On Link-Layer Reliability and Stability for Wireless Communication*

Sohraab Soltani[†‡], Kiran Misra[†], and Hayder Radha[†]
[†] WAVES Laboratory, Department of Electrical & Computer Engineering
[‡]Department of Computer Science & Engineering
Michigan State University
East Lansing, MI, USA
{soltanis, misrakir, radha}@msu.edu

ABSTRACT

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 design strategy has led to a great deal of inefficiency in throughput and to other major issues (such as the well-known TCP over-wireless performance degradation phenomenon and the numerous studies in attempt to fix it). In this paper, we propose 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. We present two distinct analytical frameworks to determine optimal code embedding rates which ensure system reliability and stability for wide range of traffic demand. An important conclusion of our analysis is that various traffic demand can be met using a packet-by-packet code embedding rate constraint that is independent of traffic type. We demonstrate experimentally that ACE provides both rapid and reliable point-to-point wireless data transmission for *realtime* and *non-realtime* traffic over real channel traces collected on 802.11b WLAN. We also have conducted extensive TCP simulations in conjunction with ACE; and we demonstrate the high level of efficiency and stability that can be achieved for TCP over ACE, while not making any changes to TCP. Further, the implementation of ACE for real-time video communication shows performance gains of 5-10dB over IEEE ARQ schemes. More importantly, ACE is layer oblivious and requires no changes to higher or lower PHY layers.

*This work was supported in part by NSF Award CNS-0721550, NSF Award CCF 0728996, NSF Award CCF-0515253, and NSF Award CNS-0430436.

MobiCom'08, September 14–19, 2008, San Francisco, California, USA.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks

General Terms

Algorithm, Design, Reliability, Performance

Keywords

Wireless Network, Link-layer Protocol, Stability, Reliability

1. INTRODUCTION

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 [4]. Similarly, wireless errors are interpreted as congestion by the transport-layer resulting in an unnecessary drop in transmission rate [5, 7].

Current IEEE802.11 ARQ protocol [1], 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 [8, 10]. 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 [7] in attempt to mitigate the shortcoming of the lower layers.

In this paper, 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.

We evaluate our proposed link-layer protocol under varying channel conditions over real channel traces collected on 802.11b WLAN. We implemented ACE using OMNET++ network simulator [26], and an Adaptive Low parity Density Code (A-LDPC) [25]. Empirical results show that for realtime traffic, ACE significantly outperforms IEEE802.11 ARQ over the channels with an average BER more than 0.009. It guarantees 100% stability for wide range of traffic rates, while the stability under IEEE802.11 ARQ is only 20%. Meanwhile, for non-realtime traffic, 10% to 30% throughput gains are observed. We also demonstrate that the traditional TCP congestion control algorithm (Jacobson algorithm) gains significant throughput of 10% to 50% under ACE over lossy wireless environment. Finally, we design and implement ACE for real-time video communication using H.264 standard video codec [27]. An overall PSNR gain of 5-10dB over IEEE802.11 ARQ scheme was observed.

The remainder of the paper is organized as follows: In Sections 2 and 3, we address prior studies and preliminaries. The optimal code embedding rate under reliability and stability conditions for ACE protocol is derived in Section 4. Section 5 describes the ACE protocol. In Section 6, we evaluate the performance of ACE over real channel traces collected on 802.11b WLANs. Section 7 concludes the paper.

2. RELATED WORK

Popular link-layer protocols address reliability using different error control schemes. These schemes are classified as follows:

1. **ARQ-based Schemes.** The current IEEE 802.11 protocol is designed to be as reliable as possible [1]; it incorporates frame check sequence (FCS) to detect errors and automatic repeat request (ARQ) to retransmit corrupted packets. For channels with more severe error conditions, the IEEE standard ARQ scheme causes multiple retransmissions which in turn leads to the transmission of a large number of redundant (correct) data. To enhance the IEEE standard ARQ performance, packet combining techniques have been developed where they exploit the multiple transmissions typical of ARQ schemes [9]. The basic idea is to store previous transmissions of the corrupted copies of a packet and attempt error recovery. Techniques developed in this context are *xor combining* [16] and *majority combining* schemes [17]. The analysis in [9, 11, 12] shows these techniques are IEEE MAC compatible, however the improvement of the throughput is not remarkable.

