_i + Y_i)\}$ indicates the successful decoding of a packet in ACE buffer. Using the distortion model developed in Chapter 4, we have

$$Pr(T_i) = e^{-n(\epsilon_i + p_{Z_i})} \sum_{z=0}^{n} \sum_{r=0}^{\lfloor \alpha z \rfloor} \frac{n^{r+z} \epsilon_i^r p_{Z_i}^z}{r!z!}, \tag{6.20}$$

where $Z_i = X_i + Y_i$. Thus, the service rate of Q_3 is

$$\mu_3 = E_i[T] = \sum_{i=1}^{N+1} \pi_i Pr(T_i). \tag{6.21}$$

The derivations of transitions rates and the steady state probability distributions completely characterize the dynamics of the proposed queueing system. The average time that a particular packet spends in the system is

$$E[W] = \sum_{i=1}^{2} \left(E[W_i] + \frac{1}{\mu_i} \right) + p \left(E[W_3] + \frac{1}{\mu_3} \right). \tag{6.22}$$

where $E[W_i]$ is the average time that a packet spends in Q_i is as following

$$E[W_i] = \frac{1}{\lambda_i} \left(\frac{a_i}{1 - a_i} - \frac{(B_i + 1) a_i^{B_i+1}}{1 - a_i^{B_i+1}} \right), \tag{6.23}$$

where $a_i = \frac{\lambda_i}{\mu_i}$.

In this system, a loss occurs if a particular packet in the system is blocked by one the queues. Correspondingly, the steady state loss probability is

$$\pi_{Loss} = \frac{1}{E[W]} \left(E[W_1] \pi_{B_1} + E[W_2] \pi_{B_2} + E[W_3] \pi_{B_3} \right), \tag{6.24}$$

where

$$\pi_{B_1} = \sum_{n=1}^{B_2} \sum_{l=1}^{B_3} \pi_{B_1,m,l} \tag{6.25}$$

$$\pi_{B_2} = \sum_{m=1}^{B_1} \sum_{l=1}^{B_3} \pi_{m,B_2,l} \tag{6.26}$$

$$\pi_{B_3} = \sum_{m=1}^{B_1} \sum_{n=1}^{B_2} \pi_{m,n,B_3}. \tag{6.27}$$

In this section, we presented a queueing system that captures the behavior of packet flow in our network scenario. We determined the steady state packet loss of this system when ACE is used as an underlying error control protocol in wireless link layer. In the next section, we model our network scenario under IEEE802.11 ARQ scheme and we drive the steady state loss under this protocol. In the sequel, we will comparatively analyze the impact of different parameters on the loss behavior under ACE and IEEE802.11 ARQ schemes and finally we observe the impact of loss on the overall TCP throughput.

6.5.3 System Model under IEEE802.11 ARQ

IEEE802.11 is a de facto wireless MAC protocol designed to ensure reliable transmission in wireless channel: it incorporates frame check sequence (FCS) to detect errors and automatic repeat request (ARQ) to retransmit corrupted packets. The IEEE 802.11 link layer discards corrupted packets with no regard to the number and location of the errors. The IEEE802.11 ARQ mechanism request for a retransmission of a corrupted packet until it reaches the maximum retransmission limit (MTL). If an error-free copy of a packet is not received by then, it discards the packet and labels it as lost.

Fig. 6.10 illustrates queuing system for our network scenario under IEEE802.11. Notice that unlike our model for ACE, this system comprises only two queues which represent the router and AP buffer. Similarly, to derive the steady state loss in this system, we should compute the

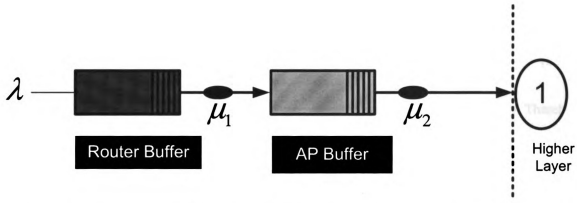


Figure 6.10 A system of queuing network for the heterogenous network under IEEE802.11 ARQ protocol.

joint steady distribution of Q_1 and Q_2. Let $X_i(t)$ denote the number of packets arrive at the Q_i, $i = 1, 2$ at time t. Then, $\mathbf{X}(t) = [X_1(t), X_2(t)]$ is a Markov Chain whose transition rates are λ, μ_1 and μ_2. The stationary distribution of $\mathbf{X}(t)$,

$$\pi_{m,n} = \lim_{t \to +\infty} Pr\{X_1(t) = m, X_2(t) = n\},$$

is computed by forming the balance equations for different states of $\mathbf{X}(t)$. The complete list of balance equation of this system is presented in Appendix.

Before solving the steady state balance equations it is necessary to compute the arrival and service rates in the queueing system. The arrival and service rate of Q_1 is given in Equation 6.15 since Q_1 represents the router buffer which its rates are characterized by wired network and is identical in both models. However, the service rate of AP is a function of IEEE802.11 ARQ mechanism. Let d represent the tolerable delay of every transmitted packet at the receiver side. According to IEEE802.11 mechanism, this delay should be at most MTL. (i.e., $d \leq MTL$). We let event Υ represent a successful delivery of a particular packet and recall ς is a packet error rate in every transmission interval under our wireless channel model. Under IEEE802.11, a particular packet is successfully delivered if it is errorless in the first transmission or in any of consecutive

$d-1$ retransmissions. Thus the expected number of successful delivery is

$$E[\Upsilon] = \sum_{i=1}^{d} \varsigma^{i-1}(1-\varsigma). \tag{6.28}$$

The service rate of AP buffer is equivalent to the expected rate of successful delivery. Thereby,

$$\mu_2 = E\left[E[S|\Upsilon]\right] = \sum_{i=1}^{d} \frac{\varsigma^{i-1}(1-\varsigma)}{iE[\tau]}. \tag{6.29}$$

Once the transition rates are computed, we form a system of linear equations similar to Equation (6.14) where $\Pi = (\pi_{1,1}, \cdots, \pi_{m,n}, \pi_{B_1,B_2})$ and $\mathbf{A}$ is a $L \times L$ coefficient matrix with $L = B_1 \times B_2$ constructed by transitions rates given in Equations (6.15) and (6.29).

By solving this system of equations, we can determine the steady state distribution presented in a vector Π. Accordingly, the average time that a particular packet is in the system is

$$E[W_{ieee}] = \sum_{i=1}^{2} \left(E[W_i] + \frac{1}{\mu_i} \right) \tag{6.30}$$

where $E[W_i]$ is the average time that a packet spends in Q_i and is computed using Equation (6.23).

The steady state loss probability captures the proportion of time that a particular packet is blocked by any of the queues in the system. Thus

$$\pi_{Loss} = \frac{1}{E[W_{ieee}]} \left(E[W_1]\pi_{B_1} + E[W_2]\pi_{B_2} \right)., \tag{6.31}$$

where

$$\pi_{B_1} = \sum_{n=1}^{B_2} \pi_{B_1,n} \tag{6.32}$$

$$\pi_{B_2} = \sum_{m=1}^{B_1} \pi_{m,B_2}. \tag{6.33}$$

6.5.4 Analytical Performance Evaluations

In this section, we comparatively analyze the impact of ACE and IEEE802.11 ARQ scheme on the overall performance of our network model. We measure the performance in terms of steady state

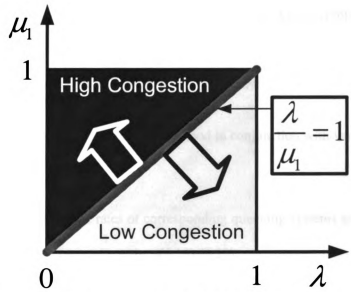


Figure 6.11 The state space of traffic intensity in wired network.

throughput π_τ which is the fraction of packets successfully transmitted to the receiver: $\pi_\tau = 1 - \pi_{loss}$. In the proposed network model, two critical factors overshadow the overall performance: (1) traffic intensity in wired network, (2) the overall condition of wireless channel. The traffic intensity in the wired section dictated by the TCP sender transmission rate and the wired link average capacity (i.e., λ and μ_1) governs the existence of congestion-based loss, where as the channel condition imposes the level of corruption in the transmitted packet.

Congestion Likelihood Classification

Let us assume that the transmission rates and link average capacity are normalized ($0 \leq \lambda, \mu_1 \leq 1$). Figure 6.11 shows the state space of traffic intensity in wired network based on the values of λ and μ_1. According to this space, the likelihood of congestion occurrence in wired network can be categorized in following regions: (1) *Low congestion:* where $\frac{\lambda}{\mu_1} \leq 1$. (2) *High congestion:* where $\frac{\lambda}{\mu_1} > 1$.

The congestion likelihood region specifies traffic condition in wired section of the network. To capture the impact of wireless channel condition on overall performance we use our channel

model to simulate various wireless channels. We use Equation (3.8) to compute the steady state average BER and PER of channel.

Our analytical performance evaluations steps are as follows:

1. We consider each class of congestion likelihood in conjunction with various wireless channels.

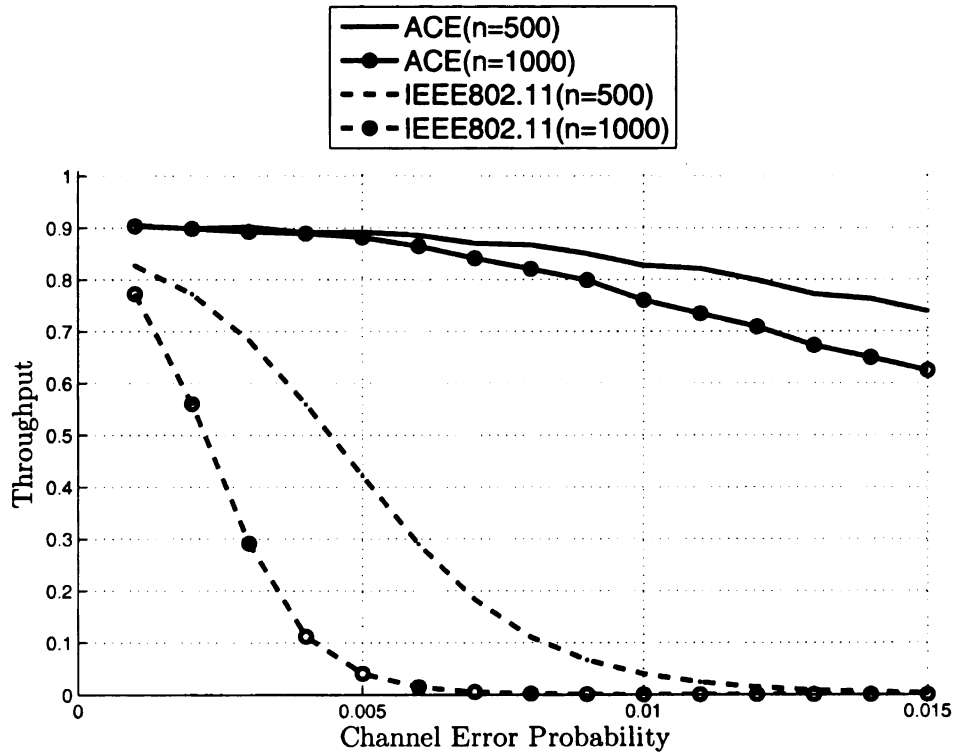
2. We compute the transitions rates of corresponding queueing systems associated with ACE and IEEE802.11 ARQ using Equations(6.15)-(6.29).

3. We utilize these rates to construct the balance equations for each queueing model (see Appendix).

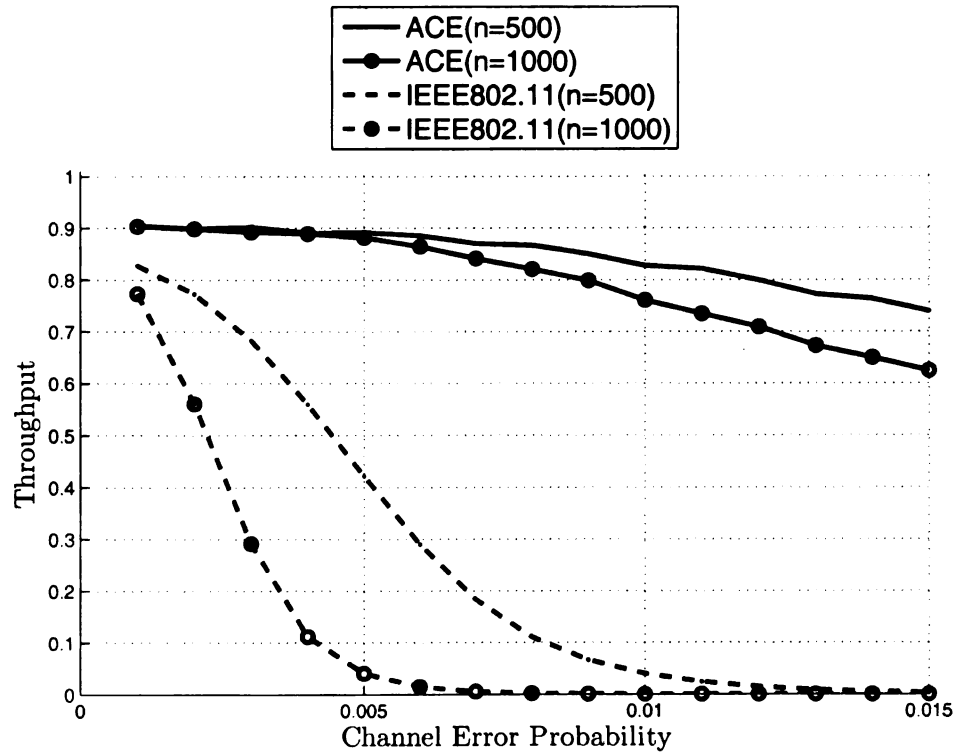
4. We apply Gauss-Jordan method to solve the balance equations to evaluate the joint steady distribution function of each model.

5. Using Equations (6.24) and (6.31), we evaluate the steady state loss likelihood under each model.

This process is repeated for different combinations of congestion likelihoods and channel conditions. Fig. 6.12 illustrates the variations in steady state throughput in different congestion likelihood regions. In this figure, the throughput is depicted for packet sizes of length $n = 500$ and $n = 1000$. We observe that regardless of underlying wireless error control protocol, the overall throughput decays when wired network becomes more congested. However, in each of congestion conditions in the wired network, ACE introduces higher throughput in comparison with the IEEE802.11 ARQ scheme as the wireless channel BER increases. Further, it is observed that IEEE802.11 ARQ throughput is very sensitive to the size variations of transmitted packets. This phenomena stems from the fact that the overall PER of a channel which has a direct impact in IEEE802.11 ARQ service rate increases as the size of the packet increases. However, since ACE



(a) Low congestion



(b) High congestion

Figure 6.12 The steady throughput of network model using ACE and IEEE802.11 ARQ over the various channel traces with respect to different congestion likelihoods.

uses parity instead of retransmission to recover errors, the overall throughput is a function of channel BER. So, although an increase in packet size would decrease a throughput, this decay is not significant as in case of IEEE802.11 ARQ scheme.

The expected congestion window variation in congestion avoidance is,

$$E[\Delta W] = \frac{\chi(t_k - T)(1 - \delta(t_k))}{W(t_k)} - \frac{1}{2}\chi(t_k - T)\delta(t_k)W(t_k). \tag{6.34}$$

Upon simplification, the equilibrium throughput is

$$\hat{\chi} = \sqrt{\frac{2(1-\hat{\delta})}{\hat{\delta}}}\frac{1}{T} \tag{6.35}$$

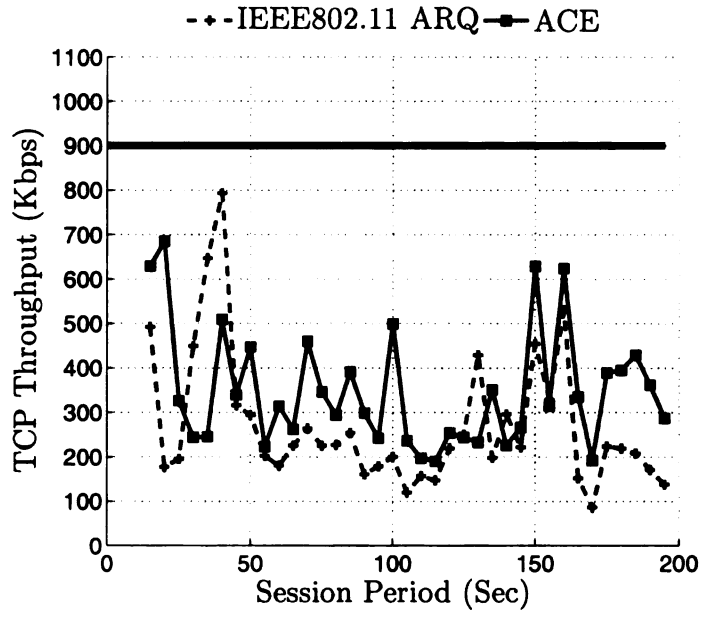
which is a well-known result on steady state behavior of TCP [83] where $\hat{\delta}$ is the equilibrium loss probability which is equivalent to π_{Loss} in our analysis.

The analysis of this section suggests that the steady state of loss probability decays dramatically when ACE is utilized at the link layer. Further, Equation (6.35) shows the direct impact of steady state loss probability of overall equilibrium throughput. Therefore, we expect that the TCP throughput improves dramatically when ACE is utilized at the link-layer. In the next section, we conduct an experimental analysis on TCP performance to demonstrate the accuracy of our intuition.

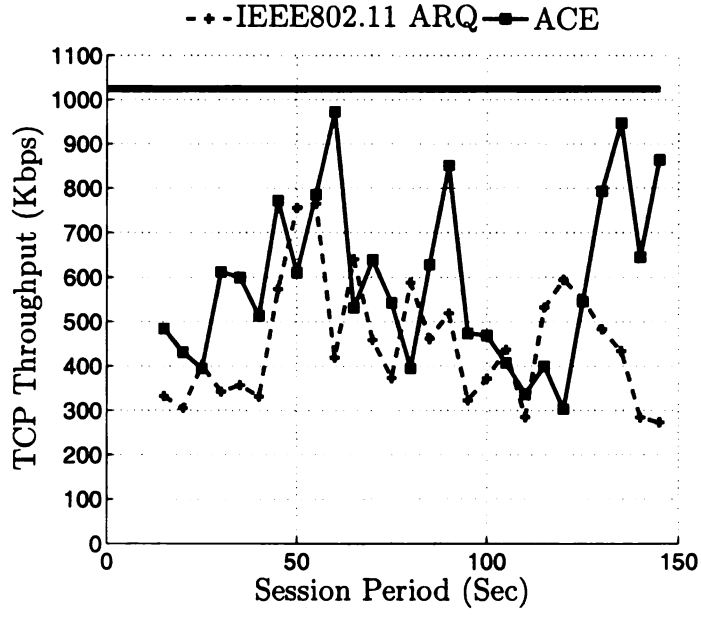
6.5.5 Experimental Performance Analysis

Using `INET Framework` TCP model in OMNET++, we analyze the TCP throughput variations over different channel traces when ACE and IEEE802.11 ARQ are deployed in the link-layer. We consider a heterogeneous network model consisting of wired and wireless sections depicted in Fig. 6.7.

