Performance Enhancements in Next Generation Wireless Networks Using Network Coding: A Case Study in WiMAX

by

Surat Teerapittayanon
S.B., Electrical Engineering and Computer Science, M.I.T., 2011
S.B., Mathematics, M.I.T., 2011

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2012

Author .....
Department of Electrical Engineering and Computer Science
May 18, 2012

Certified by .....
Muriel Médard
Professor of Electrical Engineering, Thesis Supervisor

Certified by .....
Marie-José Montpetit
Research Scientist, Thesis Co-Supervisor

Certified by .....
Kerim Fouli
Postdoctoral Fellow, Thesis Co-Supervisor

Accepted by .....
Prof. Dennis M. Freeman
Chairman, Masters of Engineering Thesis Committee

Performance Enhancements in Next Generation Wireless Networks Using Network Coding: A Case Study in WiMAX

by

Surat Teerapittayanon

Submitted to the Department of
Electrical Engineering and Computer Science
on May 18, 2012, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, we design and implement a network-coding-enhanced network architecture for next generation wireless networks. The architecture applies intra-session random linear network coding as a packet erasure code below the IP layer. Using WiMAX as a case study, a series of point-to-point single-interface experiments are conducted to compare the performance of the architecture to that of HARQ and ARQ mechanisms. The performance measures are packet loss percentage, throughput and file transfer delay. The experiments use the Global Environment for Network Innovations (GENI) WiMAX platforms. UDP traffic is considered; Iperf and UDP based File Transfer Protocol (UFTP) are used as measurement applications. The proposed architecture substantially decreases packet loss percentage from around 11-32% to nearly 0%. Compared to HARQ and ARQ mechanisms, the architecture can offer up to 5.9 times gain in throughput and 5.5 times reduction in end-to-end file transfer delay.

Thesis Supervisor: Muriel Médard
Title: Professor of Electrical Engineering

Thesis Co-Supervisor: Marie-José Montpetit
Title: Research Scientist

Thesis Co-Supervisor: Kerim Fouli
Title: Postdoctoral Fellow

Acknowledgments

This work would not have been possible without the following people, and I thank them. Muriel Médard for her advice, constant support and generosity, and for sharing her experience, knowledge and wisdom. Marie-José Montpetit, Kerim Fouli, Danail Traskov, Ali ParandehGheibi, Shirley Shi for their insightful comment, discussion, advice and support. Ivan Seskar for his invaluable work on the GENI platforms and his help in experiments at Rutgers. Harry Mussman and Abhimanyu Gosain for their support in experiments at BBN. Jason Cloud, Giovanni Pau for their assistance in experiments at UCLA. My friends, country and family for their relentless support.

Abbreviations

ACK	Acknowledgement
AMC	Adaptive Modulation and Coding
ANSI	American National Standards Institute
ARQ	Automatic Repeated reQuest
BBN	Raytheon BBN Technologies
BS	Base Station
BSN	Block Sequence Number
BTC	Block Turbo Code
CC	Convolutional Code or Chase Combining
CINR	Carrier to Interference plus Noise Ratio
CR	Code Rate
CTC	Convolutional Turbo Code
DL	Downlink
ECC	Error Correcting Code
FDD	Frequency Division Duplexing

FEC	Forward Error Correction
FTP	File Transfer Protocol
GENI	Global Environment for Network Innovations
GPU	Graphical Processing Unit
HARQ	Hybrid Automatic Repeated reQuest
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IR	Incremental Redundancy
LDPC	Low-Density Parity-Check code
LNC	Linear Network Coding
LTE	Long-Term Evolution
MAC	Medium Access Control
MCS	Modulation and Coding Scheme
MIMO	Multiple-Input and Multiple-Output
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit
NACK	Negative Acknowledgement
NC	Network Coding
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access

OS	Operating System
PDU	Protocol Data Unit
PER	Packet Error Rate
PHY	Physical
QAM	Quadrature Amplitude Modulation
QoE	Quality of Experience
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RFC	Request For Comments
RLNC	Random Linear Network Coding
RS	Relay Station
RSSI	Received Signal Strength Indication
RTT	Round-Trip Time
SC	Single Carrier
SDU	Service Data Unit
SIMD	Single Instruction Multiple Data
SNR	Signal-to-Noise Ratio
SS	Subscriber Station
SSE	Streaming SIMD Extension
TDD	Time Division Duplexing

TLER	Throughput to Loss plus Extra Ratio
TTL	Time To Live
UCLA	University of California, Los Angeles
UDP	User Datagram Protocol
UFTP	UDP based FTP
UL	Uplink
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
ZCC	Zero-terminating Convolutional Code

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter briefly states the background and motivation for this study, followed by a description of its scope. It then provides an overview of related work on network coding in conjunction with retransmission schemes. Finally, a short outline of the rest of the thesis is given.

1.1 Background and Motivation

The growing market of mobile devices is placing increasing demands on wireless networks. Indeed, at the end of 2009, the number of mobile phone subscribers exceeded 4.6 billion worldwide [65], and the global mobile data traffic has been predicted to double every year through 2014 [30]. The significant growth in mobile data traffic requires higher communication capacity. As a consequence, a crucial challenge for next generation wireless networks is to cope with the rapid increase in multimedia traffic with minimal impact on equipment complexity [30].

In past years, Network Coding (NC) has been recognized as one of the solutions to cope with network congestion [25, 26]. It uses network resources better [17] and improves dissemination of content in the network [43]. NC also allows tailoring the encoding to the dynamics of the network topology, which is an essential feature for mobile wireless networks [48]. Many

studies have shown that NC for Wireless Local Area Networks (WLANs) significantly enhances network throughput, robustness, and security; in particular, the network throughput gain is considerable [54, 71]. Random Linear Network Coding (RLNC) [13, 28], where the NC coefficients are selected randomly over a chosen Galois field, has proven particularly effective in optimizing network resource consumption in WLANs [17, 31]. In fact, using NC, COPE [35] shows 3-4x throughput gain in WLANs.

Despite the demonstrated effectiveness of NC in WLANs, NC for Wireless Metropolitan Area Networks (WMANs) has just recently gained attention, as the telecommunication industry moves toward next generation wireless networks such as 4G WiMAX and 4G LTE-Advanced. 4G requires stationary speeds of 1 Gbps and mobile speeds of 100 Mbps, while 3G only requires stationary speeds of 2 Mbps and mobile speeds of 384 Kbps [5]. That is, 4G requires 500 and 260 times faster speeds than 3G in stationary and mobile cases, respectively. Thus, it is essential to investigate potential applications of NC in these next generation wireless networks to ensure that the needs of subscribers are served by the deployed networks.

1.2 Scope

Network Coding can be applied across the OSI model [69] from the physical [31] to the network and application layers [58]. In this thesis, we design and implement an NC-enhanced network architecture below the IP layer to minimize packet loss and maximize throughput while reducing delay, thus improving both Quality of Service (QoS) for the operator and Quality of Experience (QoE) for the end-user. Intra-session random linear network coding is used as a packet erasure code. Using WiMAX as a case study, we compare the packet loss percentage, throughput and file transfer delay of the proposed architecture with those of the Hybrid Automatic Repeated reQuest (HARQ) and Automatic Repeated reQuest (ARQ) mechanisms in WiMAX (See Chapter 3). The Global Environment for Network Innovations (GENI) WiMAX platforms will be used to conduct experiments. Since the GENI WiMAX Base Stations (BSs) only support Chase Combining HARQ, only Chase Combining HARQ is considered, and not Incremental Redundancy HARQ. User Datagram Protocol (UDP) traffic

is considered; Iperf and UDP based File Transfer Protocol (UFTP) are used as measurement applications.

1.3 Related Work

Network Coding (NC) was originally proposed to mix packets at nodes to maximize the capacity of a wired network [7]. COPE [35] is considered the first system that has successfully implemented NC to wireless networks to improve the throughput using overheard packets. However, even before the advent of NC [7] and COPE [35], Metzner [49] presented a similar coding scheme to be used in conjunction with retransmission schemes in single-hop broadcast settings. In [49], the authors propose a scheme where the retransmitted frame is simply the NACKed frames of various receivers XORed together. We call such a scheme XOR NC. Since then, NC in conjunction with retransmission schemes such as ARQ and HARQ has been widely studied [6, 18, 19, 29, 32–34, 36–40, 42, 47, 49–53, 57, 59, 61–63, 67, 68, 70, 72–74]. We briefly describe these articles.

Jolfael et al. [34] apply XOR NC to ARQ for use in a point-to-multipoint communication over broadcast links while Yong et al. [72] consider XOR NC and ARQ in multicast settings. Larsson et al. [36, 38] study XOR NC and multi-user ARQ in multiple unicast settings and suggest that linear coding in some other field may also be used instead of XOR NC. Larsson et al. [37] also consider adaptive linear NC and ARQ in multicast settings, where coefficients for the linear combination of data packets are adaptively selected from a sufficiently large finite field.

NC-HARQ [62] applies NC to HARQ in single-hop wireless networks. NC-HARQ uses XOR NC in conjunction with the Forward Error Correction (FEC) of HARQ, where lost packets from different receivers are XORed together and applied FEC. NC-HARQ, in effect, combines network and channel coding. In NC-HARQ, lost packets from different receivers are XORed together. Thobaben et al. [61] and Larsson et al. [39] consider NC-HARQ and multi-user HARQ in multiple unicast settings. Peng et al. [51] consider NC-HARQ in both broadcast and unicast scenarios. Tran et al. [63] extend NC-HARQ to adjust the amount of

FEC in real time to adapt to the channel conditions in single-hop wireless networks. This technique increases the throughput efficiency up to 3.5 times over ARQ and 1.5 times over HARQ. Zhang et al. [74] extend NC-HARQ to add XOR operations to combine dynamically lost packets from the same receiver in addition to XOR operations that combines lost packets from different receivers. Lu et al. [42] look at NC-HARQ for wireless video broadcast. Abuzeid et al. [6] compare NC-HARQ and Incremental Redundancy HARQ in cooperative wireless communication systems.

MRNC [32, 33] considers RLNC at the MAC layer, where a data segment is divided and coded together. N-in-1 NC [40] extends MRNC. In N-in-1 NC, a data segment is first divided, coded and transmitted. For retransmissions, instead of coding a single data segment, N data segments are coded together. N-in-1 NC achieves a throughput gain of up to 106% against the conventional Chase Combining HARQ.

Lun et al. [45, 46] shows a capacity-achieving coding scheme based on RLNC, where coded packets are formed from random linear combinations of previously received packets and sent out whenever there is a transmission opportunity. Dana et al. [14] derive the capacity for a class of wireless erasure networks with broadcast and no interference at reception and show that linear coding at nodes in wireless erasure networks suffices to achieve the capacity region. Ghaderi et al. [19] analytically quantify the reliability gain of network coding for reliable multicasting in wireless networks and show that network coding achieves asymptotic performance results similar to that of rateless erasure coding. Sundararajan et al. [59] theoretically extend ARQ with RLNC [13, 28]. Nguyen et al. [50] provide theoretical results comparing the bandwidth efficiency of RLNC to that of ARQ. Pu et al. [52] develop an information-theoretic performance bound to predict the coding gains of Chase Combining HARQ in broadcast settings.

Recently, Manssour et al. [47] proposed a novel retransmission scheme for wireless unicast communication using a combination of channel coding and network coding and showed 68.75% throughput gains compared to Chase Combining HARQ. Qureshi et al. [53] presents BENEFIT, an efficient NC-based transmission algorithm, in a single-hop wireless multicast

network. BENEFIT starts retransmitting as soon as certain conditions are met and in effect reduces the time to decode the packet. While all previous contributions consider digital network coding, SYNC [73] considers symbol level network coding at the physical layer, where corrupted packets may be useful.

Fan et al. [18], Sun et al. [57] and Vien et al. [68] consider a scenario where two nodes communicate with the base station with the assistance of a relay. Fan et al. [18] introduce a NC Based Cooperative multicast scheme (NCBC) while Sun et al. [57] discuss cooperative HARQ based on NC (C-HARQ-NC). Vien et al. [68] investigate ARQ based on NC for two-way wireless relay networks. Recently, Vien et al. also discussed NC based Block ARQ (BACK) for wireless relay networks [67]. Hong et al. [29] propose NC-HARQ for mobile relay systems.

Jin et al. [32, 33] and Yazdi et al. [70] consider NC in conjunction with ARQ and/or HARQ in WiMAX. Jin et al. [33] introduce MAC layer Random Network Coding (MRNC), which offers a 10% gain in throughput over HARQ in single-hop transmissions. Adaptive MRNC [32] extends MRNC with adaptive schemes and outperforms regular MRNC by 28.4% and HARQ by 57.7% in terms of throughput. Adaptive MRNC uses the channel state information feedback to adjust dynamically packet size according to the current channel conditions. Recently, Yazdi et al. [70] extended MRNC to restrict the number of retransmissions to an upper bound which is important for delay sensitive applications.

Despite a number of studies on the HARQ and ARQ mechanisms, most of those studies are limited to simulations and analysis; there is lack of experimental data. All work cited above give analytical and/or simulation results, but none provides experimental results. In this thesis, we provide experimental results in a point-to-point setting. Our design uses RLNC in $\mathbf{GF}(2^8)$ as a packet erasure code. Ho et al. [27] show that using a Galois field of a limited size is sufficient to implement network coding in a practical network setting. We compare the performance of our design to that of ARQ and Chase Combining HARQ mechanisms in WiMAX.

1.4 Outline

The remainder of the thesis is organized as follows.

Chapter 2 gives a brief introduction to network coding. We present the definition, an example and a classification of network coding. In particular, Random Linear Network Coding (RLNC) is described in detail. This chapter also discusses using random linear network coding as a packet erasure code.

Chapter 3 gives a brief introduction to Worldwide Interoperability for Microwave Access (WiMAX). It discusses two retransmission mechanisms in WiMAX: Automatic Repeated reQuest (ARQ) and Hybrid Automatic Repeated reQuest (HARQ). Adaptive Modulation and Coding (AMC) in WiMAX is also discussed in this chapter.

Chapter 4 presents the network-coding-enhanced network architecture and its applications in point-to-point single- and multiple-interface networks.

Chapter 5 describes the design of the network coding application. The encoder and decoder processes are depicted and the design parameters are defined. The encoder, decoder and feedback mechanisms are also discussed in this chapter.

Chapter 6 discusses some of our design and implementation decisions. In particular, we discuss the implementation of our network coding scheme, our decoding algorithm and random seeds used to generate the code. The NC header structure and the data padding are also discussed.

Chapter 7 discusses the preliminary experiments performed using WiMAX technology. The objective of the preliminary experiments is to measure the communication channel and the network performance of different configurations of WiMAX Base Stations (BSs), focusing on switching on and off HARQ and ARQ. The measurements are collected at two different sites: Raytheon BBN Technologies (BBN) in Cambridge, Massachusetts and the University of California, Los Angeles (UCLA), California.

Chapter 8 discusses the network coding experiments and results. The objective of the network coding experiments is to validate the potential for Network Coding (NC) to replace the HARQ and ARQ mechanisms. Four fixed downlink Modulation and Coding Schemes

(MCSs) and base station transmission power levels are considered: 64 QAM CTC 1/2 at 13 dBm, 64 QAM CTC 2/3 at 17 dBm, 64 QAM CTC 3/4 at 18 dBm and 64 QAM CTC 5/6 at 20 dBm. The measurements are remotely conducted at the Rutgers University, New Jersey.

In Chapter 9, the contributions of this thesis are summarized, and a number of important future work topics are listed.

Chapter 2

Network Coding

In communication networks, files commonly are divided into packets in order to be transmitted from one node to another. Traditionally, packets are stored and then forwarded. Network coding has been proposed to replace this traditional "store and forward" model and to improve the throughput and robustness of networks. This chapter gives a brief introduction to network coding, presenting the definition, example and classification of network coding. In particular, Random Linear Network Coding (RLNC) is discussed in detail. Then, this chapter discusses random linear network coding as a packet erasure code.