2. **HARQ-based 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 [14, 15]. In the simplest version of HARQ, type-I HARQ [22], 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

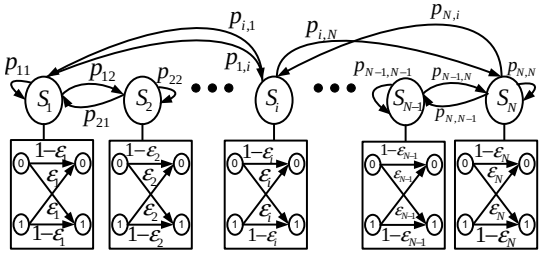


Figure 1: Markovian Channel.

received packet fails. In type-II hybrid ARQ (HARQ-II)[23], 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 ACE protocol since both schemes achieve recovery through the transmission of additional redundancy. However unlike ACE, the HARQ-II is not adaptive with respect to channel condition and does not address throughput stability issues raised by varying traffic demand.

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 [18], 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 [19] 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.

In this paper, we propose a novel error control scheme for wireless link-layer that unlike the related studies targets system stability in conjunction with reliable communication.

3. PRELIMINARIES

In this section, we capture an error process of a wireless channel and determine the likelihood of successful transmission by introducing *channel* and *distortion* models. These models provide essential tools in finding an optimal code embedding rate constraint for ACE protocol. In this paper, the terms "message", "packet" and "codeword" are used interchangeably.

3.1 Channel Model

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 1 shows a Markovian channel model. We assume a homogenous and stationary Markov chain with transitional probability matrix $\mathbf{P}$ and the limiting probabilities $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_N)$. 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 [24].

The capacity of a BSC channel with cross-over probability ϵ_i is $1 - H(\epsilon_i)$ [2]. Using the steady state probabilities π_i the average channel capacity in any interval is determined as follows:

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

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

3.2 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., $\alpha = 0.5$) [3].

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 as-

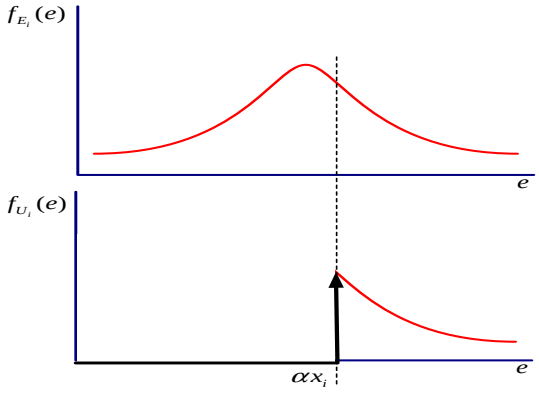


Figure 2: The density of error *after decoding* is truncated on αx_i.

sumes that no recovery is achieved. So the level of distortion in C_i *after decoding*, denoted by U_i, is

$$U_i = g(E_i, x_i) = \begin{cases} 0 & E_i \leq \alpha x_i \\ E_i & \text{otherwise} \end{cases} \quad (2)$$

Equation (2) shows that the distribution of U_i is equivalent to the distribution of E_i *truncated* on αx_i (see Fig. 2). Therefore, U_i has the probability density function

$$f_{U_i}(u) = \begin{cases} F_{E_i}(\alpha x_i) & u = 0 \\ f_{E_i}(u) & u > \alpha x_i \end{cases} \quad (3)$$

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,

$$P(U_i = 0) = F_{E_i}(\alpha x_i) = \sum_{d=0}^{\lfloor \alpha x_i \rfloor} e^{-\lambda_{e_i}} \frac{\lambda_{E_i}^d}{(d)!}. \quad (4)$$

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.