Fig. 6.13 illustrates TCP throughput under different wireless channel conditions. We observe that over channels with low BER where the probability of packet loss is low, TCP achieves similar performance under ACE and IEEE802.11 ARQ. Under these channels, mostly *Congestion-based*



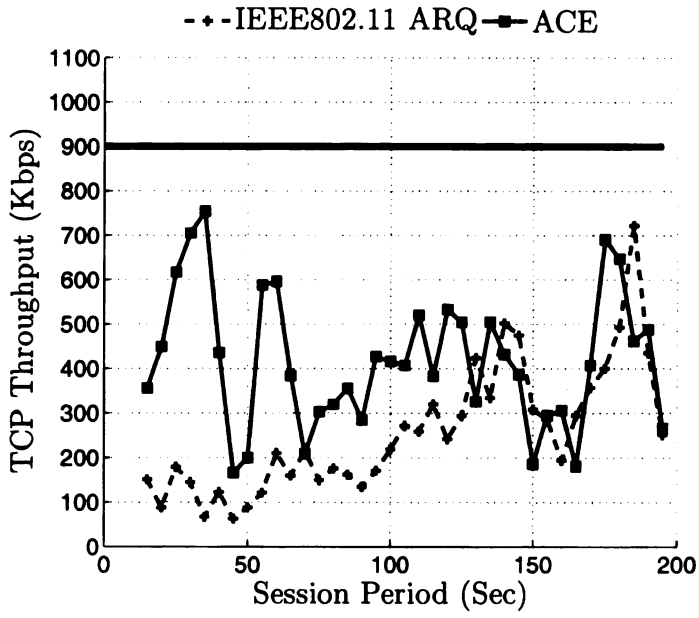
(a) BER: 0.002



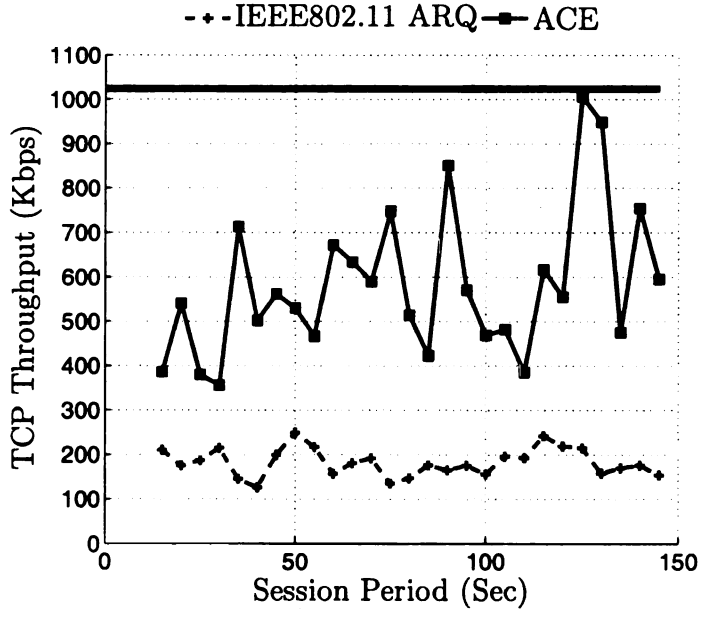
(b) BER: 0.005

Figure 6.13 TCP throughput variations over different channel traces having IEEE802.11 ARQ and ACE in the link-layer. The horizontal line in each figure represents the transmission rate of the corresponding channel trace.

losses are observed in the wired section. Meanwhile, significant throughput difference is easily identifiable as the channel BER increases. For instance, TCP gains throughput of 10% to 50% (e.g., ACE achieves 500-800Kpbs throughput while IEEE802.11 ARQ in under 200Kbps) over



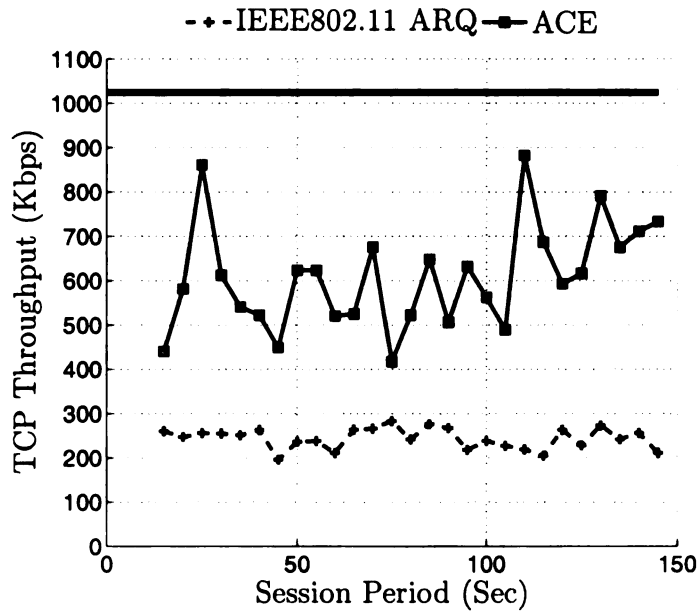
(c) BER: 0.009



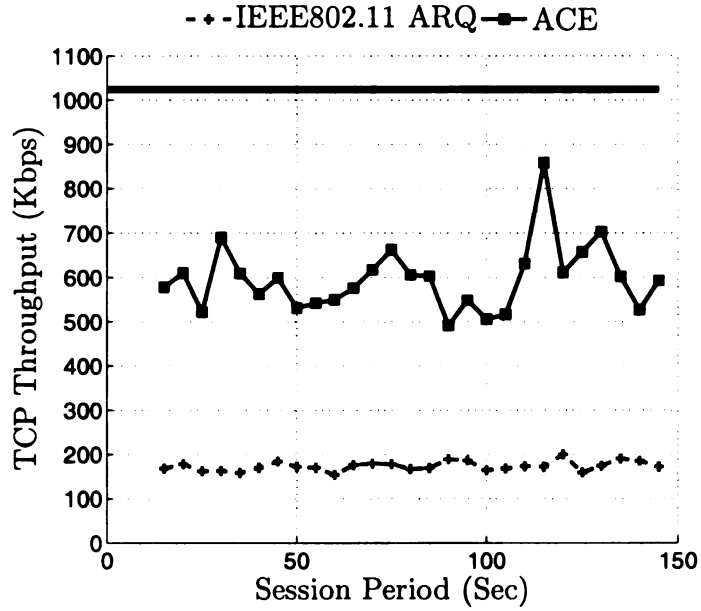
(d) BER: 0.011

Figure 6.13 continued.

channels with BER more 0.009 under ACE. This significant difference stems from the fact that ACE targets stability and reliability of wireless communication. This lead to fewer *AP-based* and *distortion-based* losses in wireless section under ACE in comparison with IEEE802.11 ARQ.



(e) BER: 0.018

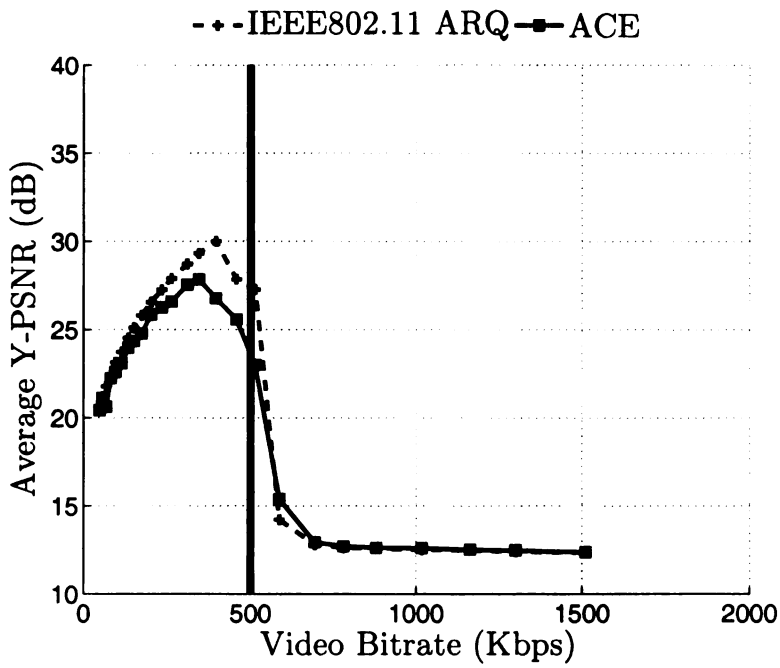


(f) BER: 0.025

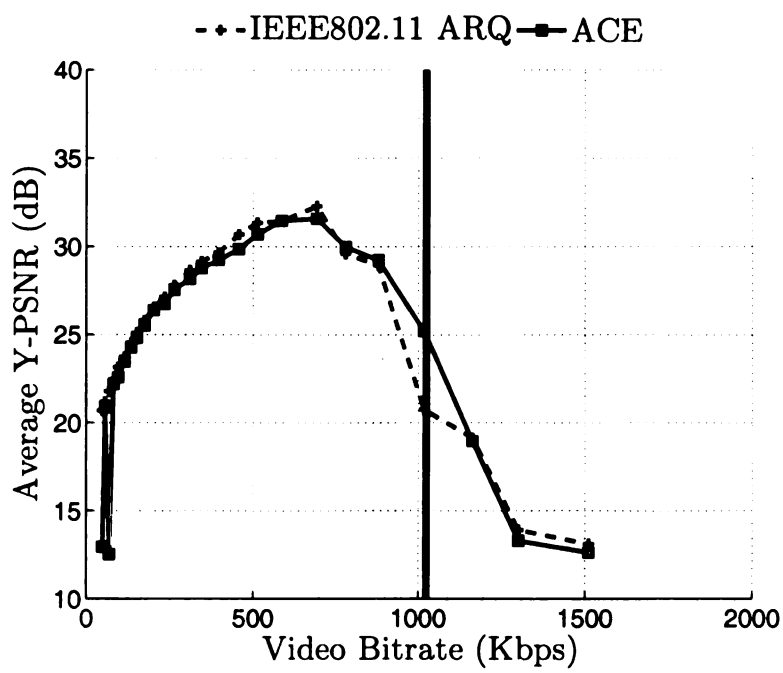
Figure 6.13 continued.

6.6 Realtime Video Simulation

To further analyze the impact of the ACE protocol and compare it with the conventional IEEE802.11 ARQ protocol over the performance of particular application, we simulate real-time

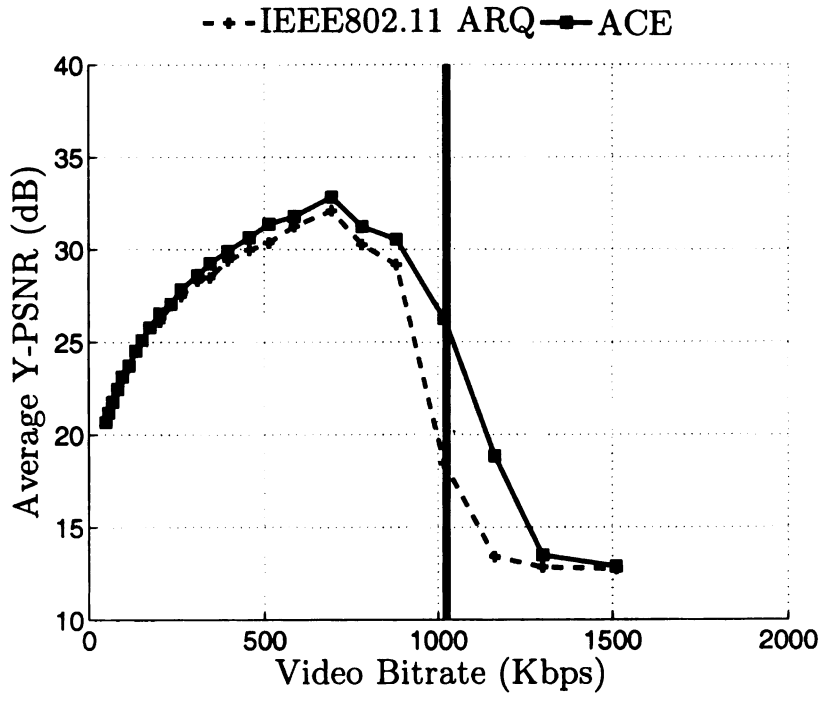


(a) BER: 0.002

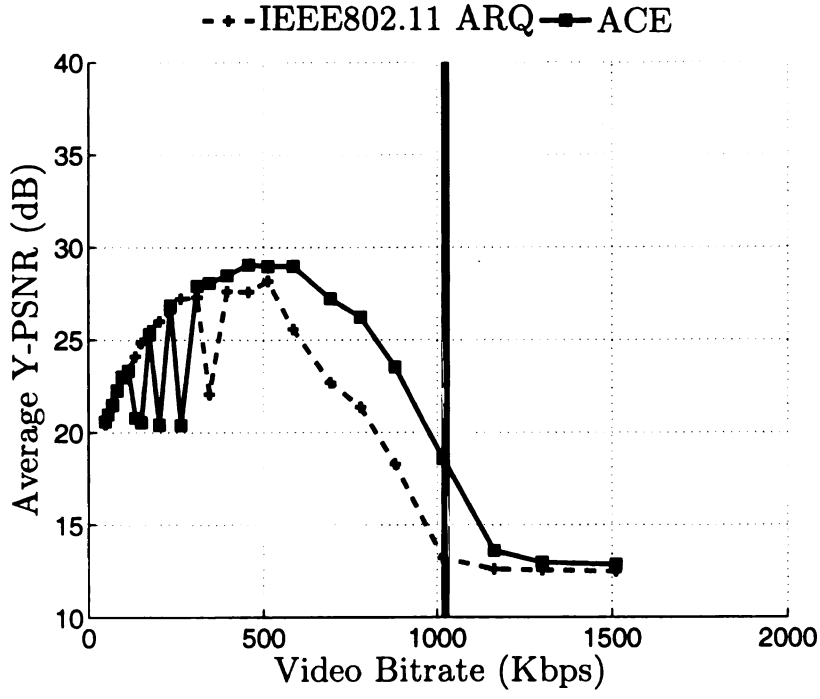


(b) BER: 0.005

Figure 6.14 Average $Y - PSNR$ with respect to variation of video rate over different channel traces. The vertical line in each figure represents the transmission rate of corresponding channel trace.

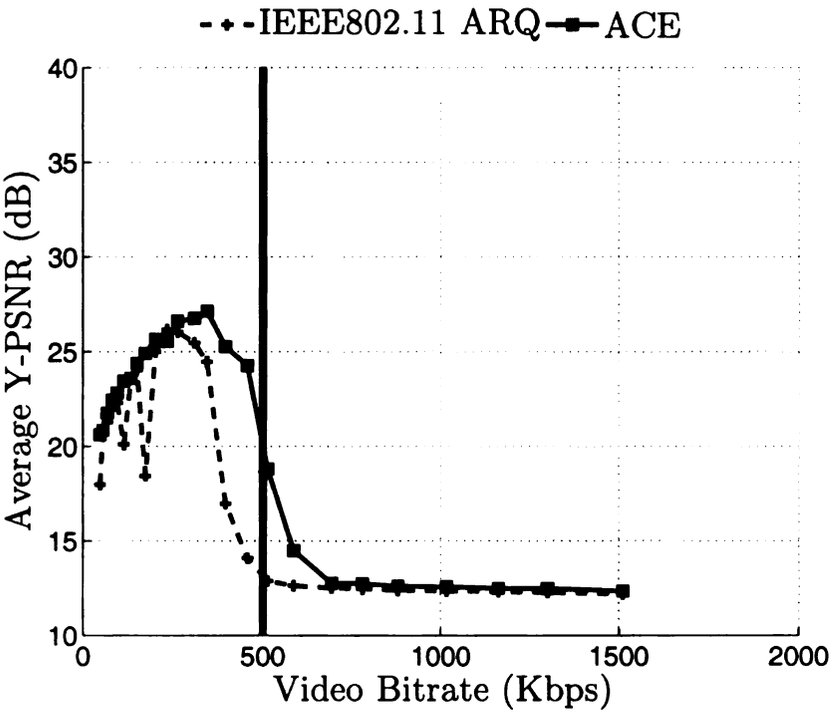


(c) BER: 0.009

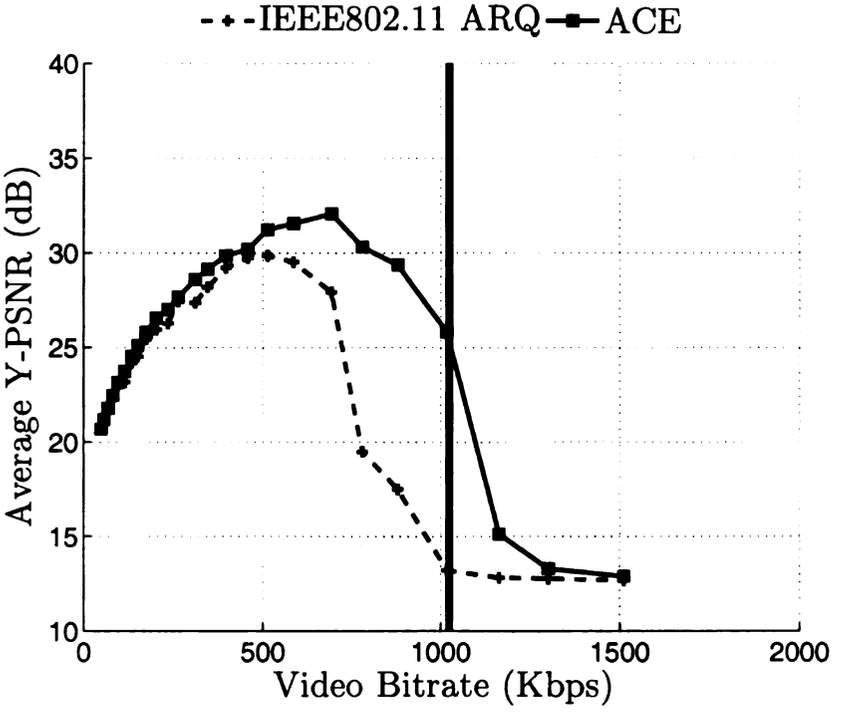


(d) BER: 0.011

Figure 6.14 continued.



(e) BER: 0.018



(f) BER: 0.020

Figure 6.14 continued.

video communication. The simulation setup is as follows: a particular video stream is encoded using the H.264/JVT standard software [106]. The encoded video streams (slices) are buffered at the sender to be transmitted over the wireless channel. The ACE protocol is simulated with the network simulator OMNET++ software [105]. ACE encodes each video slice using A-LDPC codes [104] and transmits the encoded packet over a wireless channel. Each transmitted packet is distorted based on the channel traces. Specifically, an XOR operation is performed between the trace packet and ACE packet. The corrupted packet is decoded using A-LDPC. The A-LDPC uses BER estimate determined by a channel model which was trained with previous received packets. If the packet is not decoded successfully, it is stored in receiver's buffer and additional redundancy is requested according to ACE.

To prevent frame-freezing or synchronization mismatches during real-time video communication (for e.g., video conferencing), the video packets need to arrive in a timely fashion. Those packets which miss their deadlines are unusable by the decoder, leading to degradation in video quality. To ensure smooth video playback, we require packets arrive at or above particular rate which is specified by the *video bitrate*. The simulation is terminated when all the video slices are transmitted by the sender. We measure the decoded video quality (average PSNR) for different *video bitrates*. We repeat the simulation to compute the performance of IEEE802.11 ARQ. We use IEEE802.11 [29] ARQ implemented in OMNET++ `INET Framework`. In these simulations, the maximum retransmission limit is set to four. To achieve a fair comparison, ACE receiver's buffer size is also set to four. For all simulations, the packet size is 1000 bytes and each video slice is of length 125 bytes.

Figure 6.14 illustrates the decode video quality of Stefan-CIF (30fps) sequence in terms of average PSNR over different channel traces. Notice that when video encoder/decoder uses low *video bitrate* the video quality decays. Therefore, in these plots, we observe a low PSNR value for both protocols for video rates below 100 Kbps. As the video rate increase, each video frame

is encoded using more data samples. We observe that for good channel conditions (BER less than 0.002), IEEE802.11 performs slightly better than ACE. The reason is that the level of noise over these channels is very low and since IEEE802.11 transfers video data only, more data is available for the video decoder resulting in slightly better video quality than that of under ACE. However, as the BER increases, PSNR values under IEEE802.11 ARQ tend to decline rapidly. Specifically, we observe ACE protocol ensures the video quality of 30dB for video rate 800Kbps over channel with BER 0.02 while IEEE802.11 ARQ is less than 20dB. Overall, we observe that utilizing ACE protocol over channels with BER more than 0.009 produce 5-10dB performance gain in video quality over wide range of video rates.

6.7 Discussion

In this chapter, we studied the problem of reliable and stable operations at the wireless link-layer. In particular, an Automatic Code Embedding (ACE) wireless link-layer protocol has been proposed that (a) employs a theoretically-sound framework and a corresponding strategy for embedding channel codes, using robust and well-defined code rates, in each packet; and (b) selects the code rates in an optimal and constrained manner to ensure reliability, stability, and maximum throughput. Through distinct analytical frameworks, we demonstrated that there is a unique solution for the code embedding rate at which stability and reliability at the link-layer is achievable. Our extensive analysis of ACE protocol over real channel traces collected on 802.11b WLANs for realtime and non-realtime traffic, TCP throughput and realtime video communication scenarios show that ACE significantly outperforms the conventional IEEE802.11 ARQ over varying wireless channels conditions. In the next chapter, we investigate the impact of prioritization on wireless communication. We build on the ACE framework to achieve preferred data recovery order across connections, while maintaining stable and reliable data flows in wireless networks.