Ahlswede et al. [7] first introduced network coding in 2000. Since then, it has been studied widely by the research community. Network Coding (NC) is a particular data processing technique in networks where network nodes transmit any combination of a set of available packets. Figure 2-1 shows the potential benefit of network coding in a simple example. The example shows that with network coding, one transmission can be avoided by combining two packets at node 3. In this example, network coding potentially delivers information to more than one node by sending the combination of b_1 and b_2 to node t_1 and node t_2. Node t_1 gains information about b_2, and node t_2 gains information about b_1. Compared to the store and forward model (Figure 2-1a), network coding avoids one transmission per unit time (Figure 2-1b), thus increasing throughput.

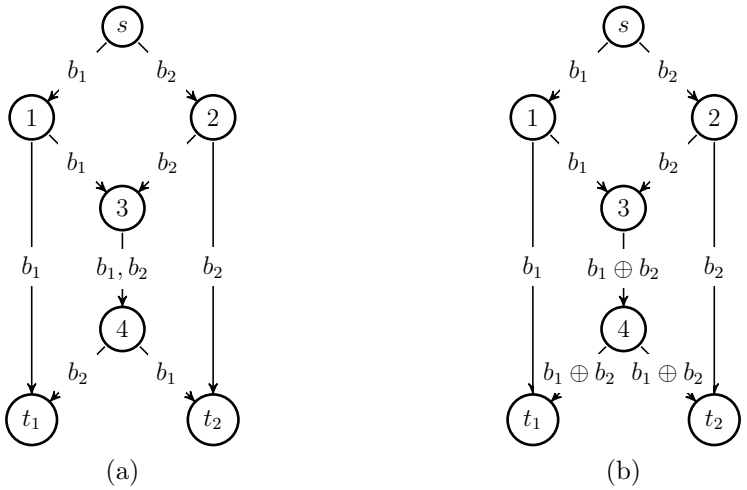


Figure 2-1: The butterfly network. (a) store-and-forward is applied. (b) network coding is applied. The source, node s, transmits b_1 and b_2 to t_1 and t_2. With network coding, one transmission can be avoided by combining the two packets at node 3.

2.1 Intra-Session and Inter-Session Network Coding

Network coding can be classified into two types: intra-session network coding and inter-session coding. Intra-session network coding is network coding where combinations are restricted to packets belonging to the same session or connection. In Inter-session network coding, however, combinations are allowed among packets belonging to possibly different sessions or connections. This thesis focuses on intra-session network coding.

2.2 Random Linear Network Coding

The most common class of network coding used in practical applications is Random Linear Network Coding (RLNC) [13, 28]. RLNC is an extension of Linear Network Coding (LNC). LNC is NC where all combinations are linear combinations: multiplying each term by a constant and adding the results. RLNC is LNC where the coefficients of the linear combinations are chosen randomly. This section describes RLNC encoding and decoding in detail. Then, it discusses systematic network coding used in conjunction with RLNC in practice.

2.2.1 Encoder

The encoder encodes information as follows.

$$c_j = \sum_{i}^{n} a_{ji} s_i, \tag{2.1}$$

where s_i is a vector of source bytes (a packet), c_j is a vector of coded bytes (a coded packet) and a_{ji}'s are random coefficients. Alternatively, Equation (2.1) can be written in matrix form as

$$\mathbf{C} = \mathbf{AS}, \tag{2.2}$$

where $\mathbf{C}$ is the matrix of coded vectors c_j's, $\mathbf{A}$ is the matrix of coefficients a_{ji}'s, and $\mathbf{S}$ is the matrix of source vectors s_i's. In this thesis, network coding is applied on packets.

2.2.2 Decoder

The decoder decodes by gathering n linearly independent coded packets, c_j, and solving Equation (2.2):

$$\mathbf{S} = \mathbf{A}^{-1}\mathbf{C}. \tag{2.3}$$

If $\mathbf{A}$ is invertible, the uncoded information can be recovered, and the uncoded information is found in $\mathbf{S}$. All calculation is done over a finite field. For more information on finite fields, see Appendix A.

2.2.3 Systematic Network Coding

In practice, RLNC is used with systematic network coding to help speed up the decoding time [44]. Systematic network coding is a type of network coding consisting of two phases. In the first phase, all n packets are transmitted uncoded. Uncoded packets have a unit coefficient vector of the form: $(0, 0, ..., 1, ..., 0)$. Then, redundancy coded packets are transmitted in the second phase.

Assuming we use Gauss-Jordan elimination [56] to decode, systematic network coding helps speed up the decoding time in three ways. First, uncoded packets are used to perform a forward elimination only in the coded packets, not in the other uncoded packets. Second, no operations have to be performed for the columns that are known to be zero in the uncoded packet. Third, a back substitution step is not needed for the uncoded packets. Additionally, uncoded packets can be used to recover partial information when there are not enough linearly independent packets.

2.3 Network Coding as Packet Erasure Codes

Network coding helps improve not only throughput, as shown in Figure 2-1, but also robustness against packet loss. In this section, we explore the use of network coding as a packet erasure code to enhance robustness in packet erasure networks. Packet erasure networks are networks with a communication channel where a packet is either received or lost. Packet erasure codes transform a message of k packets into a longer message of n packets such that the uncoded message can be recovered from a subset of the n packets. The fraction $\frac{k}{n}$ is called the code rate.

As a packet erasure code, network coding introduces redundancy packets so that the uncoded packets can be recovered when a sufficient number of coded packets (degrees of freedom) is received. Hence, a sender can potentially generate an unlimited number of packets; a receiver can continue receiving packets until it is able to decode. Additional packets that are linearly independent from already received packets always contain new information. Each additional packet reduces the code rate but increases the likelihood of decoding in a packet erasure network.

In addition to providing redundancy, network coding simplifies the acknowledgement process. In packet erasure networks, an acknowledgement (ACK) mechanism is often used to counter packet loss. Typically, an ACK packet is transmitted from the receiver to the sender to acknowledge the arrival of any particular packet or group of packets (See Chapter 3). Network coding removes the requirement for ACKs prior to decoding, thus potentially

simplifying the ACK procedure (See Chapter 5). By introducing redundancy packets and simplifying acknowledgements, network coding, therefore, can be used to protect against packet loss and improve network robustness.

Despite its advantages, network coding has limitations. One such limitation is the "all-or-nothing" property: if n packets are combined using network coding, at least n packets are needed in order to be able to recover the n uncoded packets. In this chapter, we define network coding, linear network coding and random linear network coding as well as intra-session and inter-session network coding. We also discuss the use of network coding as a packet erasure code. In this thesis, we use intra-session random linear network coding as a packet erasure code to help enhance robustness in packet erasure networks. The next chapter gives an overview of WiMAX.

Chapter 3

Worldwide Interoperability for Microwave Access (WiMAX)

Worldwide Interoperability for Microwave Access (WiMAX) is intended for Wireless Metropolitan Area Networks (WMANs). WiMAX provides a robust, reliable, and cost-effective means to deliver broadband services in metropolitan and rural areas. Figure 3-1 represents the basic WiMAX network architecture. Subscriber Stations (SSs) such as mobile phones, laptops and cars communicate with the internet and with other SSs through Base Stations (BSs). Access Service Network Gateway (ASNGW) acts as a traffic aggregation point within an Access Service Network (ASN).

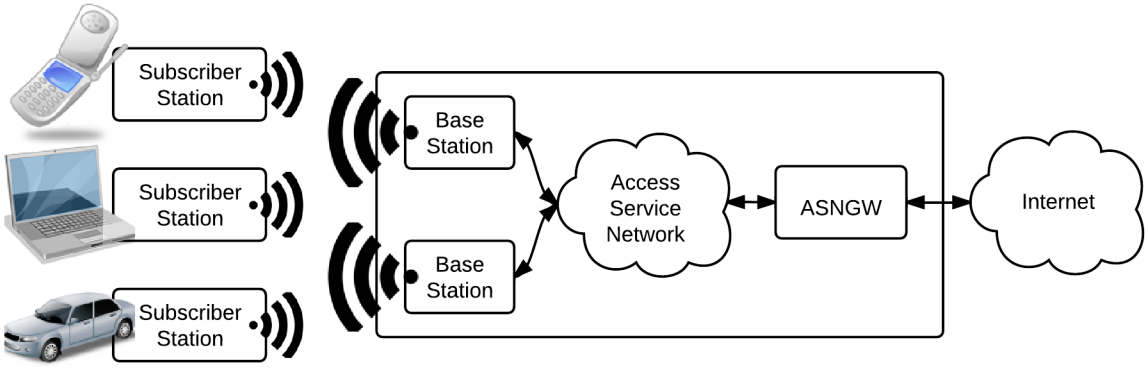


Figure 3-1: WiMAX Network Architecture. Access Service Network Gateway (ASNGW) acts as a traffic aggregation point within an Access Service Network (ASN).

WiMAX [8] is a similar technology to the Long-Term Evolution (LTE) [20]; Chang et al. [12] compare WiMAX and LTE. In this chapter, the WiMAX IEEE 802.16 standard is discussed. In particular, we discuss the basics of Automatic Repeated reQuest (ARQ), Hybrid Automatic Repeated reQuest (HARQ) and Adaptive Modulation and Coding (AMC) in the WiMAX IEEE 802.16 standard. Part of the information in this chapter is drawn from [22] and [8].

The WiMAX IEEE 802.16 standard contains specifications for the PHYsical layer (PHY) and the Medium Access Control layer (MAC). Figure 3-2 shows a typical WiMAX network stack. The PHY supports Single Carrier (SC), Orthogonal Frequency Division Multiplexing (OFDM), and Orthogonal Frequency Division Multiple Access (OFDMA). The MAC supports both Time Division Duplexing (TDD) and Frequency Division Duplexing (FDD).

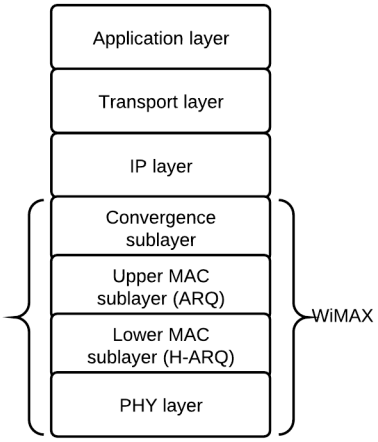


Figure 3-2: WiMAX Network Stack.

The MAC includes a convergence sublayer that can interface with a variety of higher-layer protocols such as Ethernet and IP. Besides providing a mapping to and from the higher layers, the convergence sublayer supports header suppression to reduce higher layer overhead in each packet. The MAC also performs fragmentation and packing. Fragmentation is a process in which a MAC Service Data Unit (MSDU) is divided into one or more MSDU fragments. Packing is a process where multiple MSDUs are assembled into a single MAC Protocol Data Unit (MPDU).

To alleviate the impact of wireless errors on the network performance, WiMAX adopts two retransmission mechanisms: Automatic Repeated reQuest (ARQ) at the upper MAC and Hybrid Automatic Repeated reQuest (HARQ) at the lower MAC and PHY. Both ARQ and HARQ mechanisms retransmit data when data is not correctly delivered to the receiver. ARQ and HARQ are discussed in Section 3.1 and 3.2, respectively.

To improve overall system capacity, WiMAX supports a number of advanced link adaptation techniques such as Adaptive Modulation and Coding (AMC). In AMC, modulation and coding are dynamically adapted to the current channel condition; AMC is discussed in Section 3.3.

3.1 Automatic Repeated reQuest (ARQ)

Automatic Repeated reQuest (ARQ) is an error control technique for data transmission in which the receiver asks the transmitter to resend the blocks of data in which errors are detected. The receiver verifies each block using the cyclic redundancy check code (CRC). If errors are detected, the receiver sends a negative acknowledgement (NACK); otherwise, it sends a positive acknowledgement (ACK). The transmitter retransmits the block only if it receives a NACK or its retransmission timer expires. If the transmitter receives an ACK, then the block are successfully transmitted. A retransmission timer is specified using the ARQ_RETRY_TIMEOUT parameter, which is the minimum time interval during which a transmitter waits before the retransmission of an unacknowledged ARQ block.

ARQ in the MAC can be enabled or disabled. For ARQ-enabled MAC, MSDUs are first partitioned into fixed-length ARQ blocks, where the last ARQ block is padded. Then, a Block Sequence Number (BSN) is assigned to each ARQ block. The length of ARQ blocks is specified using the ARQ_BLOCK_SIZE parameter. After the ARQ block partitioning, fragmentation and packing are applied, ARQ blocks are assembled into MPDUs. Figure 3-3 shows the operation of an ARQ-enabled upper MAC. For the ARQ-disabled MAC, MSDUs are fragmented and/or packed into MPDUs. Figure 3-4 shows the operation of ARQ-disabled upper MAC.

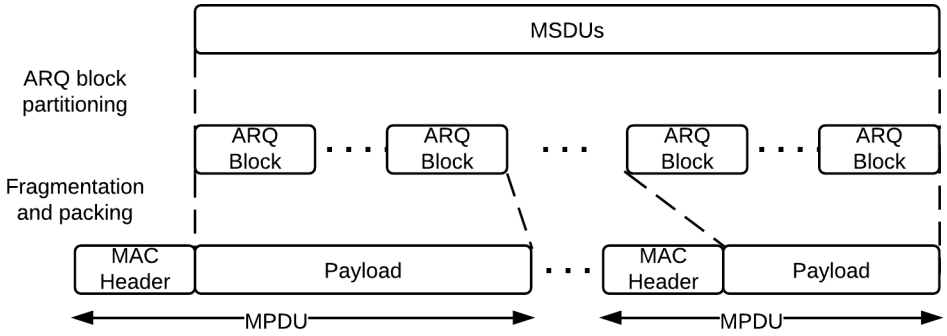


Figure 3-3: WiMAX MAC layer with ARQ enabled.

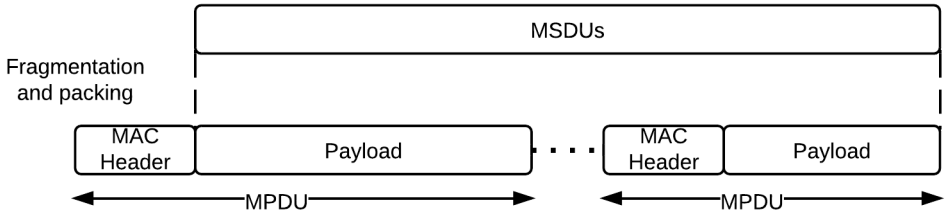


Figure 3-4: WiMAX MAC layer with ARQ disabled.

In the WiMAX ARQ scheme, a transmitter and a receiver each maintain a sliding window of ARQ blocks that shifts each time the transmitter receives an ACK. ARQ window size is specified using the ARQ_WINDOW_SIZE parameter.

The WiMAX IEEE 802.16 standard defines various ARQ parameters. However, there is no specification about how to use them. In the next section, we introduce the main ARQ parameters. Other parameters are summarized in Table 3.1.

3.1.1 ARQ Parameters

To distinguish between the ARQ window at the transmitter and the ARQ window at the receiver, the ARQ window at the transmitter is called the ARQ-Tx window, and the ARQ window at the receiver is called the ARQ-Rx window. There are two variables related to the ARQ-Tx window: ARQ_TX_WINDOW_START and ARQ_TX_NEXT_BSN. ARQ_TX_-WINDOW_START indicates the starting point of the ARQ-Tx window, whereas ARQ_TX_-

NEXT_BSN indicates the BSN of the next ARQ block to be sent in the ARQ-Tx window. The transmitter updates the ARQ_TX_WINDOW_START and ARQ_TX_NEXT_BSN variables, when it receives an ARQ feedback message which includes ACK to indicate the successful reception of an ARQ block and NACK to request the retransmission of an ARQ block owing to unsuccessful reception. The time delay before a receiver sends an ARQ feedback message is specified in ARQ_TX_ACK_DELAY.