4. 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 $R_i = \frac{k_i}{k_i + x_i}$.

4.1 Code Rate: Reliability

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

DEFINITION 1. *System is reliable when information is transmitted with no or diminishing error over a wireless channel.*

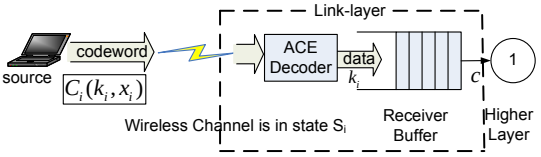


Figure 3: System model for stability analysis in wireless Communication.

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 3.2, 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:

$$\begin{aligned} &\max k_i \quad \text{subject to:} \\ &k_i\epsilon_i + x_i(\epsilon_i - \alpha) \leq 0 \text{ and } k_i + x_i \leq n_i. \end{aligned} \quad (5)$$

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. □

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

DEFINITION 2. *System is stable when higher layers are neither starved for information packets nor is there a glut of packets leading to buffer overflow.*

Figure 3 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:

$$\frac{dB(t)}{dt} = \eta(Z(t)) \tag{6}$$

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:

$$F(b, j) = \lim_{t \to +\infty} P(B(t) \leq b, Z(t) \leq j). \ 1 \leq j \leq M \tag{7}$$

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 [21] shows that $F(b)$ satisfies the differential equation of a form

$$\frac{dF(b)}{dx} D = F(b) Q. \tag{8}$$

By solving the differential equation in (8), we can obtain the limiting distributions in equation (7).

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

$$\omega_i = Pr(\text{Successful Decoding in } \tau_i) = F_{E_i}(\alpha x_i). \tag{9}$$

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) is modeled as a two-state CTMC on state space $S = \{1 = \text{on}, 2 = \text{off}\}$ with the following generator matrix

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

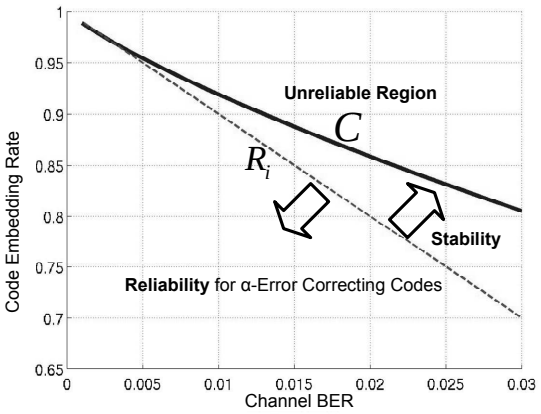


Figure 4: Operational code embedding rate domain with respect to reliability and stability.

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 (8). The final solution is given by

$$F(b,1) = \beta_i (1 - e^{\lambda b}) \tag{10}$$

$$F(b,2) = \omega_i - \frac{\beta_i (k_i - c)}{c} e^{\lambda b}, \tag{11}$$

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

$$k_i \omega_i \leq c. \tag{12}$$

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:

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. 4, 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 [2] which requires the operational code embedding rate to operate below channel capacity of equation (1) ($R_i < C$). An important conclusion of this analysis is that an optimal solution for code embedding rate that ensures reliability and stability conditions is a unique solution:

$$R_i^* = 1 - \frac{\epsilon_i}{\alpha}. \quad \epsilon_i \ll \alpha \tag{13}$$

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.

5. 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.

5.1 ACE Communication Model

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.

5.1.1 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., $C_j, j = 1, \cdots, i-1$).