CHAPTER 7

PACE: A Prioritized Wireless Link Layer

To achieve superior link-layer wireless communication one need to target the following critical objectives: (i) achieving sustained traffic stability (maintaining continuous realtime flow under delay constraints), (ii) ensuring maximal reliability and throughput, (iii) exploiting side-information for channel estimation/prediction, and (iv) interacting with the higher layers for prioritized communication and improving quality of service. We believe that these objectives represent a viable and necessary set to support a diverse wireless link-layer communication with various requirements in rate, reliability, and delay. However, achieving these objectives simultaneously is a difficult task since they have conflicting requirements. In our prior work [2], we have demonstrated the feasibility of designing stable and reliable link-layer under Automatic Code Embedding (ACE) framework over point-to-point (single-hop) 802.11 channels. However, what is ultimately needed is a comprehensive framework that targets all of the above objectives (stable-and-reliable communications, exploitation of side information, and interaction with the higher layers) jointly.

In recent years, various decoding schedulers [88–90] have been proposed to improve the overall throughput in wireless link-layer communication. The majority of these efforts either consider the standard ARQ approach [91] or the Hybrid ARQ (HARQ) schemes [92] such as: diversity combining [63], code combining [52], and incremental redundancy. Approaches outlined in [52,63,92]

are based on code puncturing methods [62]. These techniques assume (i) the complete knowledge of channel quality; and (ii) a slow-fading wireless environment. Under such assumptions, a decoder scheduling scheme only attempts to balance the quality of service for heterogeneous traffic requirements. This design strategy only targets quality of service and largely ignores the other objectives of wireless link-layer communication outlined above.

In this Chapter, we build on the ACE framework to achieve preferred data recovery order across connections, while maintaining stable and reliable data flows in wireless networks. Under the proposed Prioritized ACE (PACE) framework, our ACE based stable-and-reliable[1] link-layer will employ a novel rate-adaptive Low Density Parity Check (LDPC) channel codes while interacting with the higher layers to provide a dynamic decoder scheduling service over varying wireless channel condition. Specifically, we develop a LDPC decoding model to capture the decoding process for link-layer traffic and use it to determine an optimal code selection strategy for maximal bandwidth utilization. Further, we find an optimal code embedding rate under the PACE framework to jointly meet the reliability, stability, and delay constraints of the wireless link-layer communication. We classify heterogeneous link-layer traffic arrivals into different priority classes based on packet delay constraints and the distortion suffered. The traffic arriving in each priority class is modeled as a poisson process. Consequently, we formulate the link-layer buffer as a multiclass $M/G/1$ priority queuing system where the decoding process (service process) of the PACE buffer is captured by nonhomogeneous geometric distribution [99]. Given the link-layer buffer model and the LDPC decoding model, we determine the optimal dynamic decoder scheduling under the PACE framework. This scheduling policy is a special case of a classic scheduling problem solved by Plambeck et al. in [100] and is asymptotically optimal.

The PACE protocol incorporates the LDPC model and the dynamic scheduling policy into the original ACE protocol. We show experimentally that PACE improves the performance of the ACE

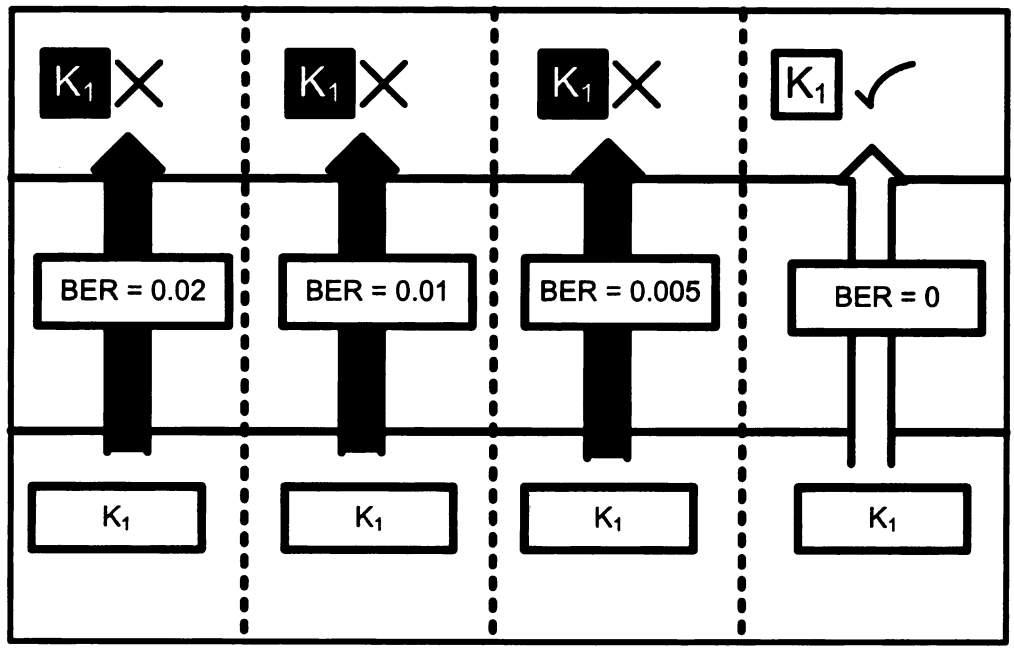
[1] For the definitions of reliability and stability under the ACE framework refer to Chapter 6.

protocol over wireless channels with varying conditions. Our contributions are:

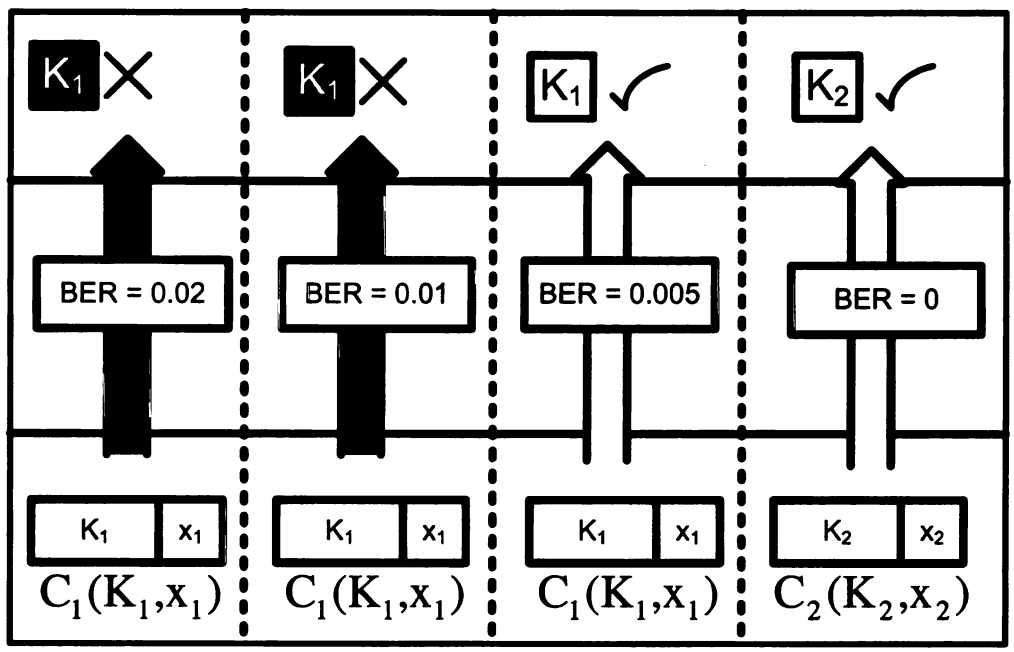
- Develop a LDPC decoding model used to select the code with best error-correcting capability from an ensemble of codes.
- Model the link-layer buffer as $M/G/1$ priority multiclass queuing system to determine an optimal dynamic decoder scheduling policy to achieve improved quality of service while maintaining sustained stability.
- We develop the PACE protocol that provides reliable and stable wireless communication for high demand and delay sensitive heterogeneous link-layer traffic.

7.1 Illustrative Example

In this section, we briefly consider a communication example between two wireless nodes. This example consists of four transmission intervals where each transmission interval τ represents a time slot during which a sender transmits a packet and receives its acknowledgment. The sender wants to transmit four data packets (e.g., $K_1, \cdots, K_4$) each with length of k bits to the receiver. The first and the last packets (K_1 and K_4) are non-realtime. That is, impact of the delay in delivering these packets to a higher layer is insignificant on the performance of the corresponding application (e.g., SMTP packets). On the other hand, K_2 and K_3 are realtime. Consequently, these packets should be passed up to the higher layer within an end-to-end delay constraint, otherwise they are unusable for the application (e.g., realtime video packets). The wireless channel condition is gradually improving. Specifically, the channel is in severe condition (BER greater than 0.02) in τ_1. During τ_2 and τ_3 the channel BER reduces to 0.01 and 0.005 and finally the channel is error-free in τ_4. We consider the following scenarios:



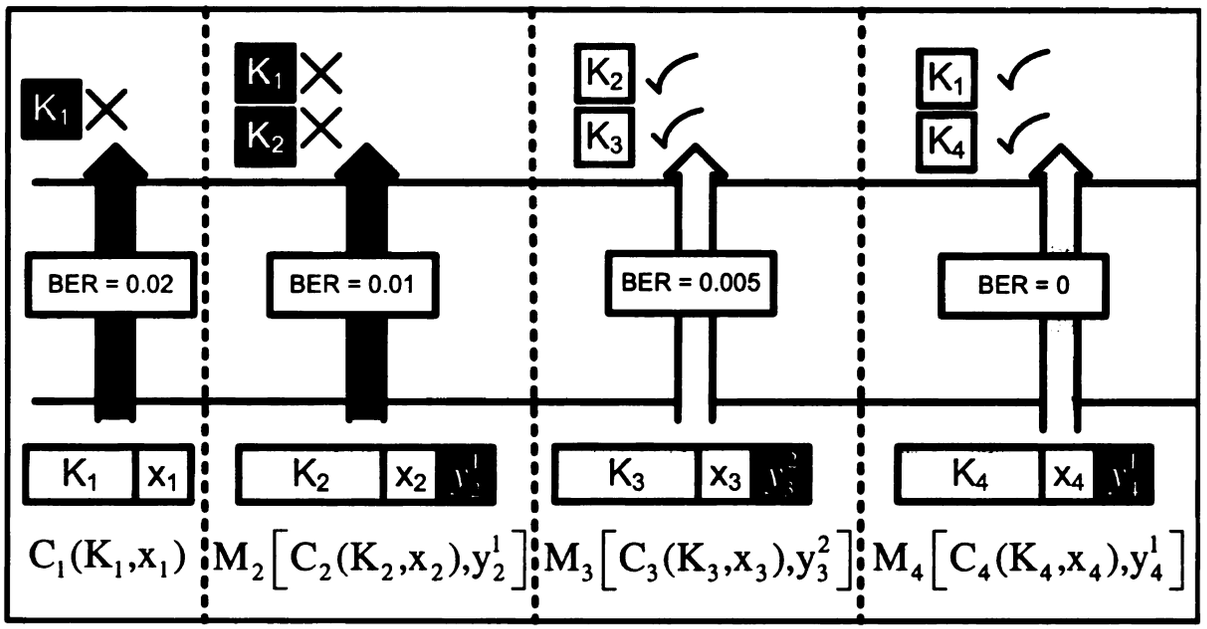
(a) IEEE802.11 ARQ



(b) HARQ

Figure 7.1 A wireless link-layer communication example consisting of four transmission intervals under different error control protocols.

1. *Ideal scenario:* Under an ideal scenario all four packets are transmitted over the channel and received without errors. This gives the throughput of one and all packets are delivered to



(c) PACE

Figure 7.1 continued.

the higher layer. However, since the channel BER is non-zero in the first three transmission intervals, the odds of receiving an error-free packet are very low.

2. *IEEE802.11 ARQ:* This scheme uses a retransmission for error recovery. As illustrated in Fig 7.1a, in τ_1, the sender transmits K_1. Since the channel BER is non-zero, the received packet is erroneous. Consequently, the receiver requests for a retransmission of K_1 in τ_2 and also in τ_3 because the channel BER is still non-zero. In τ_4, K_1 is finally delivered to the receiver without errors and passed up to the higher layer. This creates the throughput of $1 - \frac{3k}{4k} = 0.25$ since K_2, K_3 and K_4 are not transmitted during the four time-slot window of this simple example.

3. *Hybrid ARQ (HARQ):* Fig 7.1b shows the performance of HARQ. In τ_1, the sender encodes K_1 data packets with x_1 parity bits and creates a codeword $C_1(K_1, x_1)$. The receiver fails to decode C_1 and requests for a retransmission of C_1. In τ_2, the sender retransmits C_1. Since the channel BER is relatively high (BER is 0.01), the receiver cannot decode

C_1 and requests for the second copy of C_1. In τ_3, the receiver successfully decodes C_1. Consequently, in τ_4, the sender transmits $C_2(K_2, x_2)$. The channel is error-free in τ_4 and so the receiver decodes C_2 successfully. Data packets K_1 and K_2 are delivered to the higher layer. This creates the throughput of $0.25 \leq 1 - \frac{2k+x_1+x_2}{4k} \leq 0.5$ since K_3 and K_4 are not transmitted and furthermore, some of the channel bandwidth is consumed for the delivery of parity bits x_1 and x_2. Notice that in practice, the number of parity bits is significantly less than the number of data bits in a codeword (i.e., $x_i \ll k$).

4. *PACE:* Fig.7.1c shows the PACE protocol performance. In τ_1, a codeword $C_1(K_1, x_1)$ is sent. A receiver that fails to decode C_1, stores C_1 in its buffer and requests for additional parity bits (hereafter type-II parity bits) for C_1. In τ_2, the transmitter sends $M_2 = [C_2(K_2, x_2), y_2^1]$. The receiver uses x_2 to decode C_2 and employs type-II parity bits y_2^1 (y_i^j denote additional parity for $C_j, j < i$ transmitted in τ_i) in addition to x_1 to decode C_1. The receiver fails to decode C_1 and C_2. Since the codeword C_2 corresponds to the realtime data packet K_2, it has higher priority than C_1. Therefore, the receiver requests for type-II parity bits for C_2 rather than C_1. As a result, in τ_3, the sender transmits $M_3 = [C_3(k_3, x_3), y_3^2]$. In τ_3, the receiver successfully decodes C_3 using x_3 and $C_2(k_2, x_2 + y_3^2)$. Consequently, the receiver requests for type-II parity bits for C_1. Accordingly, in τ_4, the sender transmits $M_4 = [C_4(K_4, x_4), y_4^1]$. The receiver decodes C_4 using x_4 and $C_1 = (K_1, x_2 + y_2^1 + y_4^1)$ successfully. Therefore, the data packets $K_1, \cdots, K_4$ are delivered to the higher layer. The throughput under PACE is $0.75 \leq 1 - \frac{\sum_{i=1}^{4} x_i + y_2^1 + y_3^2 + y_4^1}{4k} \leq 1$.

This example demonstrates the efficacy of PACE in improving throughput-delay performance in wireless link-layer communication. However, the description above has intentionally ignored important details. For PACE to become practical we need to address the following challenges:

- *Optimal Parity Allocation:* PACE uses channel codes for error recovery and hence it requires embedding parity bits in every packet. Since the transmission of parity bits consumes channel bandwidth, it is important to identify the optimal allocation of parity bits in every transmission based on the channel condition and the channel code algorithm. A practical design of parity allocation for PACE has to ensure (i) high likelihood of successful decoding; and (ii) efficient utilization of the channel bandwidth.

- *Dynamic Decoder Scheduling* In every transmission, PACE should perform error recovery on the packets with higher priority and hence buffer less significant ones for future recovery. To achieve compliance with such policy, it is natural to consider two complementary modes of dynamic scheduling: (i) PACE should choose the order in which arriving packets are served to guarantee the delivery of the packets to the higher layer before they expired; and (ii) to reject or drop insignificant packets when the buffer length is judged to be excessive, thereby incurring penalties or lost potential throughput.

In what follows, we show how to address these two challenges. To that end, in the next section, we study the behavior of the link-layer traffic under the PACE framework and formulate necessary models which provide essential tools to develop a practical design of PACE.

7.2 Model Formulation

In this section, we develop the following models for PACE: (i) *LDPC decoding model* which captures the decoding process of link-layer traffic using LDPC codes; (ii) *Buffer model* which describes the link-layer buffer as a multiclass priority queuing system with a single server. These models will build the framework to determine optimal code selection and dynamic decoder scheduling strategies for the PACE protocol.

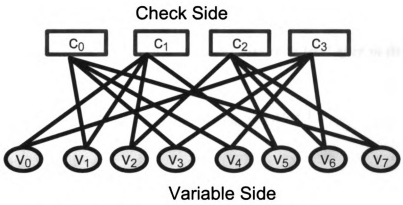


Figure 7.2 A LDPC Tanner graph with $d_v = 2$ and $d_c = 4$.

7.2.1 LDPC Decoding Model

PACE employs LDPC codes for decoding link-layer packets. Our objective in this section is to formulate the LDPC decoding process to select the code with best error correcting capability to maximize bandwidth utilization. PACE uses the LDPC check matrix represented by Tanner bipartite graph shown in Fig 7.2. The nodes of the graph are separated into two distinctive sets which are called variable and check nodes with the degree of d_v and d_c respectively. The PACE protocol uses an iterative belief propagation LDPC decoder [69] which attempts to correct errors in the received packet in m iterations. The following equation, by Gallager [68] shows the reduction of errors as the function of the iteration number m:

$$\begin{aligned} \epsilon^{(m)} = \epsilon^{(0)} - \epsilon^{(0)} \left[\frac{1 + (1 - 2\epsilon^{(m-1)})^{d_c - 1}}{2} \right]^{d_v - 1} \\ + (1 - \epsilon^{(0)}) \left[\frac{1 - (1 - 2\epsilon^{(m-1)})^{d_c - 1}}{2} \right]^{d_v - 1}, \end{aligned} \tag{7.1}$$

where $\epsilon^{(0)}$ represents the cross-over probability of the Binary Symmetric Channel (BSC) that the packet is transmitted over and $\epsilon^{(m)}$ is the probability of error in the packet after the m^{th} iteration.