There are also two variables related to the ARQ window at the receiver (ARQ-Rx window): ARQ_RX_WINDOW_START and ARQ_RX_HIGHEST_BSN. ARQ_RX_WINDOW_START represents the starting point of the ARQ-Rx window, and ARQ_RX_HIGHEST_BSN represents the BSN of the next ARQ blocks to be received in the ARQ-Rx window. The receiver first updates ARQ_RX_WINDOW_START and ARQ_RX_HIGHEST_BSN variables, when it receives ARQ blocks. Then, it sends an ARQ feedback message which includes ACK or NACK.

3.1.2 ARQ Parameter Sensitivity

We now discuss the effects of setting short or long ARQ_RETRY_TIMEOUT and ARQ_TX_ACK_DELAY values. For ARQ_RETRY_TIMEOUT, on one hand, a short ARQ_RETRY_TIMEOUT value may cause unnecessary retransmissions, which wastes resources at the transmitter. On the other hand, a long ARQ_RETRY_TIMEOUT value may increase the delay of data that has not arrived successfully at the receiver.

For ARQ_TX_ACK_DELAY, on one hand, a short ARQ_TX_ACK_DELAY value results in a large number of transmissions of ARQ feedback messages. ARQ feedback messages are sent using the same resources that are used for data transmissions. Thus, a short ARQ_TX_ACK_DELAY value increases the control overhead, which decreases the system throughput. On the other hand, a long ARQ_TX_ACK_DELAY value may increase the delay of data, especially if the data is not delivered to the receiver successfully. In addition, it may decrease the system throughput because of the ARQ stall problem that occurs when the transmitter cannot send data because the ARQ window freezes.

Table 3.1: ARQ Parameters and Description

Parameters	**Description**
ARQ_RETRY_TIMEOUT	The minimum time interval a transmitter will wait before retransmission of an unacknowledged block for retransmission. The interval begins when the ARQ block was last transmitted. On connections that use both HARQ and ARQ, the ARQ_RETRY_TIMEOUT value should be set accordingly to allow HARQ retransmission operation of the ARQ block to be completed before ARQ retransmission occurs. An ARQ block is unacknowledged if it has been transmitted but no acknowledgment has been received.
ARQ_BLOCK_SIZE	ARQ block size. Before transmission, MSDUs are partitioned into a sequence of ARQ blocks of this size.
ARQ_WINDOW_SIZE	The window size or the number of queued ARQ acknowledgement blocks at any given time for a connection.
ARQ_TX_ACK_DELAY	The time delay before a receiver sends an ARQ feedback message.
ARQ_ACK_PROC_TIME	The time allowed for ACK to be processed.
ARQ_BLOCK_LIFETIME	The maximum time interval an ARQ block will be managed by the transmitter, once initial transmission of the block has occurred. If transmission (or subsequent retransmission) of the block is not acknowledged by the receiver before the time limit is reached, the block is discarded.
ARQ_DLV_ORDER	The in-order delivery capability. It indicates whether to enable in-order delivery. If enabled, the data units will be buffered and reordered before delivery.
ARQ_RX_PURGE_TIMEOUT	The time interval the receiver will wait after successful reception of a block that does not result in advancement of ARQ_RX_WINDOW_START value.
ARQ_SYNC_LOSS_TIMEOUT	The maximum time interval ARQ_TX_WINDOW_START or ARQ_RX_WINDOW_START parameters can stay at the same value before declaring a loss of synchronization between transmitter and receiver.

3.2 Hybrid Automatic Repeated reQuest (HARQ)

Hybrid Automatic Repeated reQuest (HARQ) is an error correction and control technique. It combines Forward Error Correction (FEC) [10] and ARQ to ensure a more reliable transmission. Unlike in ARQ, where all transmissions are processed independently, in HARQ, subsequent retransmissions are jointly processed with all the previous transmissions. Instead of discarding each erroneously received block, subsequent retransmitted blocks are combined with the previous erroneously received retransmitted blocks to improve reliability. Two extensively investigated implementations of HARQ are Chase Combining (CC) and Incremental Redundancy (IR). Reference [16] compares performance of these different implementations. In CC, a retransmitted block is identical to the initial transmitted block. In IR, each retransmitted block is a different version of the coded block. Typically, a version of the coded block is created via a process called puncturing, where some of the output error-correcting coded bits are removed [66]. Different versions have different puncturing patterns. Consequently, at every retransmission the receiver gains knowledge of extra information. In WiMAX, at most four different encoded retransmitted blocks can be generated and retransmitted.

Figure 3-5 and 3-6 show the lower MAC layer, with HARQ enabled and disabled, respectively. For both cases, the MAC Protocol Data Unit (MPDU), or a concatenation of MPDUs, is padded so that the size of the resulting block of MPDUs is in the set {4, 10, 16, 22, 34, 46, 58, 118, 238, 358, 598, 1198, 1798, 2398, 2998} bytes. Subsequently, a CRC field is added, so that the resulting data unit length is in the set {6, 12, 18, 24, 36, 48, 60, 120, 240, 360, 600, 1200, 1800, 2400, 3000} bytes. The length of the data unit is mainly determined by the modulation and coding scheme selected. For the HARQ-disabled MAC, the data unit then undergoes modulation. For the HARQ-enabled MAC, before modulation, the data unit undergoes randomization, fragmentation and FEC. Four subpackets are generated and will be transmitted if needed. Various HARQ parameters are summarized in Table 3.2.

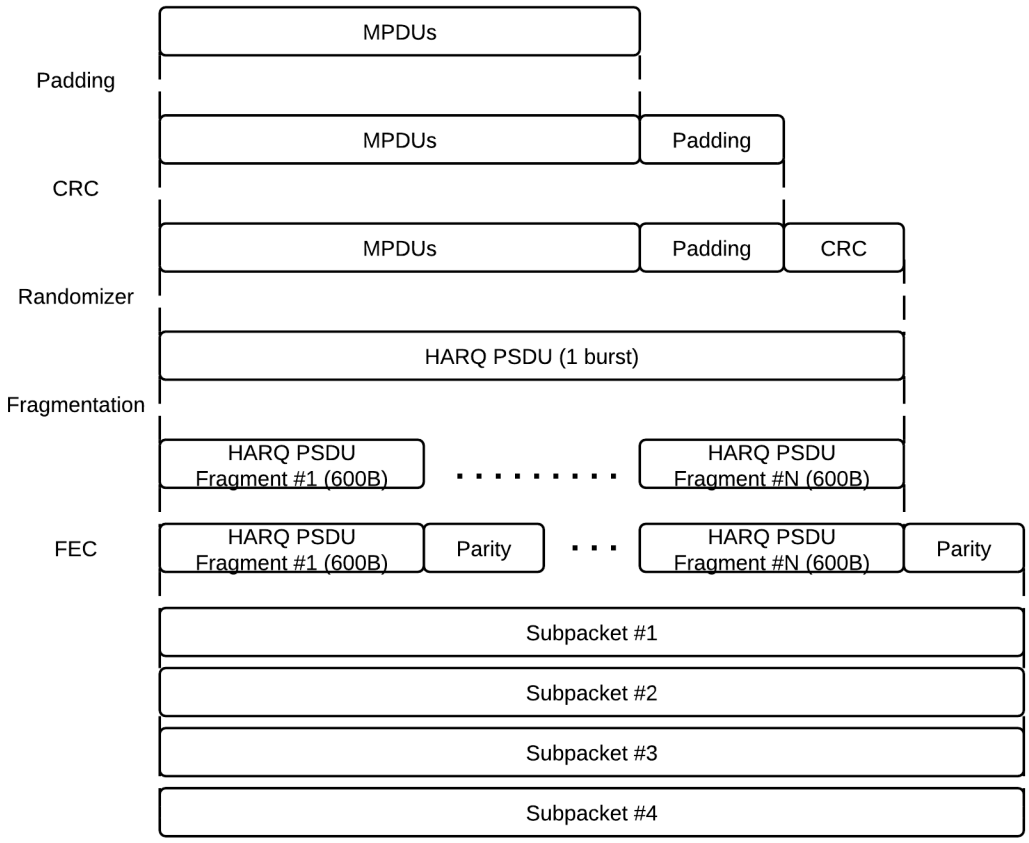


Figure 3-5: WiMAX MAC layer with HARQ enabled.

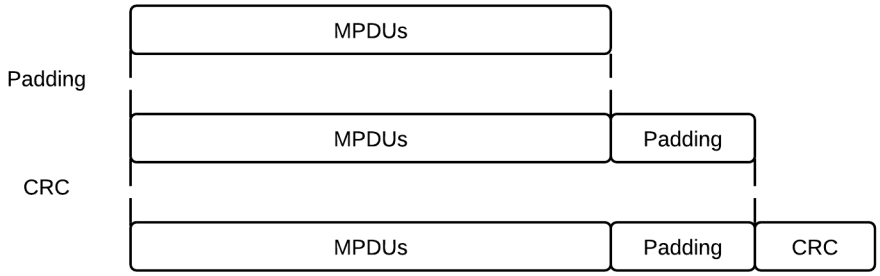


Figure 3-6: WiMAX MAC layer with HARQ disabled.

3.2.1 HARQ and SNR

We now discuss some advantages and disadvantages of HARQ. Dias et al. [15] shows that at low SNR, both CC and IR HARQ techniques provide a significant benefit. However, at high SNR, there is no apparent benefit from HARQ, since most of the FEC blocks are decoded

Table 3.2: HARQ Parameters and Description

Parameters	**Description**
HARQ_MAX_UL_BURST	The maximum number of HARQ UpLink (UL) bursts per frame.
HARQ_MAX_DL_BURST	The maximum number of HARQ DownLink (DL) bursts per frame.
HARQ_UL_ACK_DELAY	The frame offset of UL ACK delay with respect to UL Burst.
HARQ_DL_ACK_DELAY	The frame offset of DL ACK delay with respect to DL Burst.
HARQ_PDU_SN	Indicate whether PDU SN extended subheader should be applied by the transmitter on every PDU on this connection. This SN may be used by the receiver to ensure PDU ordering.
HARQ_MAX_RETRANSMIS-SION	The maximum number of retransmissions.

without error at the first transmission. Furthermore, HARQ incurs some overhead in terms of the redundant traffic, with its retransmissions and ACK or NACK packets. In addition, ACK or NACK packets may incur errors and delays because of poor channel conditions. Such errors and delays in acknowledgment packets may lead to additional redundant packet transmissions that are unnecessary.

3.2.2 HARQ and ARQ

Both ARQ and HARQ techniques pursue reliable delivery of data in the MAC of WiMAX. The WiMAX IEEE 802.16 standard specifies that a connection can be supported by both ARQ and HARQ schemes. However, [9] observes that ARQ and HARQ techniques have their own weaknesses. One such weakness is delay which may be caused by 1) in-order delivery and 2) intertwined ARQ and HARQ retransmissions. First, ARQ and HARQ that assure in-order delivery have to buffer all the out of order data units and reorder them, resulting in delay. Second, ARQ retransmissions are scheduled when HARQ retransmissions fail to deliver the data units. ARQ takes more retransmission time than HARQ, thus causing delay. Such delay increases the Round-Trip Time (RTT), resulting in a long delay of IP packets

service time [55].

3.3 Adaptive Modulation and Coding (AMC)

WiMAX supports a variety of modulation and coding schemes such as BPSK, QPSK, 16 QAM, and 64 QAM with Convolutional Codes (CC) or Convolutional Turbo Codes (CTC) at rates 1/2, 2/3, 3/4 and 5/6. In IEEE 802.16e-2005, 52 possible schemes are available and defined as burst profiles listed in Table 3.3. However, most implementations have fewer than 52 schemes.

Depending on channel conditions, the scheme can change on a per-user or per-frame basis. Using a channel quality feedback indicator such as Signal to Interference plus Noise Ratio (SINR) and Packet Error Rate (PER), Subscriber Stations (SSs) can provide the Base Station (BS) with feedback on the downlink channel quality. Based on the received signal quality, the BS can estimate the uplink channel quality. The BS takes into account the channel quality of each user's uplink and downlink and assigns a modulation and coding scheme that maximizes the throughput for the available SINR. To avoid an excessive number of dropped packets, the BS transmits at a lower rate when the channel is poor, and it transmits at as high a data rate as possible when the channel is good.

To achieve lower data rates, small constellations and low-rate error-correcting codes are used. To achieve the higher data rates, large constellations and high-rate less robust error correcting codes are used. AMC allows real-time trade-off between throughput and robustness on each link, significantly increasing the overall system capacity.

Figure 3-7 shows a block diagram of an AMC system. The transmitter attempts to transmit as fast as possible through a channel with a variable SINR, subject to the data being demodulated and decoded reliably at the receiver. To do so, feedback is crucial. Taking a channel quality feedback indicator such as channel SINR and PER, the AMC controller attempts to efficiently control the coding rate, transmit rate and transmit power.

In this chapter, we covered the basics of Automatic Repeated reQuest (ARQ), Hybrid Automatic Repeated Request (HARQ) and Adaptive Modulation and Coding (AMC) in

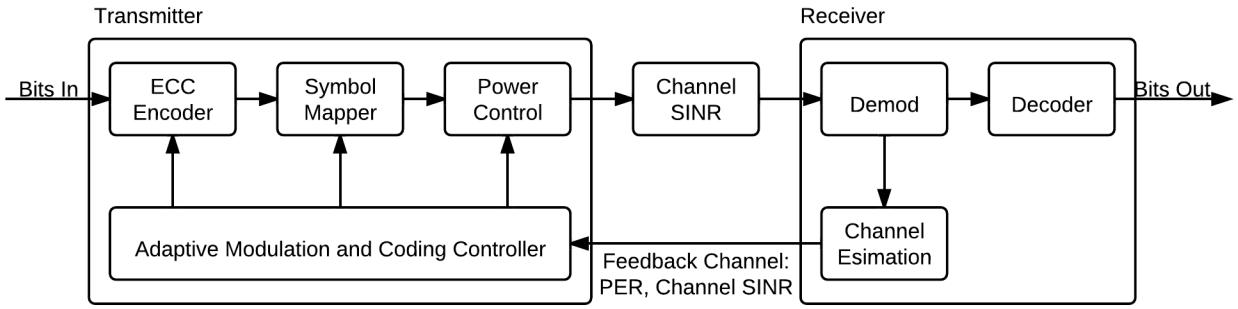


Figure 3-7: Adaptive modulation and coding block diagram.