5.1.2 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. 5). In addition, the receiver estimate

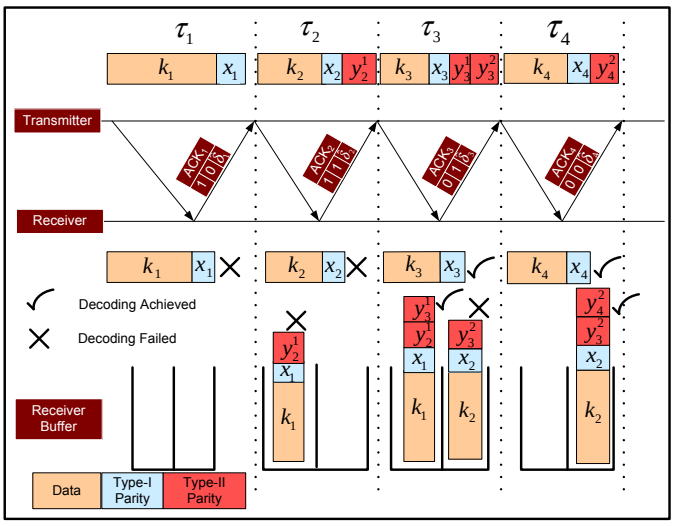


Figure 5: An example of ACE operational communication model consists of four transmission interval.

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. 5. 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 $\mathbf{y}_1 = \mathbf{0}$. 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 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 $ACK_4 = (0, 0, \hat{\delta}_4)$.

5.2 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.

5.2.1 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.

5.2.2 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 $(\mathbf{y}_i)$ 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 4, 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.

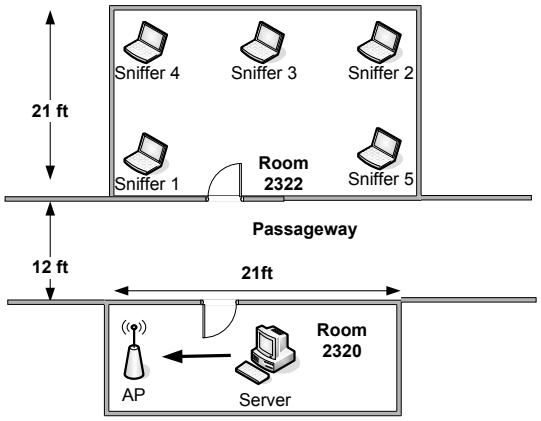


Figure 6: Trace collection setup.

	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

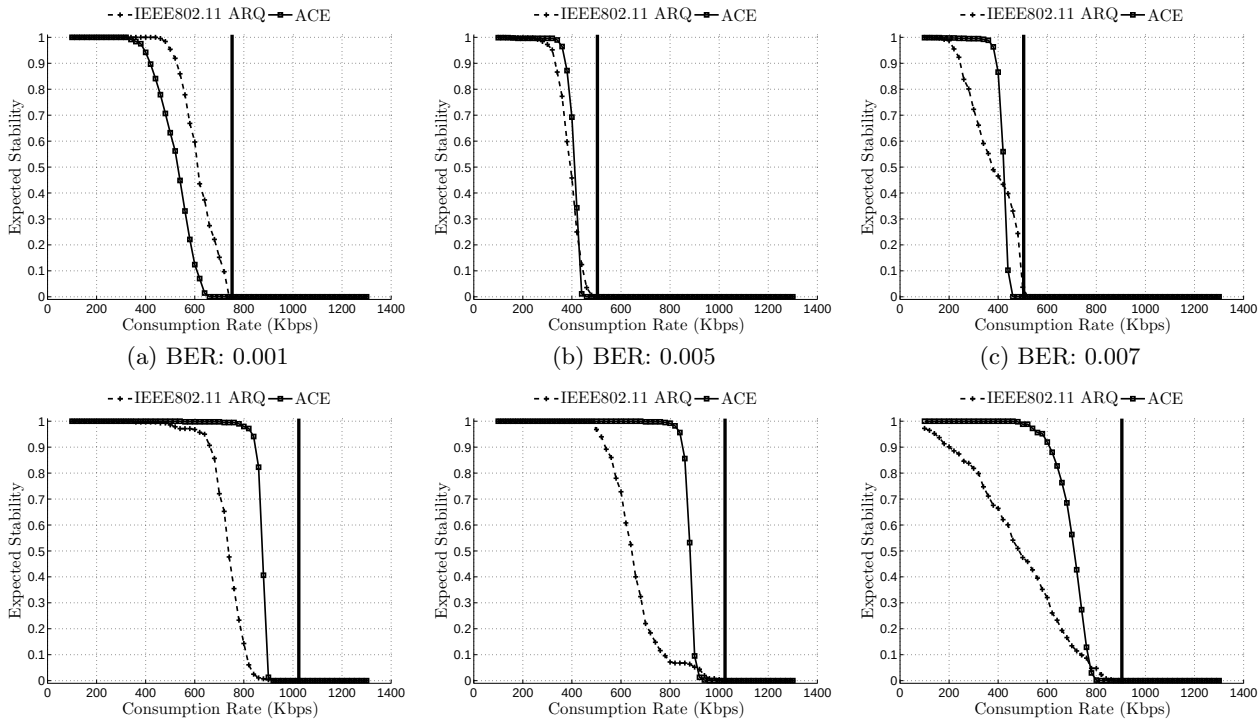
Table 1: The average BER for different channel traces.