Using Equation (7.1), we can formulate a relationship between the number of iterations in belief propagation method m, LDPC parity check matrix parameters d_v and d_c, and the amount of error in the received packet. This relationship is communicated to us by Karande [70] and presented in the following Lemma:

Lemma 1. *For a packet transmitted over BSC with cross-over probability ϵ_i that is decoded using LDPC check matrix with parameters d_v and d_c; the distortion level after m iterations can be approximated by*

$$\epsilon_i^{(m)} \approx \epsilon_i \left[\epsilon_i(d_v - 1)(d_c - 1)\right]^{m-1}$$

Proof. See Appendix. □

In a transmission interval τ_i, the PACE sender creates a codeword $C_i(k_i, x_i)$ by employing LDPC generator matrix, thus encoding k_i data bits with x_i type-I parity bits. Correspondingly, the receiver attempts to retrieve k_i data bits by utilizing x_i parity bits embedded in the received packet C_i. Specifically, the receiver utilizes a LDPC check matrix with $n_i = k_i + x_i$ variable nodes and x_i check nodes. Since the degree of each variable and check node is d_v and d_c respectively, the following equality holds:

$$n_i d_v = x_i d_c. \tag{7.2}$$

Lemma 1 suggests that LDPC belief propagation performance depends on the knowledge of channel condition ϵ_i and the check matrix parameters d_v and d_c. In addition, the receiver can correct a certain level of error proportional to the number of parity bits embedded in the packet. Therefore, the receiver is capable of correcting up to $\alpha(\epsilon_i) \times x_i$ errors out of n_i bits in the packet. Here $\alpha(\epsilon)$ measures the expected error-correcting capability of the LDPC soft decision decoder when the channel BER is ϵ. The sender uses the channel n_i times to transmit a codeword $C_i(k_i, x_i)$. Consequently, the amount of error introduced in the received codeword (on average) is $\epsilon_i n_i$. As a result the receiver fails to decode $C_i(k_i, x_i)$ if the amount of error in the received packet exceeds the error correcting capability of the receiver. That is,

$$\epsilon_i n_i \geq \alpha(\epsilon_i) x_i. \tag{7.3}$$

Using Equations (7.2) and (7.3), we obtain an upper bound on $\alpha(\epsilon_i)$:

$$\alpha(\epsilon_i) \leq \epsilon_i \frac{d_c}{d_v}. \tag{7.4}$$

Our main objective in this section is to select the code with best error-correcting capability. Toward this end, we employ Lemma 1 and Equation (7.4) to obtain the maximum value of $\alpha(\epsilon_i)$. According to Lemma 1, LDPC decoder successfully decodes $C_i(k_i, x_i)$ when the distortion level of the received packet after m iterations approaches zero. In practice, the LDPC belief propagation algorithm is configured such that the number of iterations m and the degree of variable nodes d_v are predefined and constant [104]. However, the degree of check nodes d_c is determined by the algorithm based on the number of available parity bits. Consequently, a successful decoding depends directly on d_c. On the other hand, Equation (7.4) suggests that error-correcting capability of the decoder cannot exceed the upper bound $\alpha_u(\epsilon) = \epsilon_i \frac{d_c}{d_v}$. So, to maximize $\alpha_u(\epsilon_i)$, we have the following optimization problem:

$$\begin{aligned} &\arg\max_{d_c} \alpha_u(\epsilon_i) \quad \text{subject to:} \\ &\epsilon_i \left[\epsilon_i(d_v - 1)(d_c - 1)\right]^{m-1} = 0 \ \text{ and} \\ &d_v, m \text{ are constants.} \end{aligned} \tag{7.5}$$

Solving (7.5) leads us to the best possible value of d_c and the maximum error-correcting capability $\alpha^*(\epsilon_i)$. Consequently, we select a code from an ensemble of codes which has the error-correcting capability close to $\alpha^*(\epsilon_i)$. In Section 7.3, we will use this code to obtain the optimal parity allocation strategy for PACE.

7.2.2 PACE Buffer Model

Each arriving packet at the PACE receiver was generated to serve a specific protocol or application in the higher layers. Depending on the type of the protocol or application, the packet is confined to a specific delay constraint. In addition, the packet contains a certain level of distortion due to the impact of wireless link-layer transmission. Under the PACE framework, each packet is classified into a specific priority class based on its delay constraint and distortion. The PACE

receiver then attempts to serve a specific priority class according to dynamic decoder scheduling rules. In Section 7.3.2 we will provide a thorough description of packet prioritization process and dynamic decoder scheduling policy. These procedures require a complete formulation of the PACE buffer. Therefore, in this section, we develop a comprehensive queuing model for the PACE buffer. To that end, we first model the arrival process and the service distribution of each priority class.

Arrival Process and Service Distribution

We consider the Markovian channel described in Chapter 3. The link-layer wireless channel is modeled as a discrete Markov chain with N states $S_1, \cdots, S_N$. Each state S_i is a representation of a BSC with a particular BER ϵ_i which is valued from a finite set F_N with length N: $\epsilon_i \in F_N, |F_N| = N$. In [2], and specifically Equation (4), we measured the probability of successful decoding of a packet $C(k_i, x_i)$ transmitted over the channel in state S_i. Using the LDPC decoding model, we reformulate this probability measure as follows:

$$P(E_i \leq \alpha(\epsilon_i) \times x_i) = F_{E_i}(\alpha(\epsilon_i)x_i) = \sum_{d=0}^{\lfloor \alpha(\epsilon_i)x_i \rfloor} e^{-\lambda_{E_i}} \frac{\lambda_{E_i}^d}{(d)!}. \tag{7.6}$$

Here E_i represents the packet distortion level that has a Poisson distribution with cumulative density function $F_{E_i}(u)$ and rate λ_{E_i}.

The PACE receiver stores the arriving packets in its buffer when the decodings with type-I parity bits are unsuccessful. Consequently, the arrival process of packets with distortion level E_i in the PACE buffer can be modeled as a poisson process with the rate λ_i which is equivalent to the probability of decoding failure with type-I parity bits:

$$\lambda_i = 1 - F_{E_i}(\alpha(\epsilon_i)x_i) = \bar{F}_{E_i}(\alpha(\epsilon_i)x_i). \tag{7.7}$$

On the other hand, each arriving packet at the PACE is confined to a specific delay constraint which is *independent* of the packet error. Consider the situation where a packet with delay con-

straints d_j arrives according to a Poisson process with rate λ_j. Notice that from the PACE buffer vantage point, the arrival process of this packet is *independent* of the arrival process of a packet with error E_i. However, both of these processes are Poisson processes. Therefore, if a packet with delay constraint d_j and distortion level E_i is classified into priority class l, then the arrival process of this priority class is a poisson process with rate λ_l where

$$\lambda_l = \lambda_i + \lambda_j. \tag{7.8}$$

A packet departs the PACE buffer either if (i) it is successfully decoded with additional type-II parity bits. In this case the data bits embedded in the packet are delivered to the higher layer; or (ii) its delay constraint expires, meaning that the packet is timed out and is dropped from the buffer. In the latter case the service distribution of the PACE decoder has zero density since the packet is dropped. In the former case, the service distribution depends on the probability of successful decoding of a packet with type-II parity bits. The probability of successful recovery of a particular packet increases as more type-II parity bits are added to the packet. Consequently, the error correction process of every packet under PACE has a nonhomogeneous geometric distribution [99] with the density function $f_G^i(t)$ with the parameter $p_t^i = F_{E_i}(\alpha(\epsilon_i)z_t)$:

$$f_G^i(t) = P(\text{Successful recovery on } t \text{ trial}) = p_t^i \prod_{k=1}^{t-1} (1 - p_k^i). \tag{7.9}$$

where p_t^i measures the likelihood of successful decoding (when the channel is in state S_i) on t^{th} decoding trial using total parity bits (type-I and type-II) of z_t. Therefore, the service distribution G_l for a priority class l is $f_G^l(t)$ which is a truncated $f_G^i(t)$ on d_j:

$$f_G^l(t) = \begin{cases} f_G^i(t) & \text{if } d_j - t > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.10}$$

The service rate of priority class l is the expected value of the density function $f_G^l(t)$. That is,

$$\mu_l = E[f_G^l(t)] = \begin{cases} \frac{1}{p_t^i} & \text{if } d_j - t > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.11}$$

According to Equation (7.8), the traffic arriving in each priority class is a poisson process. On the other hand, a decoding process of each priority class has a nonhomogeneous geometric distribution given in (7.10). Consequently, the PACE buffer can be modeled with a multiclass $M/G/1$ priority queuing system with a single server. Here the arriving customers represent packets with different priority classes and the PACE decoder is the server.

Consider the situation where the packets in the PACE buffer are classified into $1, \cdots, L$ priority classes, which arrive according to independent Poisson processes with respective rates $[\lambda_l]_{l=1}^{L}$ and have service distributions $[G_l]_{l=1}^{L}$ as calculated in Equations (7.8) and (7.10). Let W_l denote *priority delay*, the average waiting time of a packet with priority class l in the PACE buffer before it is successfully transmitted to the higher layer. Our objective is to compute the W_l.

Under $M/G/1$ priority queuing system, V_l the average amount of decoding time for priority class l is as follows [71]:

$$V_l = \lambda_l E[G_l] W_l + \frac{1}{2}\lambda_l E[G_l^2]. \quad l = 1, \cdots, L \tag{7.12}$$

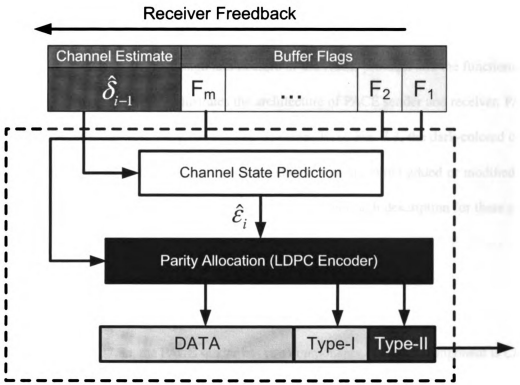
On the other hand, the *priority delay* of an arbitrary class l is equivalent to the amount of decoding time in the system requires upon its arrival plus the decoding time that remains for other arrivals classes that are already under the service. Therefore,

$$W_l = V_l + V_j. \quad j \neq l, l, j = 1, \cdots, L \tag{7.13}$$

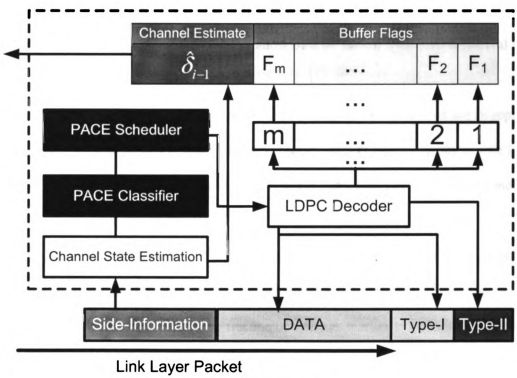
Using Equations (7.12),(7.13) and some mathematical manipulations, we can solve for individual W_l:

$$W_l = \frac{\sum_{l=1}^{L} \lambda_l E[G_l^2]}{2\prod_{j=l-1}^{l}\left[1 - \sum_{k=1}^{j} \lambda_j E[G_j]\right]}. \tag{7.14}$$

In this section, we modeled the PACE buffer as a multiclass $M/G.1$ priority queuing system and we obtained the arrival and service rates, and the expected delay for each priority class in the buffer. In Section 7.3.2, we will employ this model to find an optimal dynamic decoder scheduling policy for PACE.



(a) PACE Sender



(b) PACE Receiver

Figure 7.3 The design architecture of the PACE protocol.

7.3 PACE Protocol

In this section, we describe the design architecture of the PACE protocol and the functionality of each of its components. Fig 7.3 illustrates the architecture of PACE sender and receiver. PACE is built on the ACE framework developed in [2]. Specifically, in Fig. 7.3, the dark-colored components in sender and receiver sides are those components that are either added or modified in the PACE framework. In the following sections, we present a thorough description for these components.

7.3.1 PACE Sender

As illustrated in Fig. 7.3a, the PACE sender has two components. The first component is *Channel State Prediction* where the link-layer wireless channel condition for the next transmission interval is predicted based on the receiver feedback. Specifically, the sender uses $\hat{\delta}_{i-1}$, the receiver channel estimate for τ_{i-1} as its prediction of the channel BER in current transmission interval τ_i; so $\hat{\epsilon}_i = \hat{\delta}_{i-1}$.

The second component is *Parity Allocation* where a new codeword is generated and an appropriate number of type-II parity bits is added to a packet for the next transmission. We use the LDPC model developed in Section 7.2.1 to select an appropriate code (based on the channel condition) to find an optimal parity allocation.

In [2], we proved that under the ACE framework, there exist only one optimal code embedding rate under which the reliability and stability of link-layer traffic is ensured. Recall that the operational code embedding rate measures the fraction of data bits that are embedded in a codeword. For instance a codeword $C_i(k_i, x_i)$ is generated based on the code rate $R_i = \frac{k_i}{k_i+x_i}$. This finding is repeated in the following Lemma:

Lemma 2. *An optimal solution for code embedding rate that ensures reliability and stability in*

wireless transmission over a channel in state S_i is a unique solution and is given by:

$$R_i = 1 - \frac{\epsilon_i}{\alpha}. \quad \epsilon_i \ll \alpha$$

where α is error-correcting capability of a decoder.

Proof. See Appendix. □

Lemma 2 determines the optimal code embedding rate for a general decoder with error-correcting capability α. However, the LDPC decoder model selects a code (when the channel BER is ϵ_i) with a error-correcting capability close to $\alpha^*(\epsilon_i)$ computed in (7.5). In practice, PACE uses $\alpha^*(\hat{\epsilon_i})$ to measure the expected error-correcting capability of the LDPC soft decision. This is so since PACE sender uses $\hat{\epsilon}_i$ as an estimate of the expected error in the next transmission interval τ_i. Therefore, an operational optimal code embedding rate for parity allocation is

$$R_i^* = 1 - \frac{\hat{\epsilon}_i}{\alpha^*(\hat{\epsilon}_i)}. \tag{7.15}$$

PACE first determines the amount of type-II parity bits (y_i^k) for a particular codeword C_k transmitted in some $\tau_k, k < i$. According to Equation (7.15) C_k with length n_k requires at most $z_i^k = \frac{n_k \hat{\epsilon}_i}{\alpha^*(\hat{\epsilon}_i)}$ parity bits for error recovery. However, for C_k, $x_k + \sum_{l=k+1}^{i-1} y_l^k$ parity bits are already transmitted in the previous transmission intervals $\tau_k, \cdots, \tau_{i-1}$. Thus, type-II parity bits necessary to transmit in τ_i for C_k is $y_i^k = z_i^k - x_k + \sum_{l=k+1}^{i-1} y_l^k$.

After allocating type-II parity bits for codewords in the PACE buffer, PACE has $n_i = n - \sum_k y_i^k$ bits to transmit a new codeword $C_i(k_i, x_i)$ where n is the maximum number of bits that can be transmitted in a packet. Similarly, the amount of parity bits necessary for encoding C_i is $x_i = \frac{n_i \hat{\epsilon}_i}{\alpha^*(\hat{\epsilon}_i)}$, and therefore the amount of *new* data in C_i is $k_i = n_i - x_i$. The PACE sender then transmits a new codeword along with the additional type-II parity bits in a new link-layer packet denoted by $M_i = [C_i(k_i, x_i), \mathbf{y}_i]$ to the receiver.

7.3.2 PACE Receiver

Upon the reception of the link-layer packet $M_i = [C_i(k_i, x_i), \mathbf{y}_i]$, the type-II parity bits are extracted and sent to the receiver buffer. PACE receiver first attempts to decode a codeword $C_i(k_i, x_i)$ with type-I parity bits x_i. If the decoding is successful, the data bits k_i are sent to the higher layer. But if the decoding fails, then the codeword is sent to the receiver buffer for future recovery.

Under the PACE framework, each packet is mapped to a specific priority class based on its delay constraint and distortion suffered. The PACE receiver then attempts to serve packets with a specific priority class according to dynamic decoder scheduling rules. The *Classifier* and *Scheduler* components in Fig. 7.3b implement the packet prioritization and dynamic decoder scheduling policy in the PACE receiver.

PACE Classifier

Consider a codeword $C_i(k_i, x_i)$ was transmitted over the channel with state S_i. Let d_j represent the delay constraint of the packet with the distortion level E_i. The PACE *Classifier* assigns this packet to a priority class l based on the parameters d_j and E_i using a classification function $g : [i, j] \rightarrow l$, where

$$g = g\left(h_D(\tau_i, d_j), f_G^i(\tau_i)\right) \quad i \in R^N \times j \in R^M : l \in R^{M \times N} \tag{7.16}$$

where $h_D(.,.)$ is the delay penalty function, and $f_G^i(t)$ is the error-correcting density function given in Equation (7.9). The domains of packet distortion and delay constraint values are presented by R^N and R^M respectively.

The delay penalty function $h_D(\tau_i, d_j)$ measures the cost of postponing the delivery of data bits k_i to the higher layer in transmission interval τ_i based on packet delay constraint d_j. Specifically, $h_D(.,.)$ ranges from zero to one where the cost of dropping a packet is one (the packet is never

delivered). We use the following delay penalty function [101]:

$$h_D(\tau_i, d_j) = a \exp\left[b(\tau_i - d_j)\right], \tag{7.17}$$

where a and b are normalizing coefficients.

The PACE *Classifier* first computes the penalty weight for each packet stored in the PACE buffer. Specifically, the penalty weight w_l for a packet with distortion level E_i and delay constraint d_j is

$$w_l = h_D(\tau_i, d_j)\left[1 - f_G^i(\tau_i)\right]. \tag{7.18}$$

Notice that w_l approaches one if and only if the delay penalty function $h_D(.,.)$ reaches one and $f_G^i(.)$ is very small. This is an indication of a critical situation where the packet is reaching its deadline and has to be decoded immediately and at the same time the likelihood of decoding the packet with the current number of parity bits is very low. Therefore, the PACE *Classifier* classifies a packet with the highest penalty weight to a higher priority class. That is the priority classes $1, \cdots, L$ are numbered so that

$$w_1 \geq w_2 \geq \cdots \geq w_L.$$

PACE Scheduler

Accordingly to the PACE buffer model, the l^{th} priority class has the arrival rate λ_l, service rate μ_l, and the *priority delay* W_l, as obtained in Equations (7.8), (7.11) and (7.14). Consider a deterministic fluid analogy in which fluid of class l arrives at a constant rate λ_l and can be drained at rate μ_l if the PACE receiver devotes all its capacity to class l. If the fluid level of class l in the buffer is $N_l(t)$ at time t, then the oldest fluid of that class arrived $\frac{N_l(t)}{\lambda_l}$ time units earlier. Thus the fluid level of class l is not increasing if

$$\frac{N_l(t)}{\lambda_l} \leq W_l \rightarrow N_l(t) \leq \lambda_l W_l. \tag{7.19}$$

Therefore, the *backlog* of class l in the system at time t is given by:

$$\eta_l(t) = \frac{N_l(t)}{\lambda_l W_l} \tag{7.20}$$

Consequently, for a workload process $Q(t)$ defined as

$$Q(t) = \sum_{l=1}^{L} \mu_l N_l(t), \tag{7.21}$$

the associated threshold level is

$$q = \sum_{l=1}^{L} \mu_l \lambda_l W_l. \tag{7.22}$$

The PACE *Scheduler* performs the *dynamic decoder scheduling policy* which has two parts: (i) *Sequencing rule*: the PACE receiver at each decision point t (every transmission interval), decodes the oldest packet from the class l having the largest backlog $\eta(t)$; and (ii) *Rejection rule*: The PACE receiver drops a packet of class L (class L has the lowest priority significance) from its buffer if and only if $Q(t) > q$. Plambeck et al. in [100] proved that the above scheduling policy is asymptotically optimal under the heavy traffic condition.

In this section, we described the functionality of the *Classifier* and *Scheduler* components added in the PACE receiver. In the next section, we conduct extensive performance evaluations of PACE to demonstrate the advantage of the new components added to ACE in comparison with the original ACE protocol and other leading link-layer protocols.

7.4 Experiment

In this section, we present performance evaluations of the PACE protocol on real wireless channel traces collected on an 802.11b WLAN [2]. Specifically, we use 41 channel traces, each with unique average BER to simulate 41 various channel conditions. We compare the performance of the PACE protocol as opposed to the ACE, HARQ, and IEEE802.11 ARQ protocols. In particular, we first show the impact of throughput-delay tradeoff on the performance of each protocol. Then,

we measure the performances of a *realtime* video and a *non-realtime* TCP applications in conjunction with these link-layer protocols. For the following experiments, all four protocols PACE, ACE, HARQ, IEEE802.11 ARQ are implemented with OMNET++ network simulator [105]. We use an Adaptive LDPC (A-LDPC) codes [104] for channel coding operations in PACE and ACE, and Reed-Solomon codes [67] for HARQ[2].