Table 3.3: Available Uplink and Downlink Burst Profiles in IEEE 802.16e-2005. CC is Convolutional Code. CTC is Convolutional Turbo Code. 44-49 uses the optional interleaver with the convolutional codes. BTC is Block Turbo Code. ZCC is Zero-terminating Convolutional Code. 38-43 use the B code of Low-Density Parity-Check code (LDPC); other burst profiles with LDPC use A code

#	**Format**	#	**Format**	#	**Format**
0	QPSK CC 1/2	18	64 QAM CTC 1/2	36	64 QAM LDPC 2/3
1	QPSK CC 3/4	19	64 QAM CTC 2/3	37	64 QAM LDPC 3/4
2	16 QAM CC 1/2	20	64 QAM CTC 3/4	38	QPSK LDPC 2/3
3	16 QAM CC 3/4	21	64 QAM CTC 5/6	39	QPSK LDPC 3/4
4	64 QAM CC 1/2	22	QPSK ZCC 1/2	40	16 QAM LDPC 2/3
5	64 QAM CC 2/3	23	QPSK ZCC 3/4	41	16 QAM LDPC 3/4
6	64 QAM CC 3/4	24	16 QAM ZCC 1/2	42	64 QAM LDPC 2/3
7	QPSK BTC 1/2	25	16 QAM ZCC 3/4	43	64 QAM LDPC 3/4
8	QPSK BTC 3/4	26	64 QAM ZCC 1/2	44	QPSK CC 1/2
9	16 QAM BTC 3/5	27	64 QAM ZCC 2/3	45	QPSK CC 3/4
10	16 QAM BTC 4/5	28	64 QAM ZCC 3/4	46	16 QAM CC 1/2
11	64 QAM BTC 5/8	29	QPSK LDPC 1/2	47	16 QAM CC 3/4
12	64 QAM BTC 4/5	30	QPSK LDPC 2/3	48	64 QAM CC 2/3
13	QPSK CTC 1/2	31	QPSK LDPC 3/4	49	64 QAM CC 3/4
14	Reserved	32	16 QAM LDPC 1/2	50	QPSK LDPC 5/6
15	QPSK CTC 3/4	33	16 QAM LDPC 2/3	51	16 QAM LDPC 5/6
16	16 QAM CTC 1/2	34	16 QAM LDPC 3/4	52	64 QAM LDPC 5/6
17	16 QAM CTC 3/4	35	64 QAM LDPC 1/2		

WiMAX IEEE 802.16 standard [21]. In this thesis, the BS specifically implements IEEE 802.16e-2005 standard [22]. Its PHY uses OFDMA with a 10 Mhz channel bandwidth at 2.59 Ghz and a single antenna (without Multiple-Input Multiple-Output (MIMO) support). Its MAC uses TDD, and its HARQ only supports CC.

Chapter 4

Network-Coding-Enhanced Network Architecture

This chapter presents our NC-enhanced network architecture and its applications in point-to-point single- and multiple-interface networks.

4.1 NC-Enhanced Network Architecture

In the NC-enhanced network architecture, a network coding application is inserted into the network stack as shown in Figure 4-1. We use a Linux packet filtering framework (netfilter) [3] to intercept, copy and forward IP packets to the network coding application. Owing to this framework, the NC architecture is IP-based. After processing the filtered IP packets, the network coding application injects processed IP packets into the IP layer.

The network coding application is designed as a user-space application and can act as either a decoder or encoder process. Figure 4-2 shows the end-to-end packet flow path. Source applications in user space send outgoing IP packets to the Operating System (OS). Using netfilter, a network coding encoder process in user space intercepts those packets, codes them and sends coded IP packets to the operating system. Coded IP packets then traverse the WiMAX stack, passing through the Convergence Sublayer (CS), the upper MAC

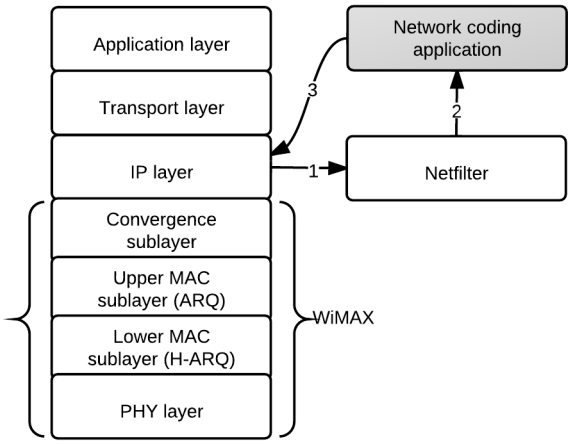


Figure 4-1: Network Architecture. Shows where our network coding application is inserted in the system. 1) IP packets are intercepted. 2) Netfilter copies and forwards IP packets to the network coding application. 3) The network coding application injects IP packets into the IP layer.

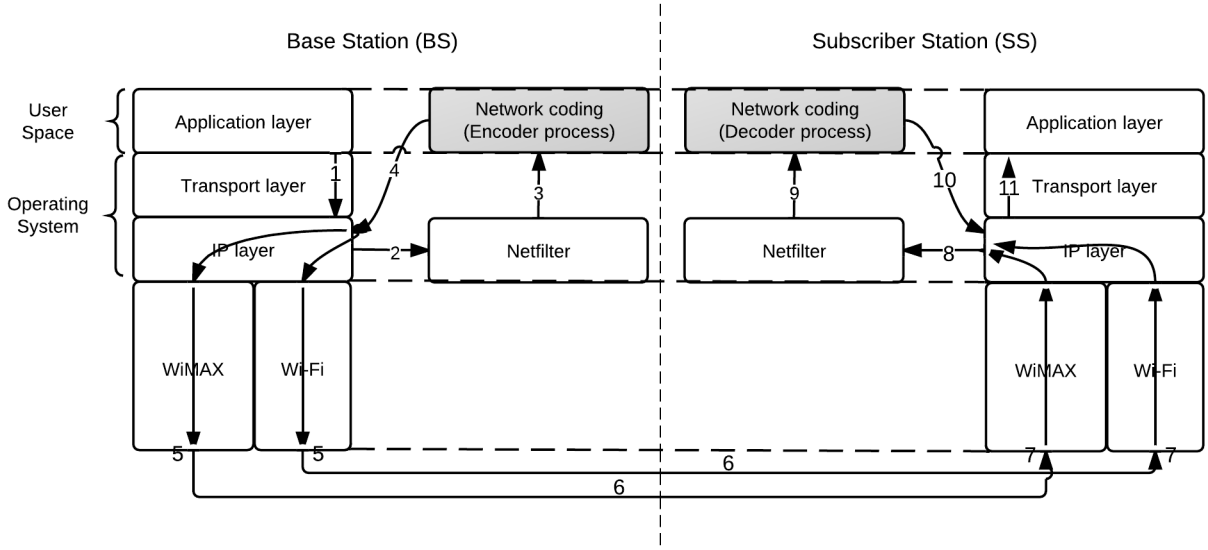


Figure 4-2: IP Packet Path. Shows the path of IP packets through the system. 1) Applications sends IP packets 2) Outgoing IP packets are intercepted. 3) Netfilter copies and forwards IP packets to network coding encoder process. 4) Network coding encoder process injects coded IP packets into the IP layer. 5,6,7) IP packets pass through WiMAX stack. 8) Incoming IP packets are intercepted. 9) Netfilter copies and forwards IP packets to network coding decoder process. 10) Network coding decoder process injects coded IP packets into the IP layer. 11) Applications receives IP packets.

sublayer where ARQ is run, the lower MAC sublayer where HARQ is run and the PHY layer. At the receiver, the OS receives incoming coded IP packets. Using netfilter, a network coding decoder process in user space intercepts those packets, decodes them and sends them to the OS, which forwards them to the destination applications.

The next section describes potential applications of this architecture in point-to-point single- and multiple-interface networks.

4.2 Applications

In this section, first, a straightforward application of the NC-enhanced network architecture in a point-to-point single-interface network is described, followed by the application in point-to-point multiple-interface networks.

4.2.1 Point-to-Point Single-Interface Networks

A point-to-point single-interface network provides communications between two endpoints (single-hop or multi-hop) using only a single interface at each endpoint. The interface may be different at the source and destination, e.g., Wi-Fi on one and WiMAX on the other.

In a point-to-point single-interface network, the architecture can be applied directly as shown in Figure 4-2. Although the figure depicts two WiMAX interfaces, the architecture is independent of the type of network interfaces used. In other words, Ethernet, Wi-Fi or LTE could be used instead of WiMAX.

4.2.2 Point-to-Point Multiple-Interface Networks

A point-to-point multiple-interface network provides communications between two endpoints (single-hop or multi-hop) using multiple interfaces on at least one endpoint. A number of recent wireless devices support multiple network interfaces. Yet, most applications only use one interface at a time.

Figure 4-3 shows how the NC architecture can enable such devices and applications to use multiple network interfaces at the same time. In Figure 4-3, there are two endpoints, both using Wi-Fi and WiMAX interfaces. With multiple network interfaces, the network coding encoder process can send coded IP packets to both Wi-Fi and WiMAX interfaces, and the network coding decoder process can receive coded IP packets from both interfaces. Not only can the architecture be applied through multiple network interfaces, but it can also be applied from multiple interfaces to a single interface and vice versa. For example, a device using Wi-Fi and WiMAX can send to a device using Ethernet only.

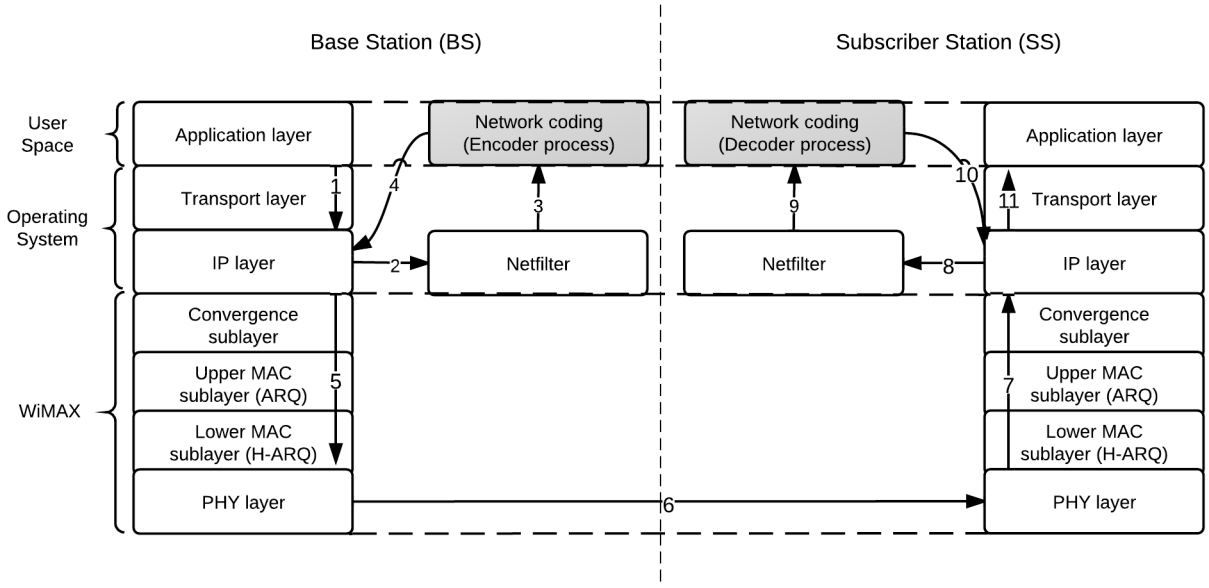


Figure 4-3: Multiple Interface Networks. Shows the application of architecture in multiple interface networks. 1) Applications sends IP packets 2) Outgoing IP packets are intercepted. 3) Netfilter copies and forwards IP packets to network coding encoder process. 4) Network coding encoder process injects coded IP packets into the IP layer and sends them via Wi-Fi and WiMAX. 5,6,7) IP packets pass through WiMAX and Wi-Fi stacks. 8) Incoming IP packets are intercepted. 9) Netfilter copies and forwards IP packets to network coding decoder process. 10) Network coding decoder process injects coded IP packets into the IP layer. 11) Applications receives IP packets.

In this chapter, we described how an IP-based NC-enhanced network architecture is constructed as well as how it is applied in point-to-point single and multiple interface networks. The following chapter will detail the design of a network coding application.

Chapter 5

Network Coding Application Design

This chapter presents the design of our network coding application. First, the encoder and decoder processes are described. Then, the design parameters are defined. Finally, the encoder, decoder and feedback mechanisms are discussed.

5.1 Encoder and Decoder Processes

The encoder or decoder processes have a master thread and a number of worker threads. Encoder and decoder worker threads operate in pairs identified by the Thread ID (TID) as shown in Figure 5-1. Figure 5-2 and 5-3 depict the encoder and decoder processes. The number of encoder worker threads in the encoder process must match the number of decoder worker threads in the decoder process. Each encoder-decoder thread pair encodes and decodes independently from other pairs. The encoder master thread load-balances encoder worker threads by distributing packets in a round-robin fashion. The decoder master thread dispatches incoming coded IP packets from an encoder worker thread to the corresponding decoder worker thread according to the TID.

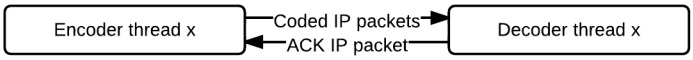


Figure 5-1: Encoder-decoder worker thread pair. Shows a pair of encoder-decoder threads, exchanging coded IP packets and an ACK packet.

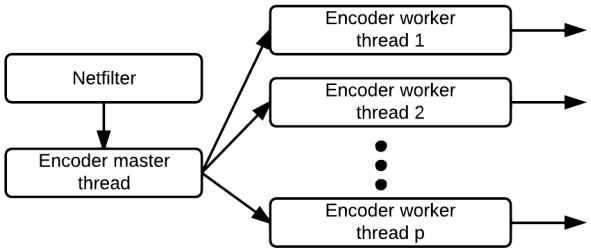


Figure 5-2: Encoder Process. Shows an encoder master thread and p different encoder worker threads. The master thread load-balances worker threads in a round-robin fashion.

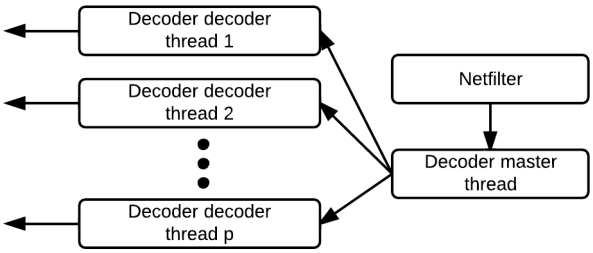


Figure 5-3: Decoder Process. Shows a decoder master thread and p different decoder worker threads.

5.2 Design Parameters and Variables

The proposed network coding application has a number of key design parameters defined in Table 5.1. These parameters can be specified by users. In addition, a few key design variables used throughout this and later chapters are defined in Table 5.2. We shall refer back to these parameters and variables as we describe the encoder, decoder and feedback mechanisms in the next sections.

Table 5.1: List of network coding application parameters and their description.

Parameters	**Description**
p	the number of concurrent encoder-decoder thread pairs
h	the processing length threshold of the buffer list
i	the processing time interval of the buffer list
u	the maximum length of segments
n	the preferred number of segments
k	the number of rounds of redundancy transmission
m	the number of redundancy packets per round
t	the time interval between each round

Table 5.2: List of network coding application variables and their description.

Variables	**Description**
s_l	the calculated segment length
n_s	the calculated number of segment
t_l	the total length of an outgoing IP packet

5.3 Encoder, Decoder and Feedback Mechanisms

5.3.1 Encoder Mechanism

The full encoder mechanism is shown in Figure 5-4. Incoming IP packets are first buffered at the master thread. The successive buffered IP packets form a buffer list. At the master thread, Algorithm 1 and Algorithm 2 run concurrently and determine when the buffer list is distributed to the worker threads. While Algorithm 1 applies a timeout mechanism,

Algorithm 2 applies a size trigger for buffer list concatenation. Together, they ensure that concatenation occurs before time interval i or buffer length threshold h are reached.