6. 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 describe the trace collection methodology. Second, we compare the performance ACE protocol as opposed to the conventional IEEE802.11 ARQ protocol on *realtime* and *non-realtime* traffic. Third, 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.

6.1 Channel Trace Collection

In our analysis, we use channel traces that are collected over the wireless setting depicted in Fig. 6. 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



(a) BER: 0.001 (b) BER: 0.005 (c) BER: 0.007

(d) BER: 0.009 (e) BER: 0.018 (f) BER: 0.020

Figure 7: 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.

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) 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 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 SSR value (i.e., $SSR = S - N$) observed with each packet. The SSR value is used by the receiver for channel BER estimation.

6.2 Realtime Traffic

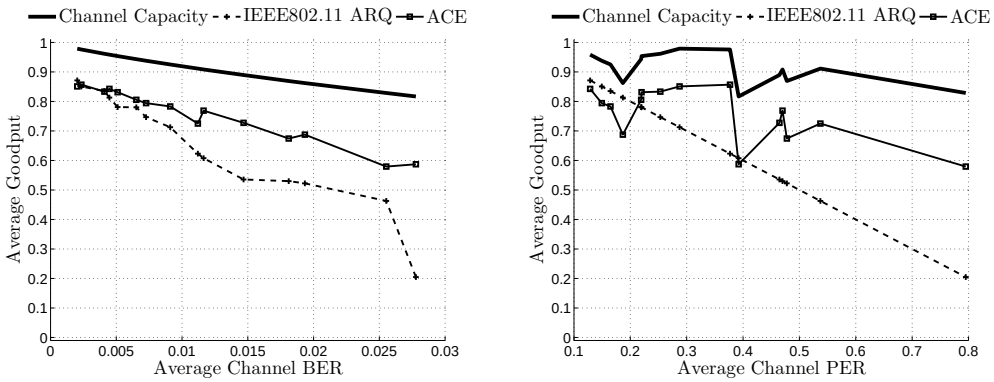
An important aspect of realtime traffic delivery is to prevent instability (by delivering information in timely fashion) while ensuring the required reliability. To determine the impact of deploying ACE protocol on stability for *realtime* traffic, we introduce the following parameters:

- *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 4.2).

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++ [26] 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 [25] into the software for embedded coding operations. The simulation setup is as follow: we let a sender transmits a codeword.



(a) Channel condition (Average BER) (b) Channel condition (Average PER)

Figure 8: 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.