7.4.1 Throughput-Delay Tradeoff

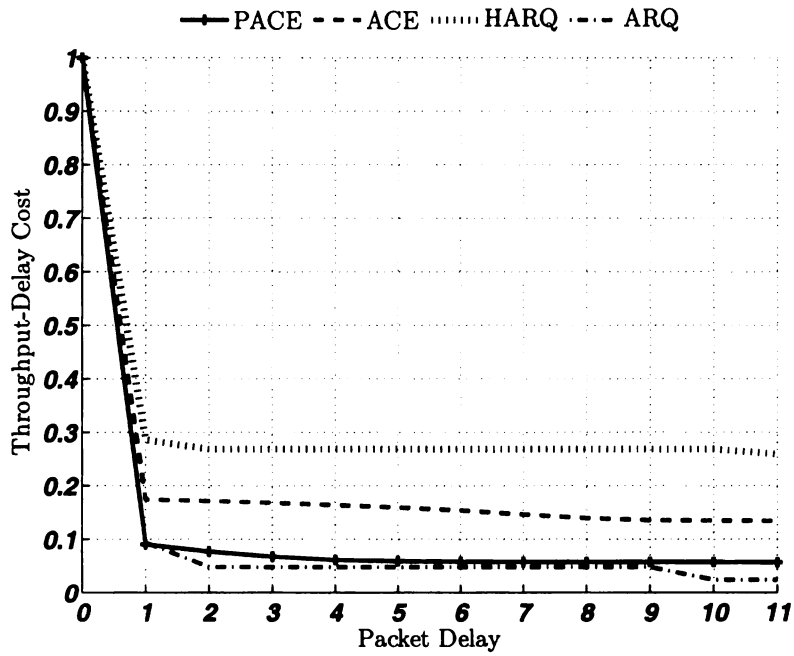
The throughput-delay tradeoff suggests that a higher throughput can be achieved with a higher tolerable delay. Consequently, applications with time sensitive delays (realtime delay constraints) suffer from low throughput. In this section, we measure the cost of this tradeoff on the link-layer protocol performance. We define the throughput-delay cost $\zeta(t)$ as follows:

$$\zeta(t) = 1 - \theta(t)\varsigma(t), \tag{7.23}$$

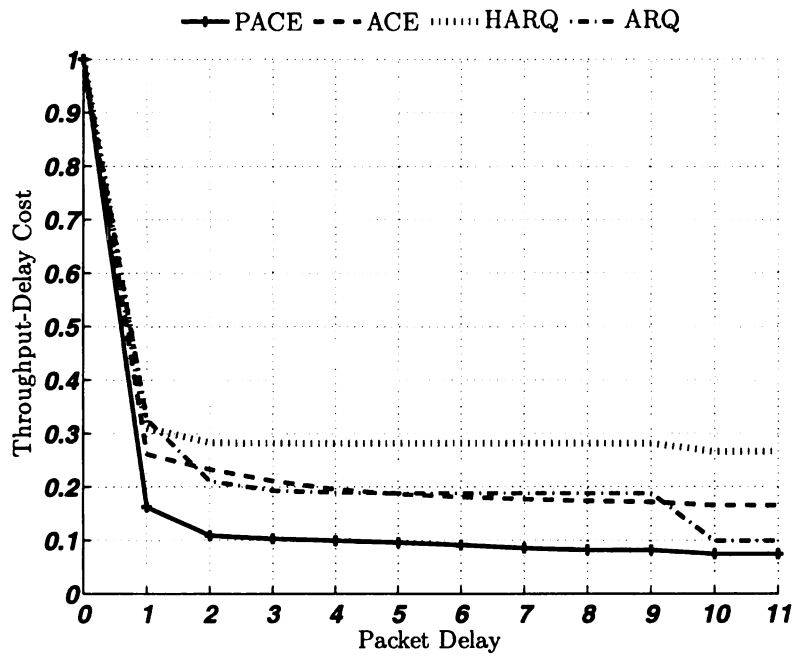
where the $\theta(t)$ is the throughput cumulative density function (CDF), measuring the fraction of data bits, and $\varsigma(t)$ is the delay CDF, measuring the fraction of packets that are delivered *successfully* to a higher layer by the time t. Notice that the throughput-delay cost function $\zeta(t)$ approaches zero if and only if both $\theta(t)$ and $\varsigma(t)$ approach one at time t.

Consider a packet arrives at time t_0 and has not decoded by time $t \geq t_0$, then its "delay" at time t is by definition $t_n = t - t_0$. In Fig. 7.4, we show the throughput-delay cost $\zeta(t_n)$ for each protocol with respect to the packet delay t_n over six different wireless channel conditions. Specifically, in Fig. 7.4a, where the channel average BER is very low (0.001), we observe that the cost reduces dramatically for the packet delay greater than one. This is so since the likelihood of packet error over this channel is very low. Consequently, most of the packets are received without

[2] It is important to note that HARQ protocols use hard decision algorithms. Reed-Solomon codes are known to be Maximum Distance Separable (MDS) codes and therefore used in HARQ protocols.

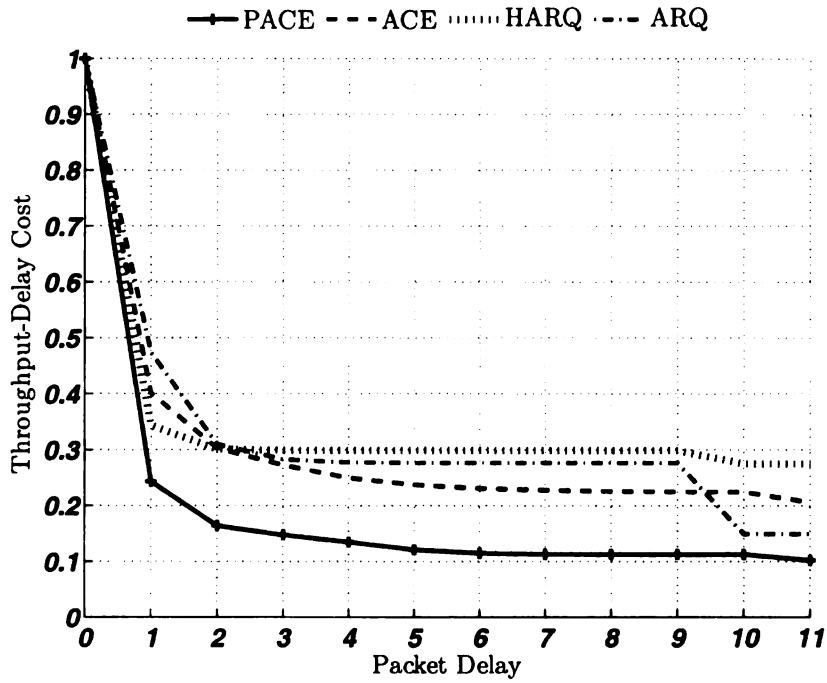


(a) BER: 0.001

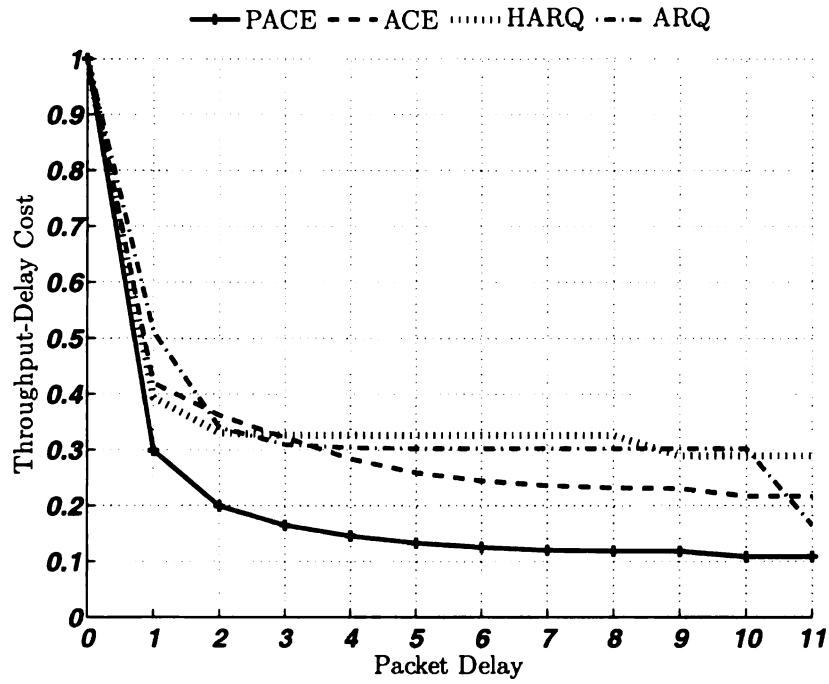


(b) BER: 0.005

Figure 7.4 The variation in throughput-delay cost with respect to the packet delay over different channel traces.



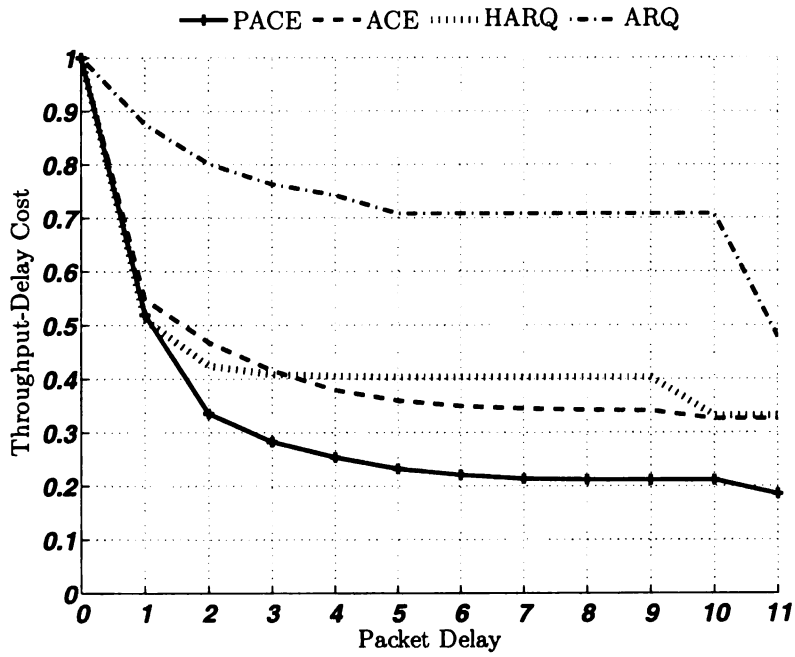
(c) BER: 0.007



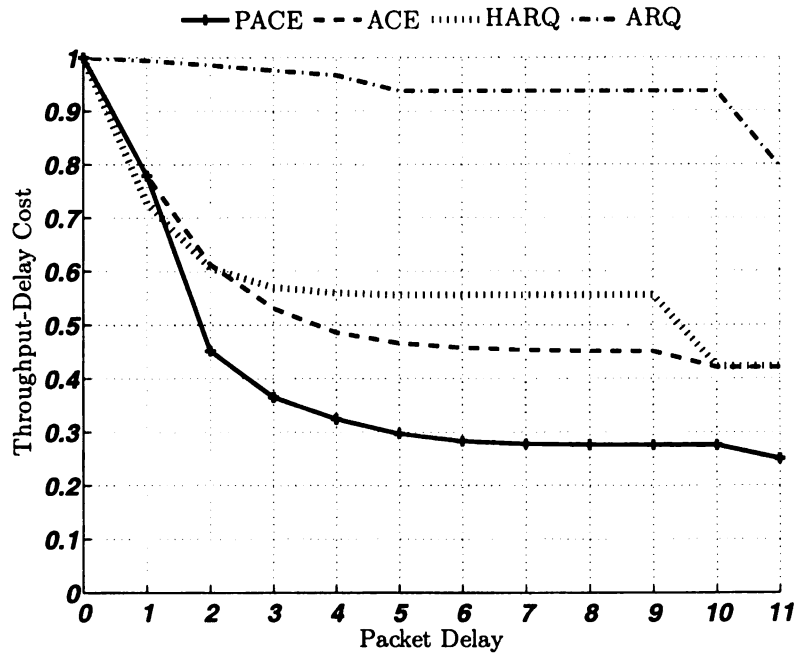
(d) BER: 0.009

Figure 7.4 continued.

errors and passed up to the higher layer immediately. However, as the channel BER increases, the likelihood of receiving errornous packets increases. Accordingly, IEEE802.11 ARQ performs



(e) BER: 0.018



(f) BER: 0.020

Figure 7.4 continued.

retransmissions to deliver an error-free packet. This in turn will result in the consumption of channel bandwidth and longer packet delays. Consequently, we observe that the throughput-

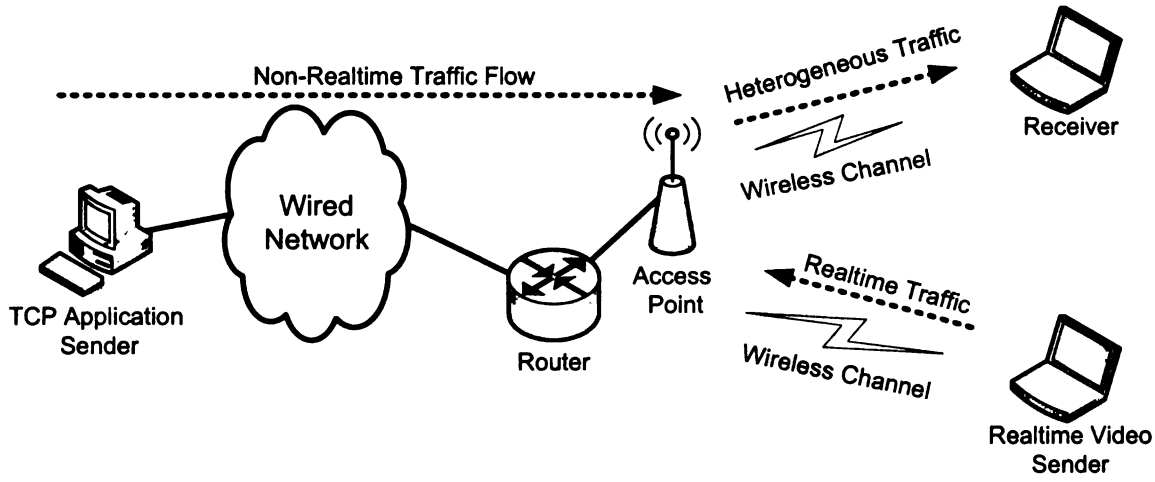


Figure 7.5 Simulation setup where heterogeneous traffic is generated by non-realtime TCP and realtime video flows.

delay cost for IEEE802.11 ARQ does not reduce significantly as the channel BER increases. For instance, the cost of IEEE802.11 ARQ hovers around 0.7 over the channel with BER 0.018. On the other hand, unlike HARQ, PACE and ACE employ adaptive parity allocation based on the channel condition. Accordingly, we observe that the cost function of these protocols is always lower than HARQ protocol regardless of channel condition. However, for the channel with low BER, where the packet prioritization has no significant impact, we observe that PACE performs better than ACE. We observe this because PACE selects an optimal code to maximize channel bandwidth utilization. In addition, for the channels with higher BER value, PACE still outperforms other protocols since it employs packet prioritization. For instance, the PACE cost reduces below the 0.3 point for packet delays greater than two while ACE, HARQ, and IEEE802.11 ARQ have respective costs of more than 0.4, 0.5 and 0.9. Overall, the PACE protocol gains $10\% - 40\%$ reduction in throughput-delay cost.

7.4.2 Realtime and Non-Realtime Application Performance

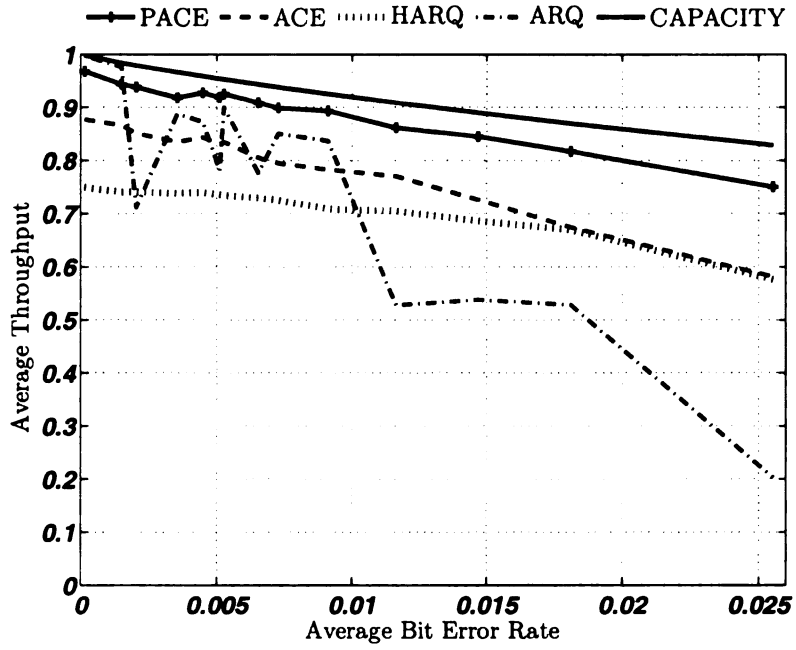
In this section, we evaluate the performances of a *realtime* video and a *non-realtime* TCP applications in conjunction with PACE, ACE, HARQ and IEEE802.11 ARQ protocols at the link-layer.

Fig. 7.5 illustrates the simulation setup of this section. Specifically, a TCP application sender is sitting at the wired section of the network and transmitting the TCP packets to the wireless receiver. At the same time, a realtime video sender is also transmitting video packets. The TCP and video traffic both are redirected to the receiver by the access point (AP). Using OMNET++ `INET framework`, we implemented this simulation setup for each link-layer protocol. Specifically, for the TCP application sender, we use the `TCPGenericCliAppBase` module to simulate a generic TCP application at the sender and receiver. We evaluate the average throughput of TCP packets. The average throughput measures the fraction of channel capacity that is utilized for a transmission of TCP packets. Further, for the realtime video application, we use `H.264/AVC JM14.0` codec [106] to serve as video encoder/decoder. The realtime video sender encodes *Stefan CIF 30fps* sequence and transmits it to the receiver at 300Kbps video rate. The decoder receives the video packets and decodes them if and only if the video packets arrive before the realtime deadline. Hence, those video packets that miss the deadline are unusable for video decoding. This leads to degradation in video quality as measured by Y-PSNR. Y-PSNR is a function of the mean square error (MSE) between the values of the original and decoded Y frame pixels:

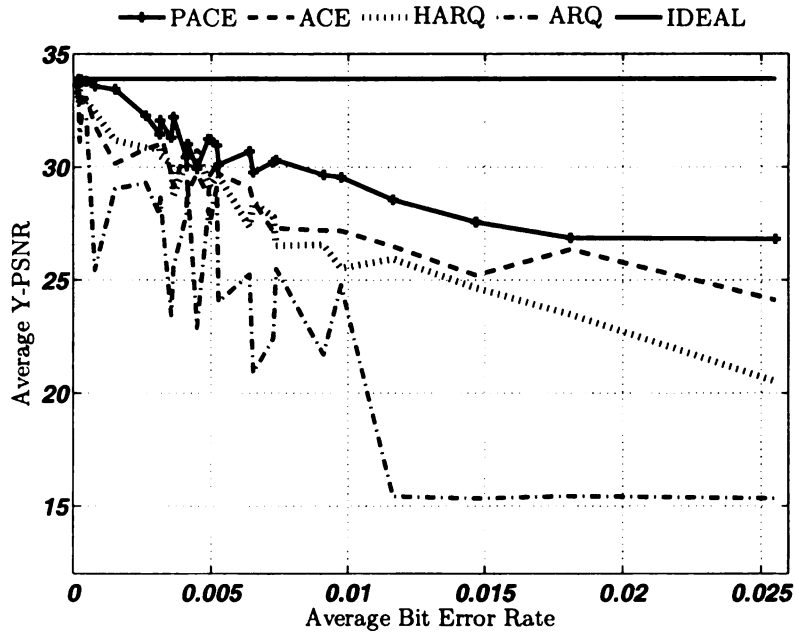
$$Y_{PSNR} = 10log_{10}\left[\frac{255^2}{||\mathbf{Y}_{org} - \mathbf{Y}_{dec}||_2^2}\right]. \tag{7.24}$$

We repeat this simulation to evaluate the average throughput and PSNR values over 41 channel traces. Therefore, a total of 164 simulations were conducted for this section.