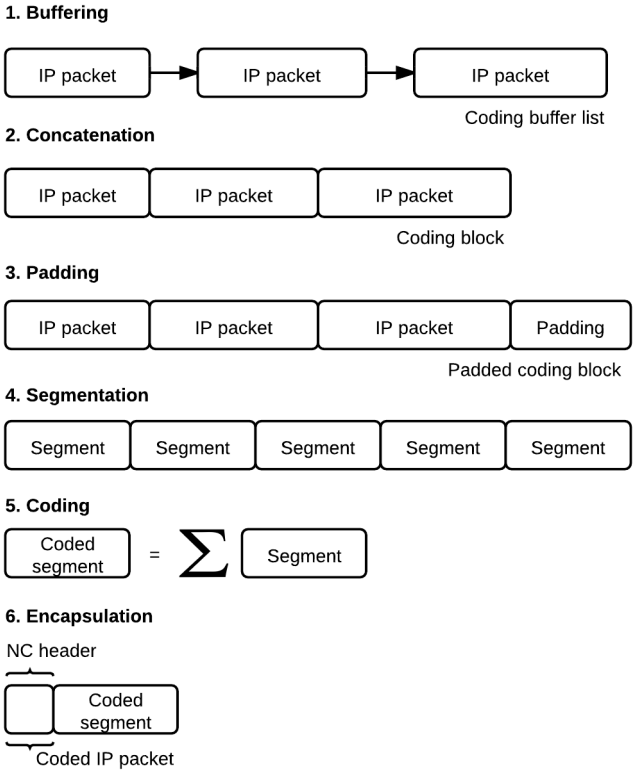


Figure 5-4: Encoder Mechanism. Shows the successive steps of the proposed encoder mechanism. 1) Incoming IP packets are buffered at the master thread forming a coding buffer list. Algorithm 1 and Algorithm 2 run concurrently and determine when the buffer list is distributed to the worker threads. 2) At each worker thread, the list is concatenated into coding block. 3) The number of segments (n_s) and segment length (s_l) are calculated according to Algorithm 3, and byte padding is added. 4) The block is segmented. 5) The resulting segments are coded according to Algorithm 4. 6) Encapsulation produces the coded IP packet.

Algorithm 1 An algorithm to decide when to concatenate the coding buffer list. i is the input time interval to concatenate the buffer list. The algorithm is run in its own separate thread.

1: **while** true **do**
2: wait for duration i
3: concatenate the coding buffer list
4: **end while**

Algorithm 2 An algorithm to decide when to concatenate the coding buffer list. b_l is the current length of the buffer list. p_l is the length of an incoming IP packet. h is the processing length threshold of the buffer list. The algorithm is run for every IP packet.

1: $b_l \leftarrow b_l + p_l$
2: **if** $b_l > h$ **then**
3: concatenate the coding buffer list
4: **end if**

At each worker thread, the buffer list is concatenated into a coding block. After that, the number of segments (n_s) and segment length (s_l) are calculated according to Algorithm 3. Algorithm 3 first adds 1 byte for the padding boundary to the length of coding block length (b_l). Then, the segment length (s_l) is initialized at $\frac{b_l}{n}$, and the number of segments (n_s) is initialized at the preferred number of segments (n). s_l and n_s are adjusted (line 5 and 6) to make s_l less than or equal to the maximum length of segments (u).

Algorithm 3 An algorithm to determine the segment length (s_l) and the number of segments (n_s), given the length of the coding block (b_l), the maximum length of segments (u), the preferred number of segments (n) and the length of the NC header without coding coefficients (l).

1: $b_l \leftarrow b_l + 1$ ▷ 1 byte for the padding boundary.
2: $s_l \leftarrow \frac{b_l}{n}$
3: $n_s \leftarrow n$
4: **while** $s_l > u$ **do**
5: $n_s \leftarrow n_s + 1$
6: $s_l \leftarrow \lceil \frac{b_l}{n_s} \rceil$
7: **end while**

Then, byte padding is added so that the padded block is a multiple of n_s. The block is then segmented and the resulting segments are coded according to Algorithm 4. Algorithm 4 first generates n_s uncoded or coded segments, depending on whether systematic network coding is used or not. Then, for additional segments, coded segments are generated with random coefficients. Finally, encapsulation produces the coded IP packet from the segment generated. Figure 5-5 shows the mechanism as a protocol in terms of Service Data Units (SDUs) and Protocol Data Units (PDUs).

The structure of the NC header used in the encapsulation process is shown in Figure 5-

Algorithm 4 Encoding Algorithm. n_s is the number of segments. k is the number of rounds of redundancy transmission. m is the number of redundancy packets per round. t is the time interval between each round. The algorithm is terminated immediately when ACK of the same coding block is received.

```
1: for x = 1 → n_s do
2:     generate an uncoded or coded segment.
3: end for
4: while ACK has not yet been received. do
5:     for y = 1 → k do
6:         for z = 1 → m do
7:             generate a coded segment.
8:         end for
9:         wait for duration t            ▷ terminate immediately when an ACK is received.
10:    end for
11: end while
```

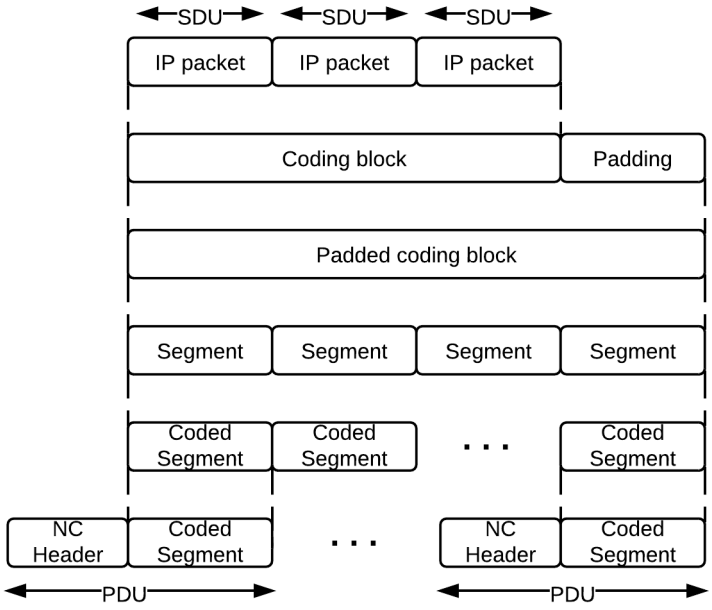


Figure 5-5: Service Data Units (SDUs) to Protocol Data Units (PDUs).

6. The encapsulation header contains the IP header, Thread ID (TID), Block ID (BID), Segment ID (SID), the number of segments (n_s), and coding coefficients. The segment length (s_l) is not included in the encapsulation because it can be derived from the packet length field in the IP header. TID is used to identify which packet belongs to which thread. BID is used to identify which block in a thread a packet belongs to. For a particular thread, BID is incremented for every coding block. BID is used to differentiate coded IP packets of different blocks of the same thread. When a packet with a different BID from the current block arrives, and the decoder has not yet decoded the current block, the decoder will drop the current undecoded block and start decoding the new block. GID is used to keep track of how many segments for a particular block have been generated. For a particular coding block, GID is incremented for every IP packet generated. n_s, s_l and coding coefficients are necessary for decoding.

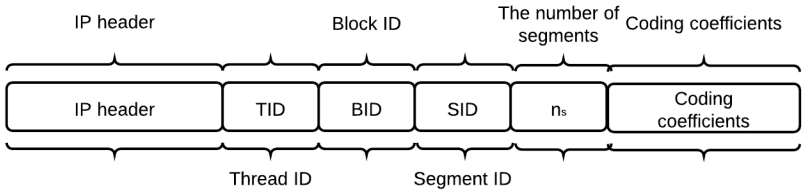


Figure 5-6: NC Header Encapsulation. The NC header contains the IP header, Thread ID (TID), Block ID (BID), Segment ID (SID), the number of segments (n_s), and coding coefficients.

5.3.2 Decoder Mechanism

Incoming coded IP packets are decapsulated, decoded, desegmented, depadded and deconcatenated to get the uncoded IP packets as shown in Figure 5-7. Decapsulation strips off the NC header. For each coded block, incoming coded IP packets are decoded progressively using Algorithm 5, which is based on Gauss-Jordan elimination [56]. Once the block is decoded, it is desegmented, depadded and deconcatenated, yielding the uncoded IP packets.

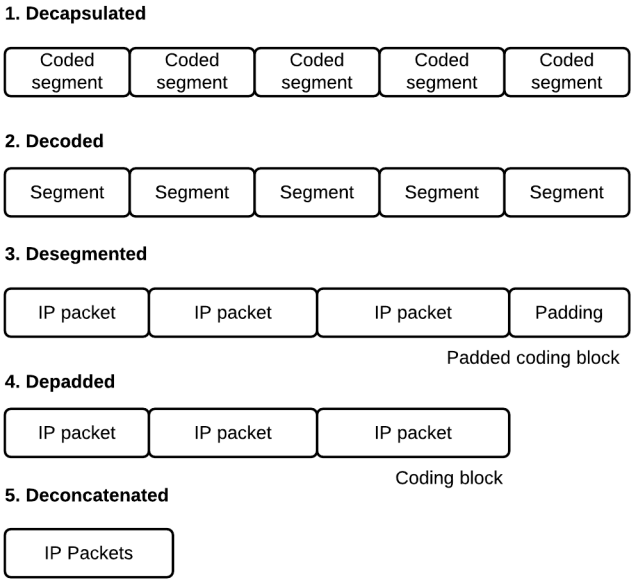


Figure 5-7: Decoder Mechanism. Shows the successive steps of the decoder mechanism. Incoming coded IP packets are decapsulated, decoded, desegmented, depadded and deconcatenated to get the uncoded IP packets.

Algorithm 5 Decoding algorithm. $\mathbf{M}$ is the current coefficient matrix of incoming coded packets. $\mathbf{M}[r+1]$ refers to row $r+1$ of $\mathbf{M}$. $rank(\mathbf{M})$ is the rank of $\mathbf{M}$. This algorithm runs per coded block.

$r \leftarrow 0$
$\mathbf{M}_{n_s \times (n_s + s_l)} \leftarrow \mathbf{0}$
for each incoming coded IP packet P **do**
    $\mathbf{M}[r+1] \leftarrow$ coefficients and segment of P
    Gauss-Jordan elimination on $(r+1) \times (n_s + s_l)$ of $\mathbf{M}$
    if $rank(\mathbf{M}) = r + 1$ **then**
        $r \leftarrow r + 1$
        if $r = n_s$ **then**
            done decoding
        end if
    end if
end for

5.3.3 Feedback Mechanism

Once a decoder decodes a block, it sends an ACK packet to the corresponding encoder identified by TID. Figure 5-8 shows the structure of an ACK packet. If the encoder worker threads are still running the coding algorithm on the block with the same BID as that in the ACK packet, the worker will terminate Algorithm 4 on that block. Note that the encoder does not require ACK packets to operate, since ACK packets can have errors or be lost.

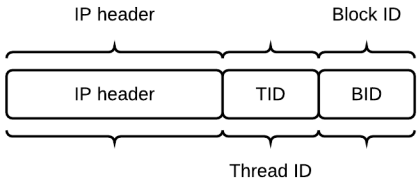


Figure 5-8: Structure of an ACK packet. An ACK packet contains the IP header, Thread ID (TID), Block ID (BID).

5.4 Design Analysis

This section discusses the design presented. We first discuss the advantages of NC over ARQ and HARQ. Then, we analyze the code rate and overhead of the design. Finally, we look at potential limitations of the design.

First, the design has advantages over ARQ and HARQ in that NC does not rely on an acknowledgement (ACK). ACK can have errors, get lost or get delayed. In ARQ and HARQ, a packet is unacknowledged if it has been transmitted but no acknowledgement has been received. In NC, a successfully transmitted packet is useful as long as it is linearly independent from the ones already received.

5.4.1 Code Rate

Code Rate (CR) of the presented design is defined as the ratio of the number of segments (n_s) by the sum of the number of segments and the redundancy segments ($n_s + k \times m$):

$$CR \equiv \frac{n_s}{n_s + k \times m}.$$

5.4.2 Overhead

The total NC header length is $l + n_s$, where l is the length of the NC header without coding coefficients and n_s is the number of segments. The NC header overhead ratio is

$$\frac{l + n_s}{s_l}, \tag{5.1}$$

where s_l is the segment length. If n_s is 120, l is 24, and s_l is 1400, the overhead is 10.29%. This overhead can be reduced in three ways: 1) by increasing the maximum length of segments (u), thus increasing s_l, 2) by reducing n_s and 3) by sending a seed of a pseudo-random number generator instead of a coefficient vector. If 3) is implemented, the overhead becomes

$$\frac{l + q}{s_l}, \tag{5.2}$$

where q is the size of the seed value, which is typically 4 bytes. If l is 24, and s_l is 1400, the overhead is 2% which is much lower. Next, we discuss potential limitations of the design and their mitigation.

5.4.3 Limitations

Potential limitations arise from coded packet duplication and reordering. With intra-session network coding, every packet received contains the same amount of new information. As long as there are enough independent segments, the block can be decoded. Thus, coded packets within the same block can be reordered or duplicated, and the block can still be

decoded. Nevertheless, there are problems when packets from different blocks duplicate or reorder. Let's begin with the definition of a defective block.

Definition 1. *A block is defective if it cannot be fully decoded.*

The decoder needs to have enough consecutive coded packets from the same block to fully decode the block. Thus, when there are not enough consecutive coded packets from the same block, the block is defective (Definition 1). Figure 5-9 shows the effects of reordering or duplication on the input packet stream at the receiver.

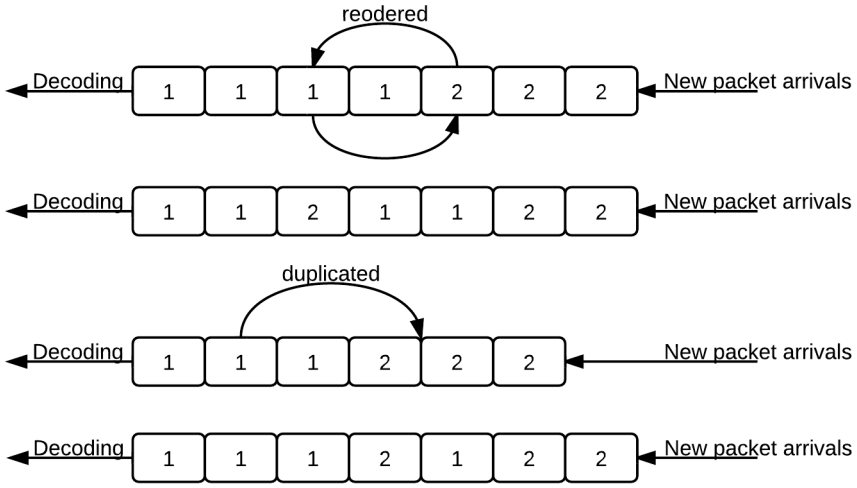


Figure 5-9: Duplication and reordering. Segments of block 1 and 2 from the same thread are reordered or duplicated. Block 1 becomes defective; block 2 loses a coded packet.

Duplication and reordering shorten the number of consecutive packets of the same block as depicted in Figure 5-9. Since a new BID triggers the drop of the previous undecoded block, duplication and reordering increase the chance of having defective blocks. This issue can be mitigated by increasing the number of concurrent encoder-decoder thread pairs (p). In Figure 5-10, block 1 and 2 from the white thread are interleaved with block 1 from the grey thread; block 1 and 2 from the white thread are less likely to reorder or duplicate among themselves and thus to be defective. Nevertheless, note that increasing p will increase the reordering of the uncoded packets since each encoder-decoder thread pair operates independently.

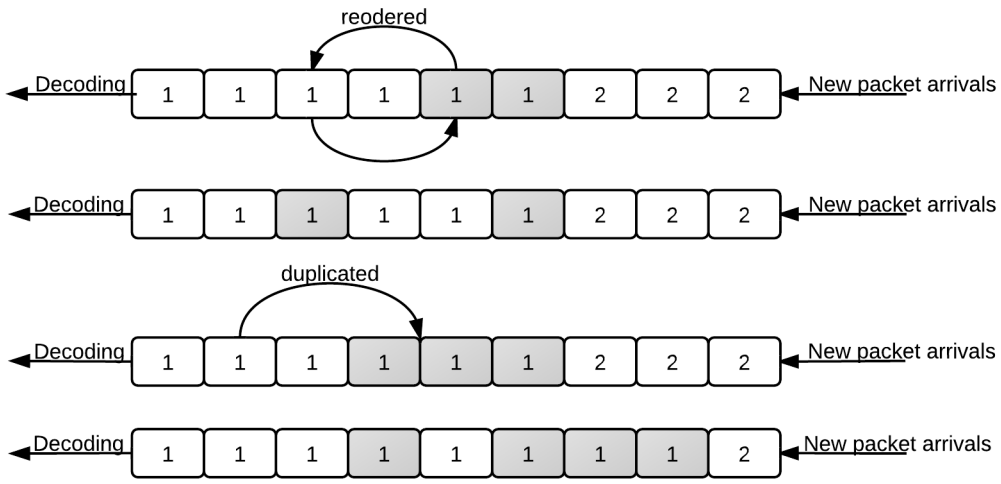


Figure 5-10: Mitigation by increasing the number of threads. Segments of blocks from one thread (white) are interleaved with those of blocks from another thread (gray). Segments of block 1 from thread 1 are reordered or duplicated with those of block 1 from thread 2. None is defective.