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 [25] 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 5.2.1 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. Fig. 7 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 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 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 exceeds 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.

6.3 Non-Realtime Traffic

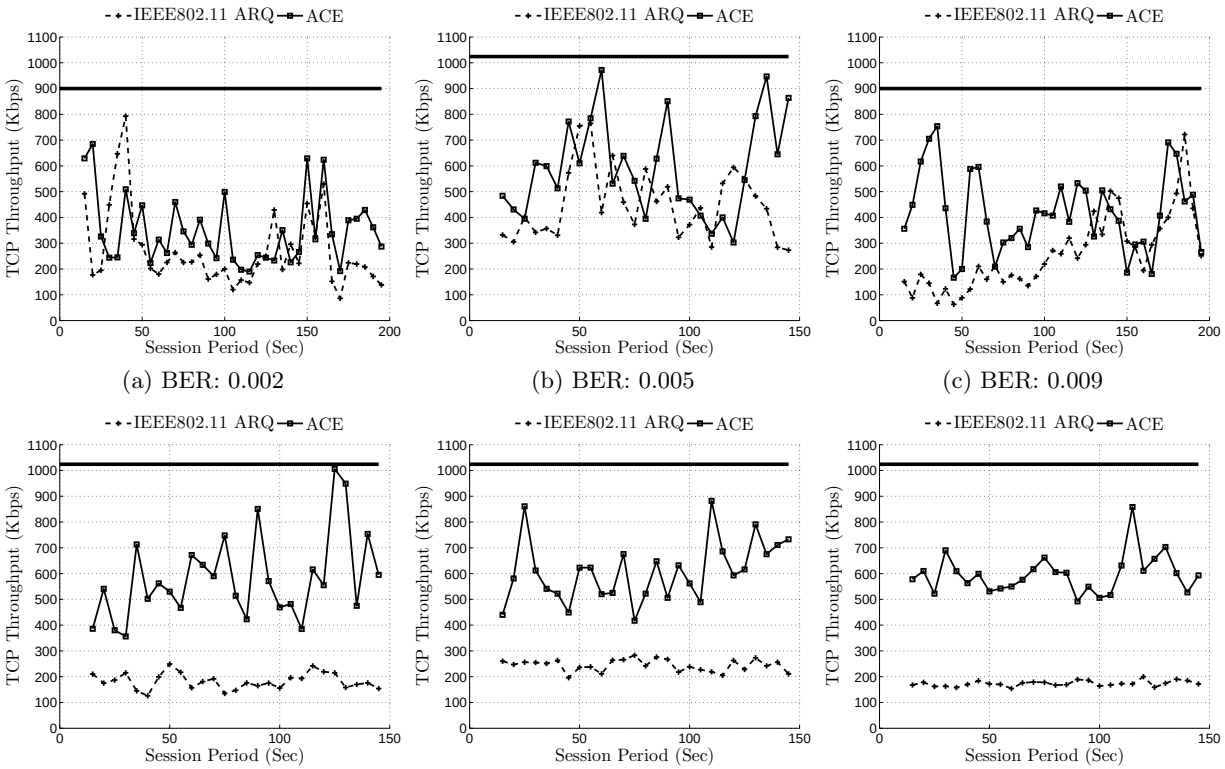
The main objective in *non-realtime* traffic delivery is to maximize the bandwidth utilization of the wireless medium "per channel use" while providing essential reliability. We compare the performances of ACE and IEEE802.11 ARQ under different channel conditions. The performance is in terms of *average 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 8 illustrates the average goodput achieved by ACE and IEEE802.11ARQ over variety of channel conditions. In Fig. 8(a) 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. 8(b). 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 IEEE802.11 ARQ.

6.4 Throughput analysis of TCP

The Jacobson's adaptive window flow control algorithm is common for most TCP variations [6]: 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



(a) BER: 0.002 (b) BER: 0.005 (c) BER: 0.009

(d) BER: 0.011 (e) BER: 0.018 (f) BER: 0.025

Figure 9: 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.

ignore the stability aspect of wireless communication, numerous studies have led to vast variety of TCP congestion control algorithms [7] 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.