Fig. 7.6a shows the average throughput of the TCP packets over various channel conditions. In this figure the solid line is the channel capacity. The channel capacity gives an upper bound on the average reliable information that can be transmitted over the channel. We observe that for a very low channel BER, IEEE802.11 ARQ achieves a higher throughput than other protocols. This is so since the channel is almost error-free and IEEE802.11 ARQ uses the channel to transmit only data bits. Consequently, its throughput is near channel capacity. On the other hand, PACE and ACE protocols achieve higher throughput than HARQ due the efficacy of adaptive parity



(a) Throughput



(b) PSNR

Figure 7.6 The performances of non-realtime and realtime applications in terms of throughput and video quality.

allocation. Meanwhile, as the channel BER increases slightly, the performance of IEEE802.11 ARQ decays significantly. The ACE and HARQ protocols manage to perform relatively better

than IEEE802.11 ARQ protocol but still their performances are noticeably below the channel capacity. The PACE protocol, however, achieves the average throughput within a maximum of 0.1 distance of the channel capacity. Overall, PACE achieves $20\% - 60\%$ throughput improvement over the IEEE802.11 ARQ and HARQ protocols. In addition, the PACE protocol increases the performance of ACE by 10% regardless of channel condition. This performance gain clearly illustrates the impact of code selection and the decoding scheduling policy of the PACE protocol.

Fig 7.6b suggests that the average Y-PSNR experiences a similar trend. For low BER values, the PSNR value is high and close to the ideal video quality (here $33dB$). However, the PSNR value for IEEE802.11 ARQ and HARQ degrades for channels with BER more than 0.01. The PACE protocol manages to achieve significantly better video quality regardless of channel condition. For instance, the PSNR value is around $31dB$ under PACE while ACE, HARQ, and IEEE802.11 ARQ achieve $28dB$, $27dB$, and $25dB$ over the channel with BER 0.007. Overall PACE achieves $2 - 10dB$ PSNR gain with respect to the IEEE802.11 ARQ and HARQ protocols. Further, it improves the ACE performance by $1 - 3dB$.

7.5 Discussion

In this Chapter, we introduced Prioritized Automatic Code Embedding (PACE) protocol which takes into consideration the stability, reliability, and delay constraints in wireless link-layer communication. We developed a LDPC decoding model to capture the decoding process of link-layer traffic and the PACE Buffer model to describe link-layer buffer as a multiclass priority queuing system. Using these models, we determined the optimal code selection for parity allocation and dynamic decoder scheduling for heterogeneous link-layer traffic. We showed experimentally that PACE significantly outperforms IEEE802.11 and HARQ protocols and improves the performance of ACE over various wireless channel conditions.

CHAPTER 8

Conclusions and Future Work

In this thesis, we aimed to tackle the critical issues associated with the inefficiencies of current wireless link-layer protocols and pursued a paradigm shift in the conventional 802.11 link-layer design. We developed a new link-layer framework to overcome the shortcomings of current link-layer standards and to provide both the reliability and stability for point-to-point contention free wireless communication. Using this framework, we introduced four link-layer protocols: 1) *PEEC*, a link-layer protocol designed to ensure reliable wireless communication by reducing the number of retransmissions which essentially leads to improving system throughput. PEEC is layer oblivious and uses the packet formats of current IEEE802.11 standard; 2) *DC-PEEC*, an extension of the PEEC protocol that targets the flow control of realtime video traffic in addition to reliability in wireless communication. DC-PEEC adjusts its parameters to provide low-latency communication to satisfy the delay constraint (provided by the application) while utilizing the channel bandwidth effectively; 3) *ACE*, the first effort to develop a theoretical framework for analyzing and designing a wireless link-layer protocol that targets system stability in conjunction with reliable communication. The ACE protocol uses a unique and optimal code embedding rate to construct coded link-layer packets in every transmission to ensure stability, reliability and maximum throughput; 4) *PACE*, the ACE based stable-and-reliable link-layer that employs a novel

rate-adaptive Low Density Parity Check (LDPC) channel codes while interacting with the higher layers to provide a dynamic decoder scheduling service over varying wireless channel condition. PACE provides prioritized wireless link-layer communication that takes into consideration the level of importance/impact of each packet to improve the overall performance.

Although the link-layer frameworks developed in this thesis provide a thorough groundwork that address reliability and stability issues, it is designed for a *single* point-to-point connection (e.g., between an access point and a wireless device). Meanwhile, emerging multi-hop ad-hoc and mesh wireless networks supporting heterogeneous high-end applications require significantly further research into new link-layer and corresponding cross-layer frameworks that achieve an overall ("global") reliability and stability while maximizing effective throughput and bandwidth utilization throughout the network. That is, it is necessary to ensure the information packets flow among relays nodes in a timely fashion (i.e, stability) while experiencing minimum distortions (i.e., reliability) over wireless channels with varying error conditions. Consequently, it is necessary to develop an underlying *network* of reliable and stable wireless link-layer connections that meet several critical objectives simultaneously and jointly: (1) achieving sustained stability: supporting legacy TCP applications and maintaining continuous realtime flow among dynamic and heterogeneous wireless devices, (2) ensuring maximal reliability and throughput, (3) interacting with the higher layers for prioritized communication, optimal path selection and rate-allocation, and (4) exploiting side-information for channel estimation/prediction in a distributed manner. For future work, we will build on our findings (realized in this thesis) and develop a Global Rate-Adaptive Code Embedding (GRACE) framework for a network of reliable and stable wireless link-layer communication. Under the GRACE framework, we aim to identify different challenges and research questions and our proposed solution to address these problems to achieve global reliability and stability.

8.1 Link Layer Assignments

Latency in delivering information packets (due to wireless errors) over multipath wireless systems such as ad-hoc and mesh networks is critical and has a direct impact on the overall performance of realtime and high-demand applications (e.g., video conferencing, HDTV, etc.). The link-layer frameworks developed in this thesis are designed to ensure 100% accuracy of each packet. That is, the link-layer protocol follows a *correct-it-send-it* strategy where every relay node has to verify that each packet is successfully decoded (thus passed to the next hop) or otherwise is dropped at the link-layer (marked as a lost packet). Although this design strategy guarantees the validity of information at any time throughout the path, it a) suffers from a significant end-to-end delay due to the complexity of error correction process since every packet has to be decoded completely before transmitted to the next-hop; and b) introduces unnecessary packet losses at the link-layer since partially corrupted packets (which are likely to be expired) are dropped from the link-layer buffer.

We will investigate and develop a new decoding strategy to tackle the critical issues associated with the inefficiency of the *correct-it-send-it* approach and pursue a paradigm shift in the conventional link-layer design. This approach attempts to reduce the latency while maintaining maximum throughput by partially and optimally decoding each packet in different relay nodes. The design of the proposed framework requires a *network-of-queues* model that captures the error correction process and networking effects of traffic flows over multi-hop path. In particular, we aim to find an optimal (low complex) error correction scenario for a given path to ensure a) effective utilization of channel bandwidth, b) rapid delivery of packets to the destination, and c) improvement of the overall throughput; and d) minimum information loss. Toward that end, we will develop solutions that address the following key questions:

Research question 1. *At what intermediate nodes (if any) a link-layer packet should be examined*

for being decodable or not?

Here, there are two extreme options. First, one can implement a hop-by-hop link-layer, where full rate-adaptive channel decoding/encoding is employed; and hence reliability and stability can be achieved throughout. This option may not be feasible under certain scenarios due to high-complexity and delay considerations (since it requires buffering/queuing in conjunction with iterative feedback transmissions over every single hop). The other extreme option is to perform channel decoding at the final destination only. This option may lead to high levels of redundancy since a packet needs to be protected over the whole route from the source to the destination by a single set of channel codes. The optimum solution resides between the two extreme options, depending on the network condition, the nature of the supported application and most importantly the structure of the network. For instance, the optimal GRACE assignment for a network with tree structure might not necessarily be optimal for a network with cliques. Therefore, to find the optimal GRACE assignment, first we should model the connections of the network with a graph structure. However, the network connections are wireless channels with varying error conditions and also (in case of mobile ad-hoc network) it is possible that some nodes join or leave the network. Consequently, random graphs will provide more accurate formulation of the stochastic processes in the network than the traditional graph models. Hence, to solve the GRACE assignment problem, we plan to first model a given network with a random graph. Then, we will formulate the assignment problem to find an optimal solution for the GRACE assignment $\wp^*$. The optimal solution for the GRACE assignment $\wp^*$ requires a global knowledge of network structure and channel conditions. However, this solution might not lead to a practical solution since the availability of global information may be unachievable. Consequently, the final phase of addressing this research question includes *distributed* solution for GRACE assignment that the achieves a performance close to $\wp^*$.

8.2 Code Assignment

Under the GRACE framework, a new packet is encoded using minimum (but necessary) number of type-I parity symbols. Further, if the packet is decoded using type-I parity symbols and the decoder cannot successfully retrieve the corrupted symbols in the packet, it will request for additional type-II parity symbols. The management of the transmission of type-I and type-II parity symbols is trivial for point-to-point communication scenario where the type-II parity symbols are always concatenated to future data packets. We envision that this management will not be trivial for multi-hop network.

Research question 2. *How to partition the handling of different code types at different nodes?*

Under this research question, one can envision a scenario where a packet M_0 that is carrying *type I* and *type II codes* gets segmented at a particular intermediate node n_1, where the type-II code is extracted (from M_0) because it is needed for decoding a previously transmitted packet M_1 awaiting in the buffer of that intermediate node n_1. In this case, the current packet M_0 may continue its journey toward the receiver without being decoded at node n_1. Further, M_0 may carry a new type-II code to be delivered to another intermediate node n_2 that is located on the M_0's route toward its destination. Here this new type-II code is needed for the recovery of a corresponding packet M_2 awaiting in the buffer of n_2. This example illustrates the importance of transportation plan for type-II parity symbols in the GRACE network. Suppose that we have a collection of m packets propagating throughout the network and a collection of l buffered packets in different locations of network waiting for the arrivals of type-II parity symbols. Further, let a cost function $g(y_j, M_i)$ be the cost of delivering type-II parity y_j to a packet M_i. We define a transport plan $T : \mathbf{Y} \rightarrow \mathbf{M}$ an arrangement where each type-II parity block $y_j \in \mathbf{Y}$ is delivered to its corresponding packet $T(y_j) \in \mathbf{M}$. We wish to find the optimal transport plan, the plan T^*

such that its total cost function

$$g(T^*) = \sum_{y_j \in \mathbf{M}} g(y_j, T^*(y_j)),$$

is the least of all possible transport plans.

8.3 GRACE Interactions with higher-layers

The proposed link-layer framework primarily focuses on the enhancement of the link-layer protocol and the improvement of wireless link-layer communication. Another aspect of the GRACE framework for multi-hop ad-hoc/mesh wireless networks is the design and analysis of dynamic feedback mechanism between the link-layer and the higher layers. Using this feedback mechanism, the higher layers (while interacting with link-layer) reconfigure and readapt in reaction to heterogeneous traffic fluctuations and wireless channel noise disruptions. This approach will enable higher-layers to intelligently adjust the operating parameters of their protocols based on the knowledge of wireless channel conditions passed on to them by the link-layer. This in turn will provide the source and relay wireless nodes the capability of reacting rapidly to random failures and noise over wireless links and expedites the reliable delivery of information to the destination. We will investigate the following research questions which consider the impact of the dynamic feedback at the link-layer on the performance of each of the higher layers.

Research question 3. *How is the GRACE link-layer protocol coupled with the contention control protocol at the MAC layer?*

The 802.11 family uses a Medium Access Control (MAC) mechanism [85] known as CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) [86] to provide contention control. An important objective in contention control is to ensure *fairness* among the transmitting nodes [87]. To enforce fairness, each sender is permitted to access the medium for a transmission

of a single packet. In the current IEEE802.11 standard, a wireless node that wants to transmit first listens on the desired channel. If the channel is idle (no active transmitters), then it will send a packet. However, if the channel is busy (there is another active transmitter) the node waits until transmission stops in addition to a further contention window. At the end of the contention window, if the sender senses the channel is still idle then it will transmit its packet, otherwise it repeats the carrier sense process until it gets a free channel. The receiver will return an ACK packet if it has successfully received the packet. However, if the receiver has received the packet with errors then it will not respond (i.e. there is no negative ACK). The CSMA/CA transmit window (the period that the sender is allowed to access the channel) allows for the ACK and therefore the contention period starts after the ACK is received by the sender. Under the GRACE framework, the receiver is forced to *always* send the ACK packet. This mechanism may bias the CSMA/CA mechanism toward the node that already has the medium (since all other nodes should wait until the end of the ACK transmission including additional contention window) and consequently jeopardizes the fairness. Further, the GRACE receiver is designed to store corrupted packets in its buffer for future recovery. After few contention windows different senders have transmitted their packets to the GRACE receiver. Now envision the situation where some these packets could not be decoded using type-I parities and are waiting for type-II parity symbols. Therefore, it is very likely that after few contention windows, the receiver buffer contains different packets belonging to different senders (waiting for error recovery). In this situation, each packet in the receiver is subject to wait for type-II parity symbols that should be transmitted by the corresponding sender. On the other hand, the type-II parity symbols will not arrive at the receiver unless the MAC contention mechanism would give the medium to that sender. Consequently, the decoding process at the GRACE receiver is influenced by additional delay (besides the decoding delay and the buffer delay) which is governed by the contention control mechanism. This in turn will result in reduction in throughput at the link-layer. This situation introduces a tradeoff between the con-

tention *fairness* and decoding *delay* under the GRACE framework. Consequently, it is necessary to design an optimal feedback mechanism between the GRACE link-layer and the CSMA/CA contention mechanism to ensure *maximum* contention fairness with *minimum* decoding delay at the link-layer.

Research question 4. *Can GRACE improve the performance of ad-hoc routing protocols?*

The main objective of the network layer for mobile ad-hoc/mesh network is to route the wireless packets between different senders and receivers. One of the challenging types of routing protocols is an *ad-hoc routing* protocol [80]. In ad-hoc networks, nodes do not have a prior knowledge of topology of network around them. Consequently, a new node (optionally) announces its presence and listens to broadcast announcements from its neighbors. The node learns about new near nodes and ways to reach them, and may announce that it can also reach those nodes. Extensive research has been conducted to find an efficient ad-hoc routing protocol and various routing protocols have been proposed [76]- [79]. Among these protocols, Ad-hoc On-demand Distance Vector (AODV) [81] has clear advantages in its moderate overheads and its route convergence performance and became one of the promising protocols, currently available, for the mobile ad hoc network. Under the AODV protocol nodes discover routes in request-response cycles. At each node, AODV maintains a routing table where each entry for a destination contains three essential fields: a next hop node, a *sequence-number* and a *hop-count*. All packets destined to the destination are sent to the next hop node. The sequence number acts as a form of time-stamping, and is a measure of the *freshness* of a route. The hop count represents the *current distance* to the destination node.

The AODV sender performs *path selection* when there are multiple routes available to the receiver. In this situation, the receiver selects a path associated with the minimum *hop-count*. That is, the sender chooses a route with a minimum distance to the destination. However, a path with a

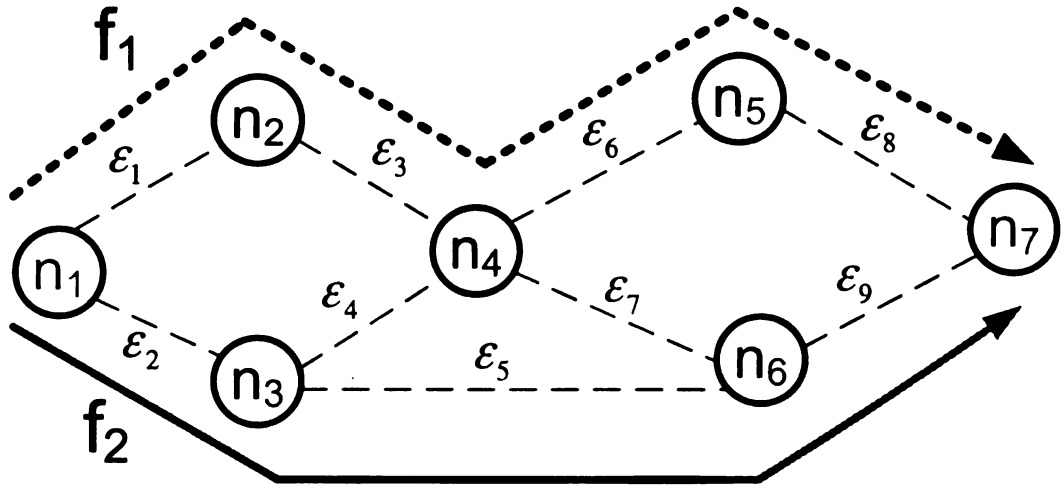


Figure 8.1 An example of multi-hop wireless network.

minimum distance is not always an optimal path. For instance consider a situation for the network depicted in Fig 8.1 where there are two path $P_1 : (n_1, n_2, n_4, n_5, n_7)$ and $P_2 : (n_1, n_3, n_6, n_7)$ from the sender n_1 to the receiver n_7. For these paths, $\epsilon_{P_1} : (\epsilon_1, \epsilon_3, \epsilon_6, \epsilon_8)$ and $\epsilon_{P_2} : (\epsilon_2, \epsilon_5, \epsilon_9)$ represent respectively the channel conditions of the routes P_1 and P_2. The standard AODV path selection policy enforces the sender to choose P_2 to communicate with the receiver since P_2 is shorter than P_1. However, assuming that the average wireless errors of P_1 is significantly lower than P_2: $\bar{\epsilon}_{P_1} \ll \bar{\epsilon}_{P_2}$, it is reasonable to select P_1.

The information about the channel conditions for different paths can result in a better path selection policy which in turn will reduce the "loss connectivity" in AODV and consequently RERR messages. To provide such information to the routing layer, we aim to establish a dynamic feedback between the GRACE link-layer to the routing layer where the channel condition is reported to the routing layer. In particular, we add a new field called *parity-usage* to each entry of the AODV routing table. The *parity-usage* field is updated by GRACE after every link-layer transmission. More specifically, the GRACE informs the routing layer the number of parity bits it employed for its coding operations. The *parity-usage* represents an average number of parity bits used in last W transmissions. The AODV sender will then employ a cost function that takes into consideration both *hop-count* and *parity-usage* and will select a path with a lower cost.

8.4 Broader Impact

The impact of the link-layer frameworks developed in this thesis is envisioned to be significantly effective in the following areas:

- **Sensor Networks and Power Constrained Systems**: In addition to enabling a major shift toward reliable and stable link-layer communications, the proposed effort will naturally have significant impact on related areas in wireless networking. For example, it is well known that sensor networks cannot afford retransmission mechanisms due to severe power constraints. Hence, we anticipate that the proposed effort could have a profound positive impact on power-constrained networks, in general, and sensor networks in particular.

- **High Demand Wireless Video Communication**: The proposed approach will have a profound impact on enabling new levels of improved performance for high-end applications such as HDTV over wireless networks. What is ultimately needed for wireless HD video is a comprehensive cross-layer framework that targets stable-and-reliable communications, exploitation of side information, and interaction with the video layer jointly. In particular, no effort has been pursued (to the best of our knowledge) of such comprehensive framework over multi-hop ad-hoc or mesh wireless networks. Consequently, a natural extension of the proposed link-layer framework (presented in this thesis) is the development of a cross-layer based stable-and-reliable link-layer that interacts with the lower physical layer and the higher video layer. We also envision realizing the proposed new link-layer approach with a high-end application over a small testbed with all supplementary softwares which can be made available to the larger research community.

- **Neurotechnology Communication Systems**: Brain-computer interfacing aims to provide new means of communication by directly assessing and interpreting brain intentional states. However, bypassing the brains biological outputs using an engineered interface comes at

the high price of beyond state-of-the-art technology. This technology should overcome important communication challenges such as a) bandwidth limitations: due to sensitivity of brain tissues, b) power limitations: due to the minute nature of the implant chips; and c) significant communication error: since it is difficult to align the sender (implant chip) and the receiver (the antenna secured in the skull). The proposed framework aim to reduce the communication errors and effectively utilize an available bandwidth while minimizing retransmissions (reducing transmission power consumptions). It is our belief that the proposed communication mechanism under the GRACE framework is well-suited for such environments and will provide a breakthrough in Neurotechnology communication.

APPENDIX A

Proof of Lemmas for PEEC

Lemma 1. *In a transmission interval τ_i with error probability ϵ_i, the ν likelihood of successful decoding of a message M_i with type-I parity probability distribution p_{X_i} can be achieved if*

$$p_{X_i} \geq \frac{1}{n\alpha}\left(n\epsilon_i + \sqrt{n\epsilon_i}\phi^{-1}(\nu)\right),$$

where $\phi^{-1}(\nu)$ is the ν-quantile of the standard normal distribution.