5.5 Extension

One possible extension is to add an adaptive controller. Figure 5-11 shows the design with such a controller. Based on the feedback, the controller can adjust the design parameters accordingly. For example, based on channel SINR, the controller can predict packet loss and adjust the number of redundant coded packets in advance. Such an extension is left for further study.

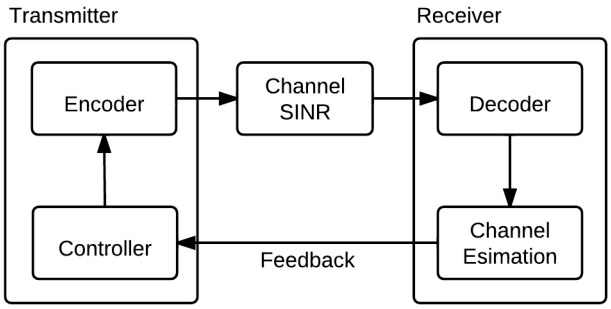


Figure 5-11: A block diagram of the design with a controller.

In this chapter, we presented the design of the network coding application, including the encoder and decoder process, the design parameters, the encoder, decoder and feedback

mechanisms and the potential limitations of the design. A design with an adaptive controller was suggested. The next chapter discusses the implementation of the design.

Chapter 6

Implementation

This chapter discusses some of our design and implementation decision. In particular, we discuss our implementation of network coding, decoding algorithm and random seeds used to generate the code. Then, we discuss the implementation of the NC header and the data padding.

6.1 Network Coding

We implement Random Linear Network Coding (RLNC) in a Galois Field of size 2^8 or $\mathbf{GF}(2^8)$, which is considered sufficient for practical applications [13, 28] and aligns nicely in a single byte unit. In our implementation, systematic network coding is used. Algorithm 4 first generates n_s uncoded segments; their coding coefficients only have one entry of value one, and other entries are zero. Then, for additional segments, coded segments are generated with random coefficients. Systematic network coding has been shown to help speed up the decoding time [44] which is important to maintain low end-to-end delay.

6.2 Decoding Algorithm

A progressive Gauss-Jordan elimination as described in Section 5.3.2 with a single twist is used for decoding. Since systematic network coding is used, the blocks that cannot be

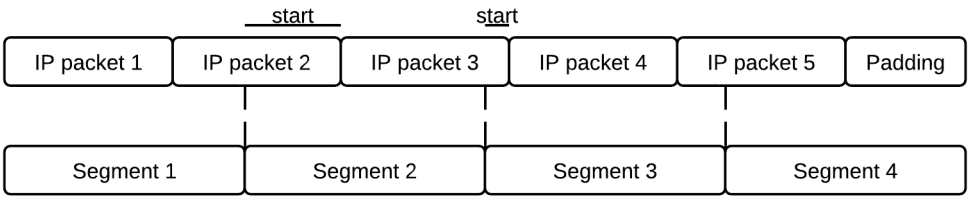


Figure 6-1: Segmented IP packets.

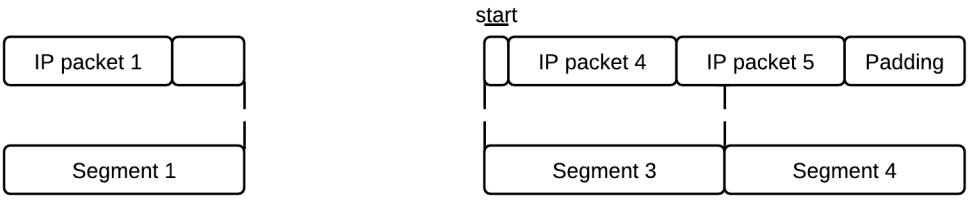


Figure 6-2: Segmented IP packets. Segment 2 is lost. IP Packets 1, 4 and 5 can be recovered.

decoded still contain useful information and some packets can still be extracted from them because of the structure of the block itself. In order to do this we need to know where an IP packet starts in a segment. Hence we added an additional 2 byte field in the NC header called *start*. In other words, *start* allows us to pinpoint the start of an IP packet in a segment. Figure 6-1 and 6-2 show an example of how *start* is used. In Figure 6-2, although segment 2 is lost, IP packets 1, 4 and 5 can still be recovered.

6.3 Random Seeds

In this section, we describe the use of random seeds instead of the actual coefficient vectors to reduce the overhead of NC header [32, 33, 60]. In order to support random seeds, we added new fields to the NC header: *type* and either segment number (*segn*) or *seed*. The parameter *type* is used to distinguish whether a packet is an uncoded packet or a coded one. The parameter *segn* is used in a systematic packet to specify the segment number or the position of the one in the $(..., 0, 1, 0, ...)$ coefficient vector. The parameter *seed* is used in a coded packet as a random seed. We use a simple pseudo-random number generator described in Algorithm 6.

Algorithm 6 A simple pseudo-random number generator known as Gerhard's generator. a is initialized to 1. Given the seed a, the function generates a pseudo-random number from 1 to lim.

```
1: a ← 1
2: function RAND(lim)
3:     a ← (a × 32719 + 3)  mod 32749
4:     return (a  mod lim) + 1
5: end function
```

6.4 NC Header

In our implementation, the length of each field in the NC header and ACK packet is chosen as follows. The IPv4 header (Figure 6-3) is used; it is 20 bytes. Figure 6-4 and 6-5 show the headers for a systematic packet and a coded packet respectively. The length of TID, BID, SID and n_s are 1 byte each. *type* and *segn* are also 1 byte each; *seed* is 2 bytes. The length of the coding coefficients is n_s bytes. Thus, the total length of the NC header is $24+n_s$ bytes. The header contains the following information. The IP Version is set to 4. The Internet Header Length (IHL) is set to 5 words. The Type of Service (ToS) is set to 0. The total length (t_l) is set to $24 + n_s + s_l$. The ID, the flags and the fragment offset are set to 0. The Time To Live (TTL) is set to 255. The Protocol is set to 252. This protocol number was chosen arbitrarily. The checksum is calculated according to RFC 1071 [23]: "the checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero." Finally, s_l needed to decode can be calculated from the Total Length (t_l) as follows.

$$s_l = t_l - 24 - n_s.$$

With these choices, there are limitation on the values of TID, BID, SID, s_l, n_s, *type*, *segn* and *seed*, thus restricting the scope of the values of the design parameters. For example, the values of TID, BID, SID, n_s, *type* and *segn* are limited to 255. The limits should be considered when choosing these parameters.

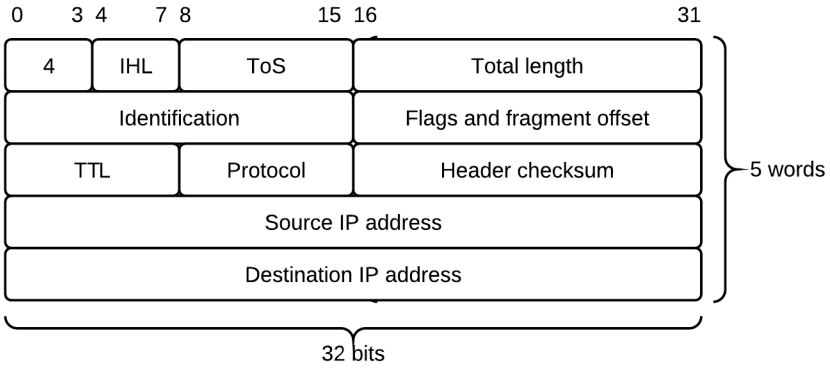


Figure 6-3: Structure of IPv4 header.

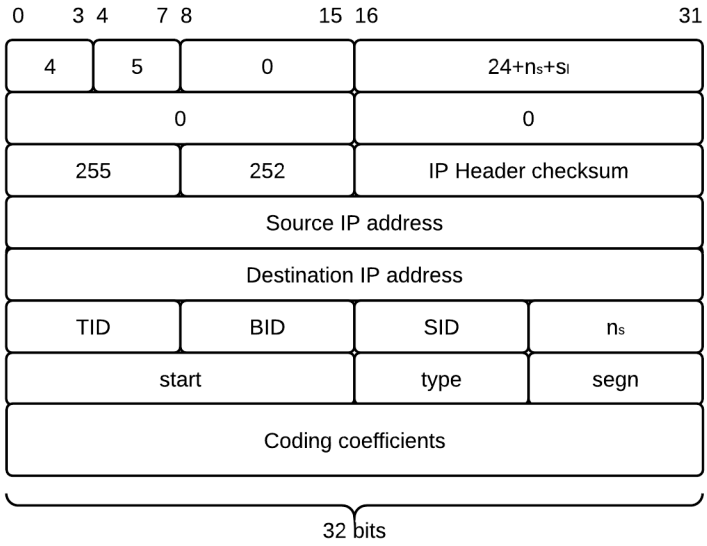


Figure 6-4: Structure of the NC header of a systematic packet.

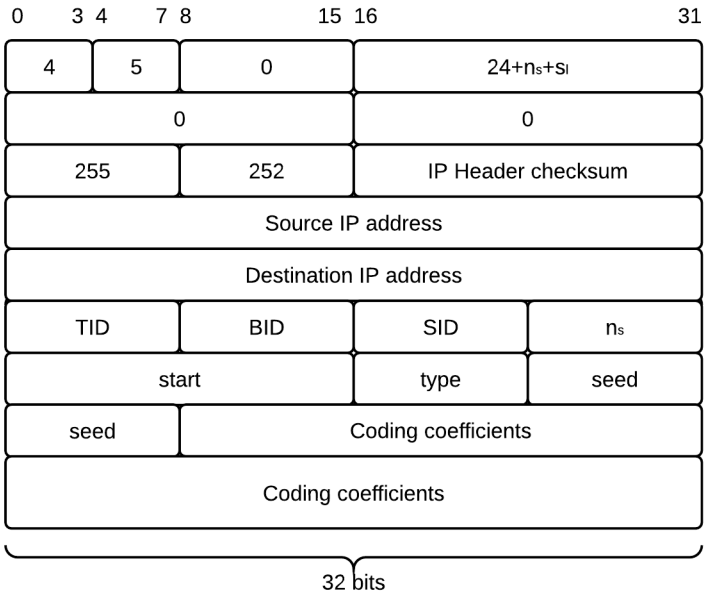


Figure 6-5: Structure of the NC header of a coded packet.

6.5 Padding

Coding blocks have to be padded such that their length is a multiple of the number of segments (n_s). Our algorithm is the ANSI X.923 byte padding algorithm [4] which is well known. In ANSI X.923, bytes filled with zeros are appended to the data and the last byte stores the number of padded bytes.

In this chapter, our implementation is discussed in detail, including the detailed packet and header structure as well as the chosen network coding and padding algorithms. Some implementation aspects that were not included in our implementation are the use of Streaming SIMD Extension (SSE) instruction set, the use of Graphical Processing Unit (GPU) acceleration and the use of Jacobi method for iterative matrix inversion instead of Gauss-Jordan elimination, all of which leading to a further to speed up gain in decoding. Haley et al. [24] show how the Jacobi method for iterative matrix inversion can be applied to finite field matrices. The next chapter discusses experimental results which will clearly demonstrate that we did not need further decoding performance.

Chapter 7

Preliminary Experiments

In this chapter, the preliminary experiments on WiMAX technology are discussed. The experimental setup, the results of our measurements and their discussion are presented. The objective of the preliminary experiments is to measure the communication channel and the network performance of different configurations of WiMAX Base Stations (BSs), focusing on turning on and off HARQ and ARQ. Note that these experiments are done without network coding. The network coding experiments will be discussed in the next chapter.

The WiMAX downlink is selected to evaluate the performance and User Datagram Protocol (UDP) traffic is used as representative of real-time traffic through the system. The Global Environment for Network Innovations (GENI) WiMAX platforms are used. The platforms leverage commercial 802.16e WiMAX Base Stations (BSs) from NEC without MIMO support. The Subscriber Station (SS) uses Intel WiMAX 6250 Series modem card. The SS is installed with Ubuntu 10.10 and has Intel WiMAX Linux driver version 1.5.1. More information about the platforms can be found in [1].

We collected measurements from WiMAX BSs from two different sites: at Raytheon BBN Technologies (BBN) in Cambridge, Massachusetts and at the University of California, Los Angeles (UCLA). The first set of measurements was collected at BBN over several days in May, June and July of 2011. The second set of measurements was collected on the UCLA campus from August 2, 2011 to August 5, 2011. The objective functions are the throughput.

Table 7.1: Base Station Parameters.

Parameters	**Value**
PHY	OFDMA
Frequency	2.59 Mhz
Bandwidth	10 Mhz
Duplexing mode	TDD
Frames per second	200 (5 ms per frame)
Uplink Modulation Coding Scheme (MCS)	Adaptive
Downlink Modulation Coding Scheme (MCS)	Adaptive
HARQ_TYPE	CC
HARQ_MAX_UL_BURST	1
HARQ_MAX_DL_BURST	1
HARQ_UL_ACK_DELAY	3 frames
HARQ_DL_ACK_DELAY	1 frame
HARQ_PDU_SN	ON
HARQ_MAX_RETRANSMISSION	4
ARQ_RETRY_TIMEOUT	1000 ms
ARQ_BLOCK_SIZE	256 Bytes
ARQ_WINDOW_SIZE	1024
ARQ_TX_ACK_DELAY	100 ms
ARQ_ACK_PROC_TIME	0 ms
ARQ_BLOCK_LIFETIME	5000 ms
ARQ_DLV_ORDER	ON
ARQ_RX_PURGE_TIMEOUT	5000 ms
ARQ_SYNC_LOSS_TIMEOUT	10000 ms

7.1 Setup

All experiments use the same default set of BS parameter values. Adaptive coding modulation is used by default. BS parameters are shown in Table 7.1. The only changes applied to the BS default parameter values is to enable or disable HARQ and ARQ as required by our experimental setup; they have their own set of values also described in Table 7.1.

When HARQ is turned on, the HARQ parameters are set as shown in Table 7.1. Chase Combining HARQ is used. The maximum number of HARQ UpLink (UL) bursts per frame is set to 1. The maximum number of HARQ DownLink (DL) bursts per frame is also set to 1. The frame offset of UL ACK delay with respect to UL Burst is set to 3. The frame offset of DL ACK delay with respect to DL Burst is set to 1. The PDU SN extended sub-header

reordering is enabled. The maximum number of retransmissions is set to 4.

When ARQ is turned on, the ARQ parameters are set as shown in Table 7.1. The minimum time interval a transmitter will wait before retransmission of an unacknowledged block is set to 1000 ms. The interval begins when the ARQ block was last transmitted. ARQ block size is set to 256 bytes. Before transmission each SDU block is partitioned into a sequence of ARQ blocks of this size. Transmission window size or the number of queued ARQ acknowledgement blocks at any given time for a connection is set to 1024. The time delay before the receiver sends an acknowledgement is set to 100 ms. ACK processing time is set to 0. The maximum time interval an ARQ block will be managed by the transmitter ARQ state machine, once initial transmission of the block has occurred is set to 5000 ms. If transmission (or subsequent retransmission) of the block is not acknowledged by the receiver before the time limit is reached, the block is discarded. In-order delivery is enabled. The time interval the receiver will wait after successful reception of a block that does not result in advancement of ARQ_RX_WINDOW_START value is set to 5000 ms. Lastly, the maximum time interval ARQ_TX_WINDOW_START or ARQ_RX_WINDOW_START parameters can stay at the same value before declaring a loss of synchronization between transmitter and receiver is set to 10000 ms.