Using `INET Framework` TCP model in OMNET++, we analyzed 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. 10. 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. 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.

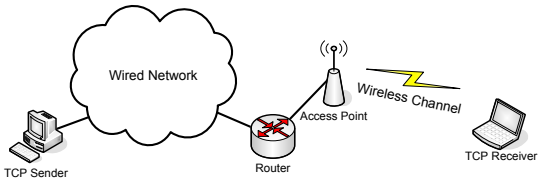
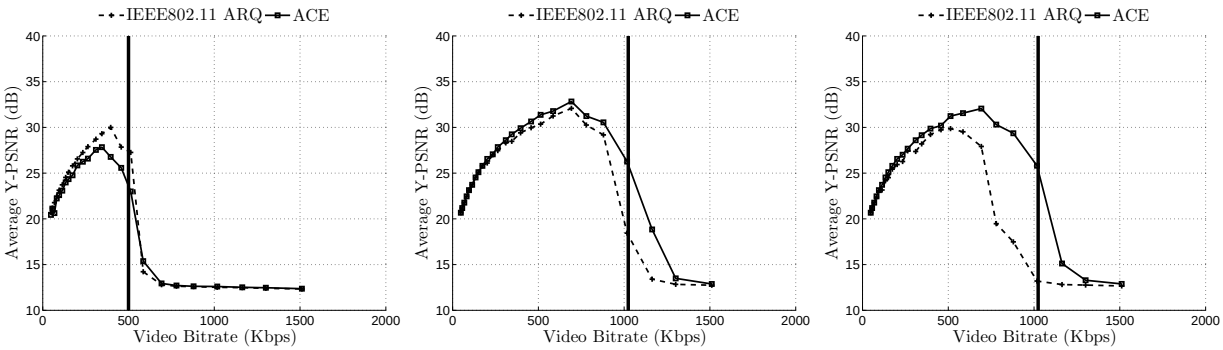


Figure 10: Heterogenous network model.

Fig. 9 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* 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 channels with BER more 0.009 under ACE. This significant difference stems from the fact that ACE targets stability and reliabil-



(a) BER: 0.002 (b) BER: 0.009 (c) BER: 0.020

Figure 11: 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.

ity of wireless communication. This lead to fewer *AP-based* and *distortion-based* losses in wireless section under ACE in comparison with IEEE802.11 ARQ.

6.5 Real-Time 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 video communication. The simulation setup is as follows: a particular video stream is encoded using the H.264/JVT standard software [27]. 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 [26]. ACE encodes each video slice using A-LDPC codes [25] 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 [1] 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 11 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.

7. CONCLUSION

In this paper, 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 framework, 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.

8. REFERENCES

[1] 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.