Proof. According to Equation (4.3), the ν likelihood of successful decoding is

$$P\{D_i \leq \alpha x_i\} = \nu.$$

Since D_i has Poisson distribution with mean λ_{D_i}, then

$$P\{D_i \leq \alpha x_i\} = \phi\left(\frac{\alpha x_i - \lambda_{D_i}}{\sqrt{\lambda_{D_i}}}\right) = \nu,$$

Thus

$$\alpha x_i = \lambda_{D_i} + \sqrt{\lambda_{D_i}}\phi^{-1}(\nu).$$

According to the message model, $x_i = np_X$, and $\lambda_{D_i} = n\epsilon_i$; so ν likelihood of successful decoding can be achieved if

$$p_X \geq \frac{1}{n\alpha}\left(n\epsilon_i + \sqrt{n\epsilon_i}\phi^{-1}(\nu)\right).$$

□

Lemma 2. *In τ_{i-1}, if a decoder with error-correcting capability α decodes a message with type-I parity symbol rate $p_{X_{i-1}} = g(\hat{\epsilon}_{i-1})$ successfully, then for τ_i, $\hat{\epsilon}_i$ has the upper bound*

$$\hat{\epsilon}_i \leq g^{-1}\left(\frac{\hat{\epsilon}_{i-1}}{\alpha}\right).$$

Proof. In τ_{i-1}, because the decoding of a message was successful, according to lemma 1, we have

$$p_{X_{i-1}} = g(\hat{\epsilon}_{i-1}) \geq \frac{1}{n\alpha}(n\hat{\epsilon}_{i-1} + \sqrt{n\hat{\epsilon}_{i-1}}\phi^{-1}(\nu)),$$

where ν is close to one (i.e., $\nu = 0.99$). By assuming very slow fading, such that the channel does not change significantly between consecutive transmissions; because of successful decoding, we assess that the actual BER for τ_i is at most $\hat{\epsilon}_{i-1}$, that is

$$\epsilon_i \leq \hat{\epsilon}_{i-1} \Rightarrow g(\epsilon_i) \leq g(\hat{\epsilon}_{i-1}),$$

given that $g(\epsilon) > \epsilon, \epsilon > 0$. Accordingly, for the estimation of ϵ_i (i.e, $\hat{\epsilon}_i$), $g(\hat{\epsilon}_i)$ is bounded by $g(\hat{\epsilon}_{i-1})$. Consequently, the maximum $\hat{\epsilon}_i$ is the one that introduces $g(\hat{\epsilon}_i)$ which satisfies the equality. Thus

$$\max \hat{\epsilon}_i = g^{-1}\left(\frac{1}{n\alpha}(n\hat{\epsilon}_{i-1} + \sqrt{n\hat{\epsilon}_{i-1}}\phi^{-1}(\nu))\right).$$

In practice, since ν is close to one, then $\phi^{-1}(\nu) \approx 8$ and n is a large number. Therefore, we can approximate the upper bound of $\hat{\epsilon}_i$ by calculating the above expression when n is infinity. That is,

$$\hat{\epsilon}_i \leq \lim_{n\to\infty} g^{-1}\left(\frac{\hat{\epsilon}_{i-1}}{\alpha} + \sqrt{\frac{\hat{\epsilon}_{i-1}\phi^{-1}(\nu)}{n}}\right).$$

Thus

$$\hat{\epsilon}_i \leq g^{-1}\left(\frac{\hat{\epsilon}_{i-1}}{\alpha}\right).$$

□

Lemma 3. *In τ_{i-1}, if a decoder with error-correcting capability α fails to decode a message with type-I parity symbol rate $p_{X_{i-1}} = g(\hat{\epsilon}_{i-1})$, then the estimation of BER for τ_i has a lower*

bound

$$\hat{\epsilon}_i \geq \alpha g(\hat{\epsilon}_{i-1}).$$

Proof. According to the distortion model, the decoding failure in τ_{i-1}, suggests that the distortion of the message was higher than its type-I redundancy. That is,

$$D_{i-1} > \alpha x_{i-1}.$$

Furthermore, the failure in decoding indicates that the actual BER ϵ_i is greater than $\hat{\epsilon}_{i-1}$. This assumption is true under very slow fading. Accordingly, the distortion created by ϵ_i in τ_i should satisfy,

$$Pr\{D_i > \alpha x_{i-1}\} > \nu,$$

where ν is close to one (i.e., $\nu = .99$) and D_i is a Poisson random variable with rate $\lambda_{D_i} = n\epsilon_i$. Thus

$$\begin{aligned} Pr\{D_i \leq \alpha x_{i-1}\} &\leq 1-\nu \\ \phi\left(\frac{\alpha x_{i-1} - \lambda_{D_i}}{\sqrt{\lambda_{D_i}}}\right) &\leq 1-\nu. \end{aligned}$$

To find a lower bound for $\hat{\epsilon}_i$, we solve the equality. That is,

$$\alpha x_{i-1} - \lambda_{D_i} = \sqrt{\lambda_{D_i}}\phi^{-1}(1-\nu).$$

By rearranging this expression, we obtain a second order equation with respect to λ_{D_i}

$$\lambda_{D_i}^2 - \lambda_{D_i}(2\alpha x_{i-1} + \phi_\nu^{-1}) + \alpha^2 x_{i-1}^2 = 0.$$

The solution of this equation leads to

$$\lambda_{D_i} = \frac{1}{2}(2\alpha x_{i-1} + \phi_\nu^{-1} \pm \sqrt{(\phi_\nu^{-1})^2 + 4\alpha x_{i-1}\phi_\nu^{-1}}.$$

Since $\lambda_{D_i} = n\epsilon_i$ and $x_{i-1} = np_{X_{i-1}} = ng(\hat{\epsilon}_{i-1})$, then the minimum $\hat{\epsilon}_i$ is

$$\hat{\epsilon}_i = \alpha g(\hat{\epsilon}_{i-1}) + \frac{\phi_\nu^{-1}}{2n} - \sqrt{\frac{(\phi_\nu^{-1})^2}{4n^2} + \frac{\alpha g(\hat{\epsilon}_{i-1})\phi_\nu^{-1}}{n}}.$$

In practice n is a large number and since ν is close to one, $\phi_\nu^{-1} \approx 8$. Consequently, we can find the lower bound for $\hat{\epsilon}_i$ by computing the limiting convergence of the above expression when n is infinity. That is

$$\hat{\epsilon}_i \geq \lim_{n \to \infty} \left(\alpha g(\hat{\epsilon}_{i-1}) + \frac{\phi_\nu^{-1}}{2n} - \sqrt{\frac{(\phi_\nu^{-1})^2}{4n^2} + \frac{\alpha g(\hat{\epsilon}_{i-1}) \phi_\nu^{-1}}{n}} \right)$$

Thus

$$\hat{\epsilon}_i \geq \alpha g(\hat{\epsilon}_{i-1}).$$

□

APPENDIX B

Proof of Lemmas for ACE

Lemma 1. *The operational code embedding rate that ensures reliability in wireless transmission over a channel in state S_i is bounded above by*

$$R_i \leq 1 - \frac{\epsilon_i}{\alpha}. \quad \epsilon_i \ll \alpha$$

where α is error-correcting capability of a decoder.

Proof. The optimization problem in (5.15) is a convex problem with the Lagrangian in a from of

$$\begin{aligned} L_i(k_i, \lambda_1, \lambda_2) = k_i + \lambda_1 \left(k_i \epsilon_i + x_i(\epsilon_i - \alpha)\right) \\ + \lambda_2 \left(k_i + x_i - n_i\right). \lambda_1, \lambda_2 > 0 \end{aligned} \tag{B.1}$$

Since the primal problem is convex, the Karush-Kuhn-Tucker (KKT) conditions are sufficient for the points to be primal and dual optimal (zero duality gap). KKT conditions suggest that based on complimentary slackness property for strong duality, we have

$$\lambda_1 \left(k_i^* \epsilon_i + x_i^*(\epsilon_i - \alpha)\right) = 0, \tag{B.2}$$

$$\lambda_2 \left(n_i - k_i^* + x_i^*\right) = 0, \tag{B.3}$$

where k_i^* and x_i^* are the optimal transmitted amount of data and parity symbols. On the other hand, with the channel is in state S_i, and maximum network utilization of n_i symbols, the amount

of transmitted data symbols is bounded above by $k_i = |C_i|R_i \leq k_i^*$, where R_i is a channel coding rate. So, by substituting $k_i^* = n_i R_i^*, x_i^* = n_i(1 - R_i^*)$ and solve the above equations for R_i^*, we have $R_i \leq R_i^* = 1 - \frac{\epsilon_i}{\alpha}$. □

Lemma 2. *The operational code embedding rate that ensures stability in wireless transmission over a channel in state S_i has a lower bound*

$$R_i \geq 1 - \frac{\epsilon_i}{\alpha}. \quad \epsilon_i \ll \alpha$$

Proof. To guarantee the stability condition, the buffer has to be always non-empty and at the same time does not overflow. To ensure that the buffer is always non-empty, the buffer length limiting distributions should not carry any density at the value zero (i.e,$F(0,j) = 0, j = 1,2$). By substituting $b = 0$ in equations (6.9)(6.10) we have

$$F(0,1) = 0 \quad F(0,2) = \omega_i - \frac{\beta_i(k_i - c)}{c}$$

The steady state probability of the buffer at length zero is always zero when the decoding is successful at the link layer (e.g., $Z(t) = 1$). To ensure that the buffer is non-empty when the link layer fails to decode new data ($Z(t) = 2$), the following equality has to be satisfied

$$\omega_i = \frac{k_i - c}{k_i}. \tag{B.4}$$

To ensure that the buffer does not overflow, the stability condition in equation (6.11) should hold. By using equations (6.8) and (6.11), we obtain $F_{E_i}(\alpha x_i) = 1 - \frac{c}{k_i} \leq \frac{1}{2}$. Therefore, $x_i \leq \frac{F_{E_i}^{-1}(0.5)}{\alpha} = \frac{\lambda_{E_i}}{\alpha} = \frac{|C_i|\epsilon_i}{\alpha}$. Correspondingly, by substituting $x_i = |C_i|(1 - R_i)$, the lower bound of code embedding rate is $R_i \geq 1 - \frac{\epsilon_i}{\alpha}$. □

APPENDIX C

Proof of Lemmas for PACE

Lemma 1. *For a packet transmitted over BSC with cross-over probability ϵ_i that is decoded using LDPC check matrix with parameters d_v and d_c; the distortion level after m iterations can be approximated by*

$$\epsilon_i^{(m)} \approx \epsilon_i \left[\epsilon_i(d_v - 1)(d_c - 1)\right]^{m-1}$$

Proof. Equation (7.1) can be simplified significantly with following approximations:

$$
\begin{aligned}
\epsilon_i^{(m)} &= \epsilon_i^{(0)} - \epsilon_i^{(0)} \left[\frac{1 + (1 - 2\epsilon_i^{(m-1)})^{d_C - 1}}{2} \right]^{d_V - 1} \\
&+ (1 - \epsilon_i^{(0)}) \left[\frac{1 - (1 - 2\epsilon_i^{(m-1)})^{d_C - 1}}{2} \right]^{d_V - 1} \\
&\approx \epsilon_i^{(0)} - \epsilon_i^{(0)} \left[\frac{1 + 1 - (d_C - 1) 2\epsilon_i^{(m-1)}}{2} \right]^{d_V - 1} \\
&+ (1 - \epsilon_i^{(0)}) \left[\frac{1 + 1 + (d_C - 1) 2\epsilon_i^{(m-1)}}{2} \right]^{d_V - 1} \\
&= \epsilon_i^{(0)} - \epsilon_i^{(0)} \left[1 - (d_C - 1)\epsilon_i^{(m-1)} \right]^{d_V - 1} + (1 - \epsilon_i^{(0)}) \left[1 + (d_C - 1)\epsilon_i^{(m-1)} \right]^{d_V - 1} \\
&\approx \epsilon_i^{(0)} - \epsilon_i^{(0)} \left[1 - (d_C - 1)\epsilon_i^{(m-1)} \right]^{d_V - 1} \\
&= \epsilon_i^{(0)} - \epsilon_i^{(0)} \left[1 - (d_V - 1)(d_C - 1)\epsilon_i^{(m-1)} \right] \\
&= \epsilon_i^{(0)} (d_V - 1)(d_C - 1)\epsilon_i^{(m-1)} = \xi \epsilon_i^{(m-1)},
\end{aligned}
$$

where $\xi = \epsilon_i^{(0)}(d_V - 1)(d_C - 1)$. By taking a unilateral Z-transform, we have,

$$
z\Upsilon_i(z) - \epsilon_i^{(0)} = \xi \Upsilon_i(z) \rightarrow \Upsilon_i(z) = \frac{z^{-1}\epsilon_i^{(0)}}{1 - z^{-1}} \tag{C.1}
$$

Now, the inverse Z-transform $u[.]$ gives us the follows:

$$
\epsilon_i^{(m)} = \epsilon_i^{(0)} \xi^{-1} u[m] = \epsilon_i^0 \left[\epsilon_i^0 (d_V - 1)(d_C - 1) \right]^{m-1}. \tag{C.2}
$$

This completes the proof. □

APPENDIX D

Steady State Balance Equations

The following equations are the list of balance equation for the proposed queueing system under PEEC protocol. In the following equations $1 \leq m \leq B_1 - 1$, $1 \leq n \leq B_2 - 1$ and $1 \leq l \leq B_3 - 1$.

$$\lambda \pi_{0,0,0} = q\mu_2 \pi_{0,1,0} + \mu_3 \pi_{0,0,1}. \tag{D.1}$$

$$(\lambda + \mu_1)\pi_{m,0,0} = \lambda \pi_{m-1,0,0} + q\mu_2 \pi_{m,1,0} + \mu_3 \pi_{m,0,1}. \tag{D.2}$$

$$\mu_1 \pi_{B_1,0,0} = \lambda \pi_{B_1-1,0,0} + q\mu_2 \pi_{B_1,1,0} + \mu_3 \pi_{B_1,0,1}. \tag{D.3}$$

$$(\lambda + \mu_2)\pi_{0,n,0} = \mu_1 \pi_{1,n-1,0} + q\mu_2 \pi_{0,n+1,0} + \mu_3 \pi_{0,n,1}. \tag{D.4}$$

$$(\lambda + \mu_2)\pi_{0,B_2,0} = \mu_1 \pi_{1,B_2-1,0} + \mu_3 \pi_{0,B_2,1}. \tag{D.5}$$

$$(\lambda + \mu_3)\pi_{0,0,l} = p\mu_2 \pi_{0,1,l-1} + q\mu_2 \pi_{0,1,l} + \mu_3 \pi_{0,0,l+1}. \tag{D.6}$$

$$(\lambda + \mu_3)\pi_{0,0,B_3} = p\mu_2 \pi_{0,1,B_3-1} + q\mu_2 \pi_{0,1,B_3}. \tag{D.7}$$

$$(\lambda + \mu_1 + \mu_2)\pi_{m,n,0} = \lambda \pi_{m-1,n,0} + \mu_1 \pi_{m+1,n-1,0} + q\mu_2 \pi_{m,n+1,0} + \mu_3 \pi_{m,n,1}. \tag{D.8}$$

$$(\lambda + \mu_2)\pi_{m,B_2,0} = \lambda_{m-1,B_2,0} + \mu_1 \pi_{m+1,B_2-1,0} + \mu_3 \pi_{m,B_2,1}. \tag{D.9}$$

$$(\lambda + \mu_1 + \mu_3)\pi_{m,0,l} = \lambda \pi_{m-1,0,l} + q\mu_2 \pi_{m,1,l} + p\mu_2 \pi_{m,1,l-1} + \mu_3 \pi_{m,0,l+1}. \tag{D.10}$$

$$(\lambda + \mu_1 + \mu_3)\pi_{m,0,B_3} = \lambda \pi_{m-1,0,B_3} + q\mu \pi_{m,1,B_3} + p\mu_2 \pi_{m,1,B_3-1}. \tag{D.11}$$

$$(\mu_1+\mu_2)\pi_{B_1,n,0} = \lambda\pi_{B_1-1,n,0} + q\mu_2\pi_{B_1,n+1,0} + \mu_3\pi_{B_1,n,1}. \quad \text{(D.12)}$$

$$\mu_2\pi_{B_1,B_2,0} = \lambda\pi_{B_1-1,B_2,0} + \mu_3\pi_{B_1,B_2,1}. \quad \text{(D.13)}$$

$$(\mu_1+\mu_3)\pi_{B_1,0,l} = \lambda\pi_{B_1-1,0,l} + q\mu_2\pi_{B_1,1,l} + p\mu_2\pi_{B_1,1,l-1} + \mu_3\pi_{B_1,0,l+1}. \quad \text{(D.14)}$$

$$(\mu_1+\mu_3)\pi_{B_1,0,B_3} = \lambda\pi_{B_1-1,0,B_3} + p\mu_2\pi_{B_1,1,B_3-1} + q\mu_2\pi_{B_1,1,B_3}. \quad \text{(D.15)}$$

$$(\lambda+\mu_2+\mu_3)\pi_{0,n,l} = \mu_1\pi_{1,n-1,l} + p\mu\pi_{0,n+1,l-1} + \mu_3\pi_{0,n,l+1}. \quad \text{(D.16)}$$

$$(\lambda+q\mu_2+\mu_3)\pi_{0,n,B_3} = \mu_1\pi_{1,n-1,B_3} + q\mu_2\pi_{0,n+1,B_3} + p\mu_2\pi_{0,n+1,B_3-1}. \quad \text{(D.17)}$$

$$(\lambda+\mu_2+\mu_3)\pi_{0,B_2,l} = \mu_1\pi_{1,B_2-1,l} + \mu_3\pi_{0,B_2,l+1}. \quad \text{(D.18)}$$

$$(\lambda+q\mu_2+\mu_3)\pi_{0,B_2,B_3} = \mu_1\pi_{1,B_2-1,B_3}. \quad \text{(D.19)}$$

$$(\mu_1+\mu_2+\mu_3)\pi_{B_1,n,l} = \lambda\pi_{B_1-1,n,l} + q\mu_2\pi_{B_1,n+1,l}p\mu_2\pi_{B_1,n+1,l-1} + \mu_3\pi_{B_1,n,l+1}. \quad \text{(D.20)}$$

$$(\mu_1+q\mu_2+\mu_3)\pi_{B_1,n,B_3} = \lambda\pi_{B_1-1,n,B_3}+q\mu_2\pi_{B_1,n+1,B_3}+p\mu_2\pi_{B_1,n+1,B_3-1}. \quad \text{(D.21)}$$

$$(\lambda+\mu_2+\mu_3)\pi_{m,B_2,l} = \lambda\pi_{m-1,B_2,l} + \mu_1\pi_{m+1,B_2-1,l} + \mu_3\pi_{m,B_2,l+1}. \quad \text{(D.22)}$$

$$(\lambda+q\mu_2+\mu_3)\pi_{m,B_2,B_3} = \lambda\pi_{m-1,B_2,B_3} + \mu_1\pi_{m+1,B_2-1,B_3}. \quad \text{(D.23)}$$

$$(\mu_2+\mu_3)\pi_{B_1,B_2,l} = \lambda\pi_{B_1-1,B_2,l} + \mu_3\pi_{B_1,B_2,l+1}. \quad \text{(D.24)}$$

$$(\lambda+\mu_1+q\mu_2+\mu_3)\pi_{m,n,B_3} = \lambda\pi_{m-1,n,B_3} + \mu_1\pi_{m+1,n-1,B_3} + q\mu_2\pi_{m,n+1,B_3} + p\mu_2\pi_{m,n+1,B_3-1}. \quad \text{(D.25)}$$

$$(q\mu_2+\mu_3)\pi_{B_1,B_2,B_3} = \lambda\pi_{B_1-1,B_2,B_3}. \quad \text{(D.26)}$$

$$(\lambda+\mu_1+\mu_2+\mu_3)\pi_{m,n,l} = \lambda\pi_{m-1,n,l} + \mu_1\pi_{m+1,n-1,l} + q\mu_2\pi_{m,n+1,l} + p\mu_2\pi_{m,n+1,l-1} + \mu_3\pi_{m,n,l+1}. \quad \text{(D.27)}$$

The following equations are the list of balance equations for the proposed queueing system under IEEE802.11 ARQ scheme. In these equations $1 \le m \le B_1 - 1$ and $1 \le n \le B_2 - 1$.

$$\lambda\pi_{0,0} = \mu_2\pi_{0,1}. \quad \text{(D.28)}$$

$$(\lambda + \mu_1)\pi_{m,0} = \lambda\pi_{m-1,0} + \mu_2\pi_{m,1}. \tag{D.29}$$

$$\mu_1\pi_{B_1,0} = \lambda\pi_{B_1-1,0} + \mu_2\pi_{B_1,1}. \tag{D.30}$$

$$(\lambda + \mu_2)\pi_{0,n} = \mu_1\pi_{1,n-1} + \mu_2\pi_{0,n+1}. \tag{D.31}$$

$$(\lambda + \mu_2)\pi_{0,B_2} = \mu_1\pi_{1,B_2-1}. \tag{D.32}$$

$$(\mu_1 + \mu_2)\pi_{B_1,n} = \lambda\pi_{B_1-1,n} + \mu_2\pi_{B_1,n+1}. \tag{D.33}$$

$$\mu_2\pi_{B_1,B_2} = \lambda\pi_{B_1-1,B_2}. \tag{D.34}$$

$$(\lambda + \mu_2)\pi_{m,B_2} = \lambda\pi_{m-1,B_2} + \mu_1\pi_{m+1,B_2-1}. \tag{D.35}$$

$$(\lambda + \mu_1 + \mu_2)\pi_{m,n} = \lambda\pi_{m-1,n} + \mu_1\pi_{m+1,n-1} + \mu_2\pi_{m,n+1}. \tag{D.36}$$

BIBLIOGRAPHY

[1] Clark, George C., Jr., and J. Bibb Cain. "Error-Correction Coding for Digital Communications." New York, Plenum Press, 1981. ISBN 0-306-40615-2.