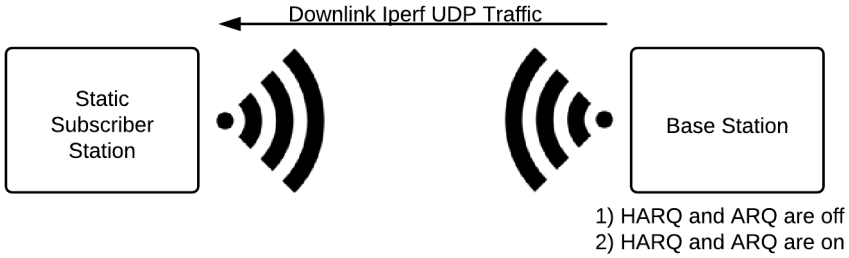


Figure 7-1: Preliminary Experiment Setup.

In the experimentation, we located a node at the BS and a static SS as shown in Figure 7-1 provided the other node. The experiments focus on the downlink channel from the BS to the static SS. Channel performance was measured using flows of UDP traffic at a few offered bandwidths and packet sizes. Iperf [2] is used to generate these flows and measure

the channel characteristics. Iperf is a network testing tool that can create UDP traffic and measure the throughput of a network. We measure throughput in two configurations: 1) when HARQ and ARQ are turned off and 2) when HARQ and ARQ are turned on.

7.2 Results

Results from three different experiments are given: two of these experiments were performed at BBN and one experiment was performed at UCLA. At BBN, the BS is located on top of the BBN building; the SS is static and located on the fifth floor inside the building. At UCLA, the BS is on top of Boelter Hall; the SS is static, 100 feet away from the BS and within the BS's line-of-sight.

7.2.1 BBN Experiment #1

The results of the first experiment at BBN are illustrated in figure 7-2. In this experiment, we measure the throughput over 60 seconds at 5 Mbps offered load for different packet sizes. The average Round-Trip Time (RRT) between the BS and the SS is around 80 ms. Received Signal Strength Indication (RSSI) is -78 dBm and Carrier to Interference-plus-Noise Ratio (CINR) is 10 dB.

We notice two interesting points from these results. First, when HARQ and ARQ are turned off, the 1440 bytes packet size has around 37% higher throughput than that of the 70 bytes packet size. Second, when HARQ and ARQ are turned on, the throughput reduces significantly in all packet sizes. The percentage reduction in throughput for each packet size is given in Table 7.2.

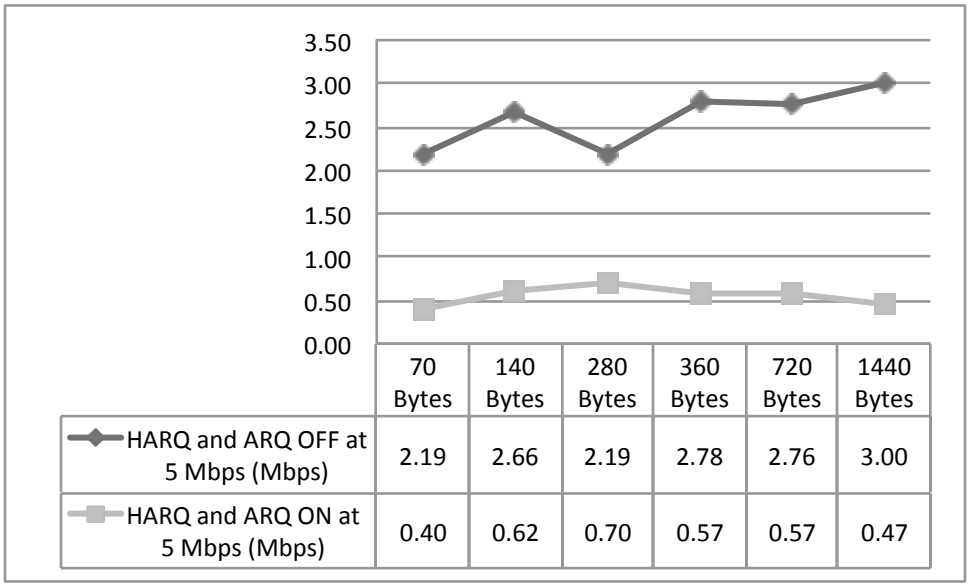


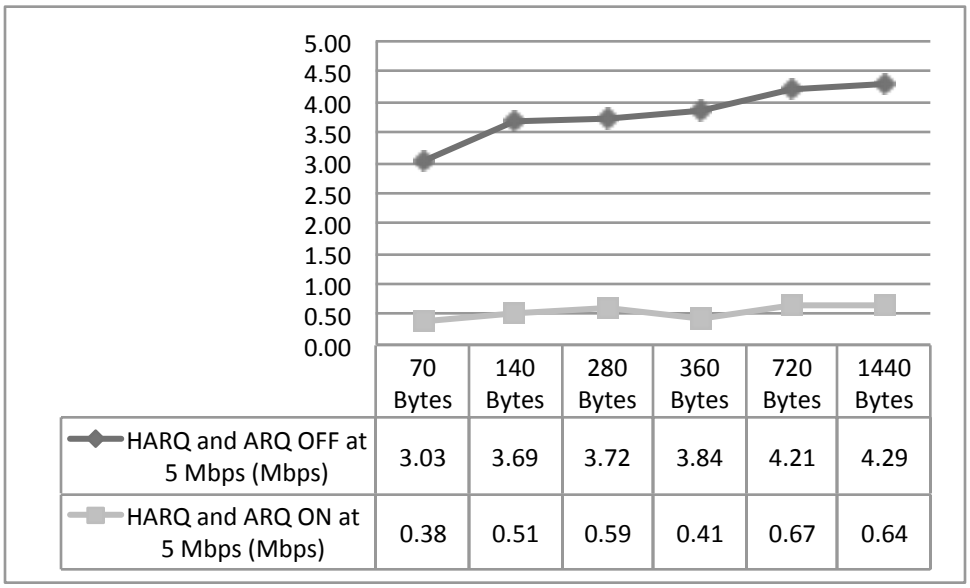
Figure 7-2: BBN Experiment #1. Shows average downlink throughput (Mbps) over 60 seconds for different packet sizes (bytes) at 5 Mbps offered load when HARQ and ARQ are off and HARQ and ARQ are on. The BS is located on top of the BBN building; the SS is static and on the fifth floor inside the BBN building.

Table 7.2: BBN Experiment #1. Shows percentage reduction in throughput for different packet sizes.

Packet Size	**Percentage Reduction**
70 bytes	82%
140 bytes	77%
280 bytes	68%
360 bytes	79%
720 bytes	79%
1440 bytes	84%

7.2.2 BBN Experiment #2

In order to confirm the results of the first experiment, a second set of measurements was taken. In this experiment, the average RTT between the BS and the SS is around 80 ms. RSSI is around -76 dBm and CINR is around 11 dB. The offered load is 5 Mbps; the throughput is averaged over 60 seconds. The results are shown in Figure 7-3.



	70 Bytes	140 Bytes	280 Bytes	360 Bytes	720 Bytes	1440 Bytes
HARQ and ARQ OFF at 5 Mbps (Mbps)	3.03	3.69	3.72	3.84	4.21	4.29
HARQ and ARQ ON at 5 Mbps (Mbps)	0.38	0.51	0.59	0.41	0.67	0.64

Figure 7-3: BBN Experiment #2. Shows average downlink throughput (Mbps) over 60 seconds for different packet sizes (bytes) at 5 Mbps offered load when HARQ and ARQ are off and HARQ and ARQ are on. The BS is located on top of the BBN building; the SS is static and on the fifth floor inside the BBN building.

We notice similar results as in the first experiment. When HARQ and ARQ are turned off, the 1440 bytes packet size has around 42% higher throughput than that of 70 bytes packet size. This experiment also confirms the previous observations that turning on HARQ and ARQ reduces the throughput for various packet sizes. The percentage reduction in throughput for each packet size is given in Table 7.3.

Table 7.3: BBN Experiment #2. Shows percentage reduction in throughput for different packet sizes.

Packet Size	Percentage Reduction
70 bytes	87%
140 bytes	86%
280 bytes	84%
360 bytes	89%
720 bytes	84%
1440 bytes	85%

7.2.3 UCLA Experiment

In the summer of 2011, we conducted another set of measurements at UCLA with different offered loads and packet sizes. In this experiment, the average RTT is also around 80ms. RSSI is around -44 dBm and CINR is around 30 dB. The average transmission power of the SS is around -54 dBm. The offered load is 20 Mbps; the throughput is averaged over 120 seconds. The results are shown in Figure 7-4.

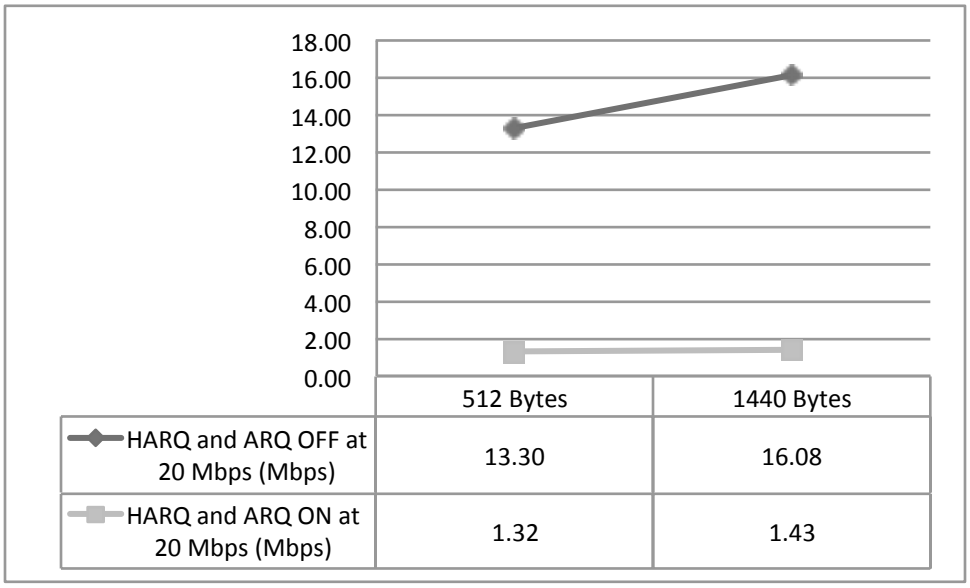


Figure 7-4: UCLA Experiment. Shows average downlink throughput (Mbps) over 120 seconds for different packet sizes (bytes) at 20 Mbps offered load when HARQ and ARQ are off and HARQ and ARQ are on. The BS is on top of Boelter Hall at UCLA; the SS is static, 100 feet away from the BS and within BS's line-of-sight.

This experiment shows similar and consistent results with those of the BBN experiments:

1) when HARQ and ARQ are off, the larger packets have a higher throughput than that of the smaller packets, and 2) turning on HARQ and ARQ reduces the throughput for all sizes The percentage reduction in throughput for each packet size is given in Table 7.4.

Table 7.4: UCLA Experiment. Shows percentage reduction in throughput for different packet sizes.

Packet Size	**Percentage Reduction**
512 bytes	90%
1440 bytes	91%

7.3 Discussion

From the results, we infer that, first, the longer the packet, the better the throughput and second that HARQ and, second, that ARQ significantly reduce the throughput. The first point is most likely due to the overhead cost of transmitting each packet. Given the fixed offered bandwidth, with smaller packet size, more packets are transmitted; thus, greater overhead is incurred. For the second point, we see significant reduction in throughput when HARQ and ARQ are on: over 68% reduction in throughput in all experiments. This shows that HARQ and ARQ may not be optimal schemes, and that the retransmission incur both reduced available bandwidth for other transmissions as well as added delay per packet transmitted. This is consistent with other research in this field [32, 33]. Hence, our experimental results clearly show that there is a potential for alternative schemes to do better than HARQ and ARQ in terms of overall performance.

As a conclusion, in this chapter, we presented and discussed the results of the uncoded experiments. The preliminary results show that the HARQ and ARQ mechanisms may not be the optimal mechanisms to cope with the residual errors at the MAC layer and that there is a potential for network coding to act as packet erasure codes in wireless networks. The following chapter confirms this potential experimentally.

Chapter 8

Network Coding Experiments

In this chapter, the network coding experiments are discussed. The experimental setup, performance metrics, the results of our measurements and their discussion are presented. The objective of the network coding experiments is to validate the potential for network coding (NC) to replace HARQ and ARQ.

As in the preliminary experiments, the WiMAX downlink is selected to evaluate the performance and User Datagram Protocol (UDP) traffic is used as representative of real-time traffic through the system and the Global Environment for Network Innovations (GENI) WiMAX platforms are used.

In the network coding experiments, the objective functions are the loss percentage, the throughput and the file transfer delay. The results will be a function of the measurement application parameters, NC parameters and BS parameters (power level, uplink/downlink modulation and coding schemes, HARQ, ARQ). We conduct the experiments remotely using the GENI platform at Rutgers University in New Jersey.

8.1 Setup

Four fixed downlink MCSs and base station transmission power levels are considered: 64 QAM CTC 1/2 at 13 dBm, 64 QAM CTC 2/3 at 17 dBm, 64 QAM CTC 3/4 at 18 dBm

and 64 QAM CTC 5/6 at 20 dBm. The uplink MCS is fixed at QPSK CTC 1/2. With 10 Mhz channel bandwidth, each modulation and code rate has a different PHY-layer data rate as shown in Table 8.1 [8].The BS parameters are shown in Table 8.2. Table 8.2 also shows HARQ and ARQ parameters when they are turned on.

Table 8.1: PHY-layer data rate with 10 Mhz Channel bandwidth for different modulation and code rate.

Modulation and Code Rate	**PHY-Layer Data Rate (Mbps)**	
	Downlink	**Uplink**
QPSK, 1/2	5.040	**1.344**
64 QAM, 1/2	**15.120**	4.032
64 QAM, 2/3	**20.160**	5.376
64 QAM, 3/4	**22.680**	6.048
64 QAM, 5/6	**25.200**	6.720

When HARQ is turned on, the HARQ parameters are set as shown in Table 8.2. Chase Combining HARQ is used. The maximum number of HARQ UpLink (UL) bursts per frame is set to 1. The maximum number of HARQ DownLink (DL) bursts per frame is set to 1. The frame offset of UL ACK delay with respect to UL Burst is set to 3. The frame offset of DL ACK delay with respect to DL Burst is set to 1. The PDU SN extended sub-header reordering is enabled. The maximum number of retransmissions is set to 4.

When ARQ is turned on, the ARQ parameters are set as shown in Table 8.2. The minimum time interval a transmitter will wait before retransmission of an unacknowledged block is set to 100 ms. The interval begins when the ARQ block was last transmitted. ARQ block size is set to 256 bytes. Before transmission each SDU block is partitioned into a sequence of ARQ blocks of this size. Transmission window size or the number of queued ARQ acknowledgement blocks at any given time for a connection is set to 1024. The time delay before the receiver sends an acknowledgement is set to 100 ms. ACK processing time is set to 0. The maximum time interval an ARQ block will be managed by the transmitter ARQ state machine, once initial transmission of the block has occurred is set to 500 ms. If transmission (or subsequent retransmission) of the block is not acknowledged by the receiver before the time limit is reached, the block is discarded. In-order delivery is enabled. The

Table 8.2: Base Station Parameters.