[2] T.M. Cover, and J. A. Thomas. Elements of Information Theory. *Wiley*, 1991.
[3] S.B. Wicker and V.K. Bhargava. Reed-Solomon Codes and Their Applications. *IEEE Press*, 1994.
[4] 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.
[5] G. Xylomenos, G.C. Polyzos, P. Mahonen, and M. Saaranen. TCP performance issues over wireless links. *IEEE Comm. Magazine*, 39(4):52-58, 2001.
[6] V. Jacobson. Congestion Avoidance and Control. In *Proc. ACM Sigcomm*'88, Aug. 1988, pp- 314-329.
[7] Y. Tian, K. Xu, and N. Ansari. TCP in Wireless Environments: Problems and Solutions. *IEEE Radio Communication*, 43(3): S27-S32, March 2005.
[8] E. Soljanin. Hybrid ARQ in Wireless Networks. *DIMACS Workshop on Network Inform. Theory*, March 2003.
[9] H. Yomo, S. S. Chakraborty, and R. Prasad. IEEE 802.11 WLAN with Packet Combining. *CODEC*, January, 2004, Kolkata, India
[10] E. C. Strinati, S. Simoens, and J. Boutros. Performance evaluation of some Hybrid ARQ schemes in IEEE 802.11a Networks. *IEEE VTC*, 2003
[11] 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.
[12] S. S. Chakraborty, E. Yli-Juuti, and M. Liinaharja. An ARQ Scheme with Packet Combining. *IEEE Comm. Letters*, 1998.
[13] G. Caire and D. Tuninetti. The throughput of Hybird-ARQ protocols for the Gaussion collision channel. *IEEE Trans. Inform. Theory*, July 2001.
[14] 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.
[15] 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.
[16] S. S. Chakraborty, E. Yli-Juuti, and M. Liinaharja. An adaptive ARQ scheme with packet combining. *IEEE Comm. Letters*, vol. 2, no. 7, pp. 200-202, July 1998.
[17] 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.
[18] L. Larzon, M. Degermark, and S. Pink. UDP Lite for Real Time Multimedia Applications. *ICC*, June 1999.
[19] S. S. Karande and H. Radha. Hybrid Erasure-Error Protocols for Wireless Video. *IEEE Trans. Multimedia*, vol. 9, no. 2, Feb 2007.
[20] 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. 1968-1981, Sep. 2000
[21] D. Mitra. Stochastic Fluid Models. *Perfomance 87*, Elsevier Science Publisher, North Holland 1988.
[22] 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. 1701-1719, July 1982.
[23] Y. Wang and S Lin. A modified selective-repeat type-II hybrid ARQ system and its performance analyses. *IEEE Trans. Commun.* 31(5), pp. 124-133, 1983.
[24] S. A. Khayam and H. Radha. Constant-Complexity Models for Wireless Channels. *IEEE Infocom*, 2006.
[25] R. M. Neal. Software for LDPC Codes. http://www.cs.utoronto.ca/radford/ldpc.software.html
[26] OMNeT++ Community, http://www.omnetpp.org.
[27] Draft ITU-T Recommendation and Final Draft Interna-tional Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14496-10 AVC), ISO/IEC JTC 1/SC29/WG11 and ITU-T SG16 Q.6, Doc. JVT-G050. http://iphome.hhi.de/suehring/tml/

9. APPENDIX

PROOF (*Proof of Lemma 1*). The optimization problem in (5) is a convex problem with the Lagrangian in a from of

$$\begin{aligned} L_i(k_i,\lambda_1,\lambda_2) = k_i + \lambda_1\left(k_i\epsilon_i + x_i(\epsilon_i-\alpha)\right) \\ + \lambda_2\left(k_i + x_i - n_i\right).\ \lambda_1,\lambda_2 > 0 \end{aligned} \tag{14}$$

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

$$\lambda_1\left(k_i^*\epsilon_i + x_i^*(\epsilon_i-\alpha)\right) = 0, \tag{15}$$

$$\lambda_2\left(n_i - k_i^* + x_i^*\right) = 0, \tag{16}$$

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

PROOF (*Proof of Lemma 2*). 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 (10)(11) 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

$$\omega_i = \frac{k_i - c}{k_i}. \tag{17}$$

To ensure that the buffer does not overflow, the stability condition in equation (12) should hold. By using equations (9) and (12), we obtain $F_{E_i}(\alpha x_i) = 1 - \frac{c}{k_i} \le \frac{1}{2}$. Therefore, $x_i \le \frac{F_{E_i}^{-1}(0.5)}{\alpha} = \frac{\lambda_{E_i}}{\alpha} = \frac{|C_i|\epsilon_i}{\alpha}$. Correspondingly, by substituting $x_i = |C_i|(1-R_i)$, the lower bound of code embedding rate is $R_i \ge 1 - \frac{\epsilon_i}{\alpha}$. □