[2] S. Soltani, K. Misra, and H. Radha, "On Link-Layer Reliability and Stability for Wireless Communication," ACM MOBICOM, 2008.

[3] S. Soltani, H. Radha, "PEEC: A Channel-Adaptive Feedback-Based Error Control Protocol for Wireless MAC Layer," *IEEE JSAC Special Issue on Exploiting Limited Feedback in Tomorrows Wireless Communication Networks*, 2008.

[4] Syed A. Khayam and Hayder Radha, "Constant-Complexity Models for Wireless Channels," IEEE INFOCOM, April 2006. [50]

[5] S. S. Karande and H. Radha, "Hybrid Erasure-Error Protocols for Wireless Video," IEEE Transactions on Multimedia, vol. 9, no. 2, Feb 2007.

[6] S. Karande, S. A. Khayam, Y. Cho, K. Misra, H. Radha, J. Kim and J. Hong, "On Channel State Inference and Prediction Using Observable Variables in 802.11b Networks," *IEEE International Conference on Communications (ICC)*, Glasgow, UK, June 2007.

[7] Syed Ali Khayam, Shirish Karande, Muhammad Usman Ilyas, and Hayder Radha, "Header Detection to Improve Multimedia Quality over Wireless Networks," to appear in IEEE Transactions on Multimedia.

[8] Syed Ali Khayam, Hayder Radha, Selin Aviyente, and John R. Deller, Jr., "Markov and Multifractal Wavelet Models for Wireless MAC-to-MAC Channels," Elsevier Performance Evaluation Journal.

[9] Shirish Karande, Utpal Parrikar, Kiran Misra, and Hayder Radha, "Utilizing Signal to Silence Ratio indications for improved Video Communication in presence of 802.11b Residue Errors ", IEEE International Conference on Multimedia & Expo (ICME), July 2006.

[10] Shirish Karande, U. Parrikar, Kiran Misra, and Hayder Radha, "On Modeling of 802.11b Residue Errors," Conference on Information Sciences & Systems (CISS), March 2006.

[11] Syed Ali Khayam, Shirish Karande, Muhammad Usman Ilyas, and Hayder Radha, "Improving Wireless Multimedia Quality using Header Detection with Priors," IEEE International Conference on Communications (ICC), June 2006.

[12] Syed Ali Khayam and Hayder Radha, "Linear-Complexity Models for Wireless MAC-to-MAC Channels," ACM/Kluwer Wireless Networks (WINET) Journal - Special Issue on Selected Papers from MSWiM'03, vol. 11, no. 5, pp. 543-555, September 2005.

[13] Syed Ali Khayam, Selin Aviyente, and Hayder Radha, "On Long-Range Dependence in High-Bitrate Wireless Residual Channels," Conference on Information Sciences and Systems (CISS), March 2005.

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

[15] Syed Ali Khayam, Muhammad U. Ilyas, Klaus Prsch, Shirish Karande, and Hayder Radha, "A Statistical Receiver-based Approach for Improved Throughput of Multimedia Communications over Wireless LANs," IEEE International Conference on Communications (ICC), May 2005.

[16] Shirish Karande and Hayder Radha, "Does Relay of Corrupted Packets Lead to Capacity Improvement?," IEEE Wireless Communications and Networking Conference (WCNC), March 2005.

[17] Shirish Karande and Hayder Radha, "Rate-Constrained Adaptive FEC for Video over Erasure Channels with Memory," IEEE International Conference on Image Processing (ICIP), October 2004.

[18] Syed Ali Khayam and Hayder Radha, "Markov-based Modeling of Wireless Local Area Networks," ACM Mobicom International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), September 2003.

[19] Syed Ali Khayam, Shirish Karande, Hayder Radha, and Dmitri Loguinov, "Analysis and Modeling of Errors and Losses over 802.11b LANs for High-Bitrate Real-Time Multimedia," EURASIP Signal Processing: Image Communication, vol.18, no.7, pp. 575-595, August 2003.

[20] Syed Ali Khayam, Shirish S. Karande, Michael Krappel, and Hayder Radha, "Cross-Layer Protocol Design for Real-Time Multimedia Applications over 802.11b Networks," IEEE International Conference on Multimedia and Expo (ICME), July 2003.

[21] Shirish Karande, Syed Ali Khayam, Michael Krappel, and Hayder Radha, "Analysis and Modeling of Errors at the 802.11b Link-Layer," IEEE International Conference on Multimedia and Expo (ICME), July 2003.

[22] Syed Irtiza Ali and Hayder Radha, "Hierarchical Handoff Schemes over Wireless LAN/WAN Networks for Multimedia Applications," IEEE International Conference on Multimedia and Expo (ICME), July 2003.

[23] Kiran Misra, Shirish Karande, Hayder Radha, "Maximal Recovery Network Coding under Topology Constraint," Proceedings of the 27th IEEE International Conference on Computer Communications (INFOCOM'08), Phoenix, AZ, United States, April, 2008.

[24] Kiran Misra, Shirish Karande, Keyur Desai, Hayder Radha, "Complexity Reduction Using Power-Law Based Scheduling For Exploiting Spatial Correlation In Distributed Video Coding," Proceedings of IEEE Conference on Image Processing (ICIP08), San Diego, USA, 2008.

[25] Muhammad U. Ilyas, and Hayder Radha, "Measurement Based Analysis and Modeling of the Error Process in IEEE 802.15.4 LR-WPANs," Proceedings of the 27th IEEE International Conference on Computer Communications (INFOCOM'08), Phoenix, AZ, United States, April, 2008

[26] Shirish Karande, Kiran Misra, Sohraab Soltani, Hayder Radha, "Design and Analysis of Generalized LT-codes using Colored Ripples," Proceedings of International Symposium on Information Theory (ISIT08), Toronto, Canada, 2008.

[27] Sohraab Soltani and Hayder Radha, "Performance Evaluation of Error Control Protocols over Finite-State Markovian Channels", Proceedings of the Conference of Information Sciences and Systems (CISS08), Princeton University, NJ, USA, March 2008

[28] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. "Link-level Measurements from an 802.11b Mesh Network". In SIGCOMM, 2004.

[29] IEEE Computer Society LAN MAN Standard Committee, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifcations," IEEE Std. 802.11-1999, New York, 1999.

[30] ISO/IEC 8802-11:1999(E), "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," August 1999.

[31] IEEE Std 802.11b-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz band," September 1999.

[32] J. G. Kim and M. M. Krunz, "Delay analysis of selective repeat ARQ for a Markovian source over a wireless channel," *IEEE Trans. Veh. Technol.*, vol. 49, no. 5, pp. 19681981, Sep. 2000

[33] P. S. Sindhu, "Retransmission error control with memory", *IEEE Transactions on Communications*, vol. COM-25, no. 5, pp. 473.479, May 1977.

[34] S. S. Chakraborty, E. Yli-Juuti, and M. Liinaharja, "An adaptive ARQ scheme with packet combining," *IEEE Communications Letters*, vol. 2, no. 7, pp. 200.202, July 1998.

[35] M. Gidlund, "Receiver-based packet combining in IEEE 802.11a wireless LAN," *in Proc. IEEE Radio and Wireless Conference (RAWCON)*, August 2003, pp. 47.50.

[36] Y. Liang and S. S. Chakraborty, "ARQ and packet combining with post-reception selection diversity," *in Proc. 60th IEEE Semiannual Vehicular Technology Conference (VTC Fall)*, 2004.

[37] Q. Zhang and S. A. Kassam, "Hybrid ARQ with selective combining for fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 5, pp. 867.874, May 1999.

[38] T. W. A. Avudainayagam, J.M. Shea and L. Xin. Reliability Exchange Schemes for Iterative Packet Combining in Distributed Arrays. *Proc. of the IEEE WCNC*, volume 2, pages 832-837, 2003.

[39] S. S. Chakraborty, E. Yli-Juuti, and M. Liinaharja. An ARQ Scheme with Packet Combining. *IEEE Comm. Letters*, 1998.

[40] H. Yomo, S. S. Chakraborty, and R. Prasad, "IEEE 802.11 WLAN with Packet Combining", International Conference on Computer and Device 2004 (CODEC-04), January, 2004, Kolkata, India

[41] Grace Woo, Pouya Kheradpour, Dawei Shen, and Dina Katabi, "Beyond the Bits: Cooperative Packet Recovery Using Physical Layer Information," ACM MOBICOM, 2007.

[42] K. C. Lin, N. Kushman, and D. Katabi. Ziptx: Harnessing partial packets in 802.11 networks. In Mobicom08, September 2008.

[43] E. Soljanin. Hybrid ARQ in Wireless Networks. *DIMACS Workshop on Network Inform. Theory*, March 2003.

[44] E. C. Strinati, S. Simoens, and J. Boutros, "Performance evaluation of some Hybrid ARQ schemes in IEEE 802.11a Networks", *IEEE VTC*, 4(4):2735- 2739, 2003

[45] G. Caire and D. "Tuninetti. The throughput of Hybird-ARQ protocols for the Gaussion collision channel". *IEEE Trans. Inform. Theory*, July 2001.

[46] S. Cheng and M. C. Valenti. "Macrodiversity packet combining for the ieee 802.11a uplink". In IEEE WCNC, 2005.

[47] M. C. Valenti. "Improving uplink performance by macrodiversity combining packets from adjacent access points". IEEE WCNC, pages 636 641, 2003.

[48] S. Lin and D. J. Costello Jr., "Error Control Coding: Fundamentals and Applications," Englewood Cliffs, NJ: Prentice-Hall, 1983.

[49] S. Lin and P. S. Yu, "A hybrid ARQ scheme with parity retransmission for error control of satellite channels," IEEE Trans. Commun., vol. 30, pp. 17011719, July 1982.

[50] Y. Wang and S Lin, "A modified selective-repeat type-II hybrid ARQ system and its performance analyses," *IEEE Transactions on Communications* 31(5), pp. 124-133, 1983.

[51] G. Caire and D. Tuninetti, "The throughput of Hybird-ARQ protocols for the Gaussion collision channel", *IEEE Trans. Inform. Theory*, 47:19711988, July 2001.

[52] D. Chase, "Code-combining: A maximum likelihood decoding approach for combining an arbitrary number of noisy packets," IEEE Trans. Commun., vol. COMM-33, no. 5, pp. 385393, May 1985.

[53] J. C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for internet telephony," in Proc. IEEE INFOCOM 99, 1999, vol. 3, pp. 14531460.

[54] K. Jamieson and H. Balakrishnan. "PPR: Partial Packet Recovery for Wireless Networks". In ACM SIGCOMM, Kyoto, Japan, August 2007.

[55] T.M. Cover, and J. A. Thomas. Elements of Information Theory. *Wiley*, 1991.

[56] J. Ha, J. Kim, and S.W. McLaughlin. Rate-compatible puncturing of low-density parity-check codes. *IEEE Trans. Inform. Theory*, 50:2824-2836, November 2004.

[57] Boris Bellalta I Jimenez and Alexandre Graell i Amat. Performance Analysis of a Type 2 Hybrid ARQ Protocol Based on RCPC Codes for 150 - the IEEE802.11a Random Access MAC Protocol. *IEEE Trans. Comm.*, May 2005.

[58] J. Li and K. Narayanan, "Rate-compatible low-density parity-check codes for capacity-approaching ARQ scheme in packet data communications", *Int. Conf. on Comm., Internet, and Info. Tech. (CIIT)*, Nov. 2002.

[59] J. Ha, J. Kim, and S.W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes", *IEEE Trans. Inform. Theory*, 50:28242836, November 2004.

[60] D.N. Rowitch and L.B. Milstein, "On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes", *IEEE Trans. Commun.*, 48:948959, June 2000.

[61] M.R. Yazdani and A.H. Banihashemi, "On construction of rate-compatible low-density parity-check codes", *IEEE Commun. Lett.*, 8:159161, March 2004.

[62] J. Hagenauer, "Rate-compatible punctured convolutional codes and their applications," IEEE Trans. Commun., vol.36, no. 4, Apr. 1988.

[63] A. Banerjee, D. Costello, and T. Fuja, "Diversity combining techniques for bandwidth-efficient turbo ARQ systems," in IEEE Int. Symp. Information Theory, Washington, DC, Jun. 2001, p. 213.

[64] V. Guruswami and M. Sudan. "Reflections on Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes". IEEE Information Theory Society Newsletter, 52(1):612, 2002.

[65] J. Hagenauer and P. Hoecher. "A Viterbi Algorithm with Soft-Decision Outputs and its Applications". In IEEE GLOBECOM, 1989.

[66] R. M. Tanner, "A recursive approach to low complexity codes," IEEE Trans. Inform. Theory, vol. IT-27, pp. 533-547, Sept. 1981.

[67] S.B. Wicker and V.K. Bhargava. Reed-Solomon Codes and Their Applications. *IEEE Press*, 1994.

[68] R. G. Gallager, "Low Density Parity Check Codes," Trans. of the IRE Prof. G. Info. Theo., vol. IT-8, January 1962, pp. 21-28.

[69] W. E. Ryan, "An introduction to LDPC codes," August 2003.

[70] S. Karande, "A Simple Relationship between Iterations and Error," Private Communication, 2008.

[71] S. M. Ross, "Introduction to Probability Models", Acad. Press. 2003.

[72] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," Comput. Commun. Rev., vol. 24, no. 2, pp. 2436, Apr. 1997.

[73] L. Larzon, M. Degermark, and S. Pink, "Efficient Use of Wireless Bandwidth for Multimedia Applications," *IEEE International Workshop on Mobile Multimedia Communications (MoMUC)*, November 1999.

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

[75] S. Shakkottai, T. Rappaport, and P. Karlsson, "Cross- Layer Design for Wireless Networks," IEEE Commun. Mag., vol. 41, no. 10, Oct. 2003, pp. 7480.

[76] P. C. Ng, and S. C. Liew. Re-routing instability in IEEE 802.11 multi-hop ad-hoc networks. *IEEE LCN*, pages 602-609, 2004.

[77] S. Biswas and R. Morris. "Opportunistic routing in multi-hop wireless networks". ACM SIGCOMM, 2005.

[78] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. "Trading structure for randomness in wireless opportunistic routing". In Proc. of ACM SIGCOMM 2007, Kyoto, Japan.

[79] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. "A high-throughput path metric for multi-hop wireless routing". In ACM MobiCom 03, San Diego, California, September 2003.

[80] J. Broch , D. A. Maltz , D. B. Johnson , Y. Hu , J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols", Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, p.85-97, October 25-30, 1998, Dallas, Texas.

[81] C. E. Perkins and E. M. Royer. "Ad hoc on demand distance vector (AODV) routing", 1998. Internet Draft.

[82] G. Xylomenos, G.C. Polyzos, P. Mahonen, and M. Saaranen. TCP performance issues over wireless links. *IEEE Comm. Magazine*, 39(4):52-58, 2001.

[83] V. Jacobson. Congestion Avoidance and Control. In *Proc. ACM Sigcomm*'88, Aug. 1988, pp- 314-329.

[84] Y. Tian, K. Xu, and N. Ansari. TCP in Wireless Environments: Problems and Solutions. *IEEE Radio Communication*, 43(3): S27-S32, March 2005.

[85] "IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", Nov. 1997, P802.11.

[86] T. S. Ho and K. C. Chen, "Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LAN's," in Proc. IEEE PIMRC Taipei, Taiwan, Oct. 1996, pp. 392-396.

[87] X. Wang and K. Kar, "Throughput Modelling and Fairness Issues in CSMA/CA Based Ad-Hoc Networks," in INFOCOM, Miami, 2005.

[88] X. Liu, E. K. P. Chong, and N. B. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," IEEE J. Sel. Areas Commun., vol. 19, no. 10, pp. 20532064, Oct. 2001.

[89] M. Andrews, K. Kumaran, K. Ramanan, A. L. Stolyar, R. Vijayakumar, and P. Whiting, "Providing quality of service over a shared wireless link," IEEE Commun. Mag., vol. 39, no. 2, pp. 150154, Feb. 2001.

[90] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. K. Tripathi, "Enhancing throughput over wireless LANs using channel state dependent packet scheduling," IEEE INFOCOM 1996.

[91] S. Shakkottai and R. Srikant, "Scheduling real-time traffic with deadlines over a wireless channel," 2002 ACM Wireless Netw. J.

[92] J. Huang, R. A. Berry, and M. L. Honig, "Wireless Scheduling With Hybrid ARQ", IEEE Trans. Wire. Commun. vol. 4, no. 6, 2005.

[93] M. Horowitz, A. Joch, and F. Kossentini, "H.264/AVC Baseline Profile Decoder Complexity Analysis," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 13, no. 7, July 2003

[94] M. van der Schaar and S. Shankar, "Cross-Layer Wireless Multimedia Transmission: Challenges, Principles, and New Paradigms," IEEE Wireless Commun., vol. 12, no. 4, Aug. 2005, pp. 5058.

[95] J.A. Zhao, B.Li, C. Kok, and I.Ahmad, "MPEG-4 Video Transmission over Wireless: A Link Level Performance Study," ACM/Kluwer Journal of Wireless Networks, vol. 10, issue 2, March 2004, pp. 130-146.

[96] I. E. G. Richardson, "Video Codec Design: Developing Image and Video Compression Systems", WIELY, 2002.

[97] D. Mitra. Stochastic Fluid Models. *Perfomance 87*, Elsevier Science Publisher, North Holland 1988.

[98] H. Chen and D. D. Yao, "Fundamentals of Queueing Networks", Springer.

[99] M. Mandelbaum, M. Hlynka, and P. H. Brill,"Nonhomogeneous geometric distributions with relations to birth and death processes," Springer journal TOP on Business and Economics, July 2007.

[100] E. Plambeck, S. Kumar, and J. M. Harrison, " A Multiclass Queue in Heavy Traffic with Throughput Time Constraints: Asymptotically Optimal Dynamic Controls," Queu. Sys. Springer J. vol. 39, No. 1, 2001.

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

[102] J. G. Kim and M. M. Krunz, "Delay analysis of selective repeat ARQ for a Markovian source over a wireless channel," *IEEE Trans. Veh. Technol.*, vol. 49, no. 5, pp. 19681981, Sep. 2000

[103] "Reed-Solomon source code", `http://www.eccpage.com/`

[104] "LDPC Software", `http://www.cs.utoronto.ca/radford/ldpc.software.html`

[105] OMNeT++ Community, `http://www.omnetpp.org`.

[106] H.264/AVC Software Coordination `http://iphome.hhi.de/suehring/tml/`