Parameters	**Value**
PHY	OFDMA
Frequency	2.59 Mhz
Bandwidth	10 Mhz
Duplexing mode	TDD
Frames per second	200 (5 ms per frame)
Power level	Fixed and Varied
Downlink Modulation Coding Scheme (MCS)	Fixed and Varied
Uplink Modulation Coding Scheme (MCS)	Fixed at QPSK CTC 1/2
HARQ_TYPE	CC
HARQ_MAX_UL_BURST	1
HARQ_MAX_DL_BURST	1
HARQ_UL_ACK_DELAY	3 frames
HARQ_DL_ACK_DELAY	1 frame
HARQ_PDU_SN	ON
HARQ_MAX_RETRANSMISSION	4
ARQ_RETRY_TIMEOUT	100 ms
ARQ_BLOCK_SIZE	256 Bytes
ARQ_WINDOW_SIZE	1024
ARQ_TX_ACK_DELAY	0 ms
ARQ_ACK_PROC_TIME	0 ms
ARQ_BLOCK_LIFETIME	500 ms
ARQ_DLV_ORDER	ON
ARQ_RX_PURGE_TIMEOUT	500 ms
ARQ_SYNC_LOSS_TIMEOUT	1000 ms

time interval the receiver will wait after successful reception of a block that does not result in advancement of ARQ_RX_WINDOW_START value is set to 500 ms. Lastly, the maximum time interval ARQ_TX_WINDOW_START or ARQ_RX_WINDOW_START parameters can stay at the same value before declaring a loss of synchronization between transmitter and receiver is set to 1000 ms.

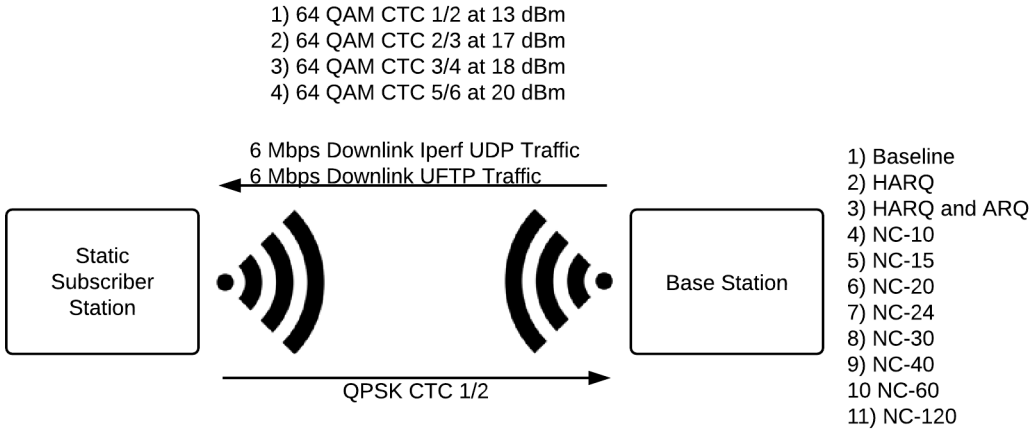


Figure 8-1: NC Experiment Setup.

We setup 2 nodes: a node at the BS and another node (SS) as shown in Figure 8-1. UDP traffic is considered; Iperf [2] and UFTP [11] are used as measurement applications. UFTP is a file transfer application that uses UDP based File Transfer Protocol. Iperf, UFTP and NC parameters are set as shown in Table 8.3, 8.4 and 8.5 respectively. For the Iperf parameters, the packet length is set to 1400 bytes. The offered load is set to 6 Mbps and the duration is set to 60 seconds. For the UFTP parameters, the packet length is also set to 1400 bytes. The offered load is set to 6 Mbps and the file size is 50 MB. For the NC parameters, the number of concurrent encoder-decoder thread pairs (p) is set to 1. The processing length threshold of the buffer list (h) is set to 22400 bytes. The processing time interval of the buffer list (i) is set to 1 s or 1000000000 ns. The maximum length of segments (u) is set to 1400 bytes. The preferred number of segments (n) is set to 120. The number of rounds of redundancy transmission (k) is set to 1. The number of redundancy packets per round (m) varies; we will experiment with different values of m. The time interval between each round is set to 0 ns.

Table 8.3: Iperf Parameters.

Parameters	**Value**
Packet length	1400 bytes
Offered load	6 Mbps
Duration	60 seconds

Table 8.4: UFTP Parameters.

Parameters	**Value**
Packet length	1400 bytes
Offered load	6 Mbps
File size	50 MB

Table 8.5: NC parameters.

Parameters	**Value**
p	1
h	22400 bytes
i	1000000000 ns
u	1400 bytes
n	120
k	1
m	Varied
t	0 ns

The offered load is fixed at 6 Mbps which is well below the downlink PHY-layer data rate of 64 QAM, 1/2, 64 QAM, 2/3, 64 QAM, 3/4 and 64 QAM, 5/6. For each MCS, 11 configurations are considered and shown in Table 8.6, where m is the number of redundancy packets per round. For each NC configuration, the approximate NC Code Rate (CR) defined in 5.4.1 is calculated and shown in Table 8.7. For instance, in NC-10, n is 120 and m is 10; 130 packets are sent, thus achieving a CR of 12/13. In theory, for the best performance, we want to match the CR with the throughput percentage of Baseline. For each configuration, we report

1. 6 Mbps offered load downlink Iperf loss percentage.
2. 6 Mbps offered load downlink Iperf throughput, lost bandwidth and extra bandwidth.
3. 6 Mbps offered load downlink Iperf Throughput to Loss plus Extra Ratio (TLER).
4. 6 Mbps offered load downlink UFTP file transfer delay.

Table 8.6: Experiment Configurations.

Configuration	**ARQ**	**HARQ**	**NC**
Baseline	OFF	OFF	OFF
HARQ	OFF	ON	OFF
HARQ-ARQ	ON	ON	OFF
NC-10	OFF	OFF	ON, $m = 10$
NC-15	OFF	OFF	ON, $m = 15$
NC-20	OFF	OFF	ON, $m = 20$
NC-24	OFF	OFF	ON, $m = 24$
NC-30	OFF	OFF	ON, $m = 30$
NC-40	OFF	OFF	ON, $m = 40$
NC-60	OFF	OFF	ON, $m = 60$
NC-120	OFF	OFF	ON, $m = 120$

Table 8.7: Code Rate (CR) for NC Configurations.

NC Configuration	Code Rate (CR)
NC-10	$12/13 = 0.92$
NC-15	$8/9 = 0.89$
NC-20	$6/7 = 0.86$
NC-24	$5/6 = 0.83$
NC-30	$4/5 = 0.80$
NC-40	$3/4 = 0.75$
NC-60	$2/3 = 0.67$
NC-120	$1/2 = 0.50$

8.2 Performance Metrics

This section describes the four performance metrics measured in our experimentations.

8.2.1 Iperf loss percentage

The first metric is the loss percentage reported by Iperf. The loss percentage is the percentage of the number of packets lost over the total number of packets sent by Iperf (at the application layer) over the duration of the experiment.

8.2.2 Iperf throughput, lost bandwidth and extra bandwidth

The second metric is the throughput reported by Iperf. The throughput is the number of packets successfully received by Iperf over the duration of the experiment (at the application layer). Two related values are the lost bandwidth and the extra bandwidth. The lost bandwidth is calculated by subtracting the throughput from the offered load. The extra bandwidth is the additional bandwidth used beyond the offered capacity for the propose of redundancy. For Baseline, the extra bandwidth is 0. For HARQ and HARQ-ARQ, since we do not know the extra bandwidth used, we assume the best case scenario where the extra bandwidth is also 0. For NC, we simply approximate the extra bandwidth as

$$\frac{m}{n} \times o,$$

where m is the number of redundancy packets per round, n is the preferred number of segments and o is the offered load. Note that for the exact calculation, the calculated number of segments n_s should be used instead of n and the actual extra bandwidth should also include the NC header overhead.

8.2.3 Iperf Throughput to Loss plus Extra Ratio (TLER)

The third metric is the Iperf Throughput to Loss plus Extra Ratio (TLER). TLER is calculated as follows.

$$TLER \equiv \frac{T}{L+E},$$

where T is the throughput, L is the lost bandwidth and E is the extra bandwidth. An efficient scheme should give high throughput while keeping low lost and extra bandwidth. Thus, TLER is a measure of an efficiency of a scheme.

8.2.4 UFTP file transfer delay

The fourth and last metric is the file transfer delay reported by UFTP. In UFTP, a file is divided and packetized into UDP packets of a specified length [11]. The transmitter first sends the packets; the receiver receives them and responds with NACKs for missing packets. The transmitter then resends the missing packets. The file transfer is completed when the transmitter receives zero NACKs from the receiver.

8.3 Results

In this section, we first present separately the results of 4 fixed MCSs and power levels. For each one, in addition to the loss percentage, throughput, TLER and file transfer delay, we report the Carrier to Interference plus Noise Ratio (CINR), Received Signal Strength Indication (RSSI) and Average Tx Power measured at the SS. After this, we summarize the results.

8.3.1 64 QAM CTC 1/2 at 13 dBm

The MCS and power level considered in this section is 64 QAM CTC 1/2 at 13 dBm. The measured CINR, RSSI and Average Tx Power at the SS are shown in Table 8.8.

CINR	13 dB
RSSI	-76 dBm
Average Tx Power	-63 dBm

Table 8.8: 64 QAM CTC 1/2 at 13 dBm. Shows Carrier to Interference plus Noise Ratio (CINR), Received Signal Strength Indication (RSSI) and Average Tx Power measured at the SS.

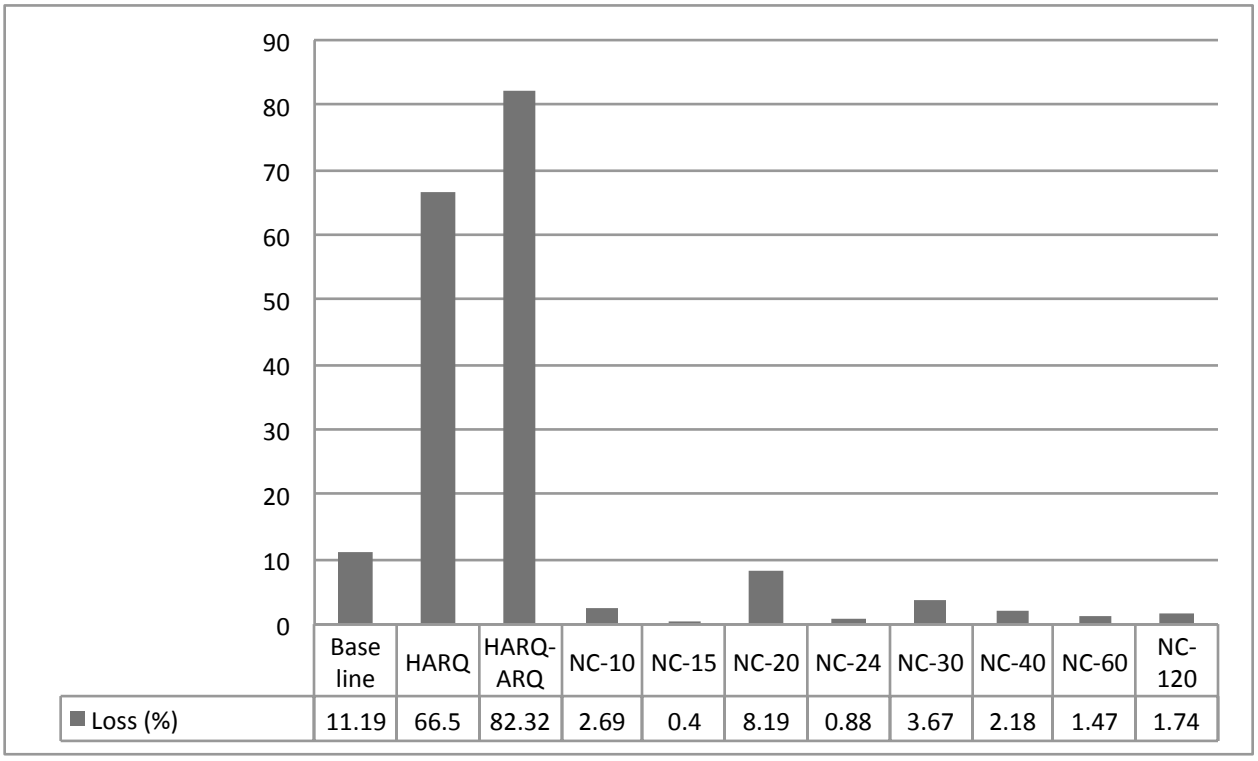
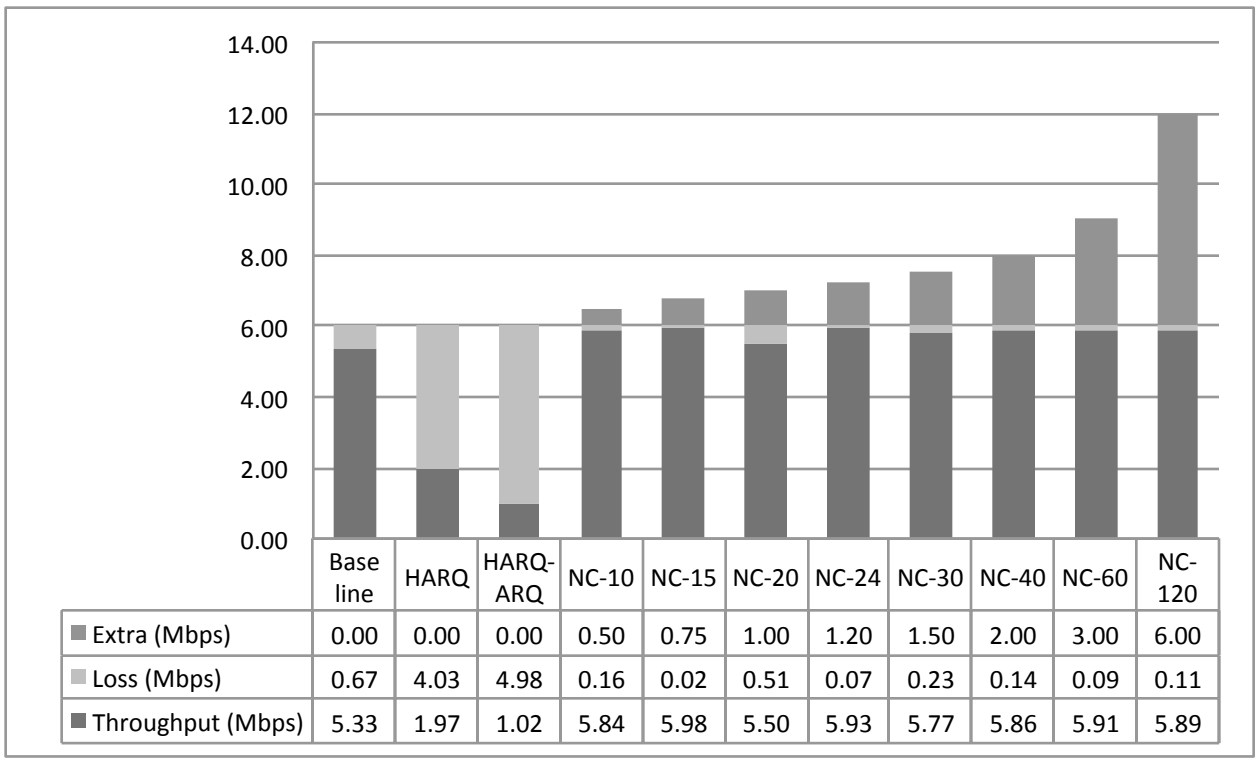


Figure 8-2: 64 QAM CTC 1/2 at 13 dBm. Shows 6 Mbps offered load downlink Iperf loss percentage.

Figure 8-2 shows the loss percentage. Notice that all the NC configurations reduce the loss percentage while HARQ and HARQ-ARQ increase it. NC-15 is the best in terms of the loss percentage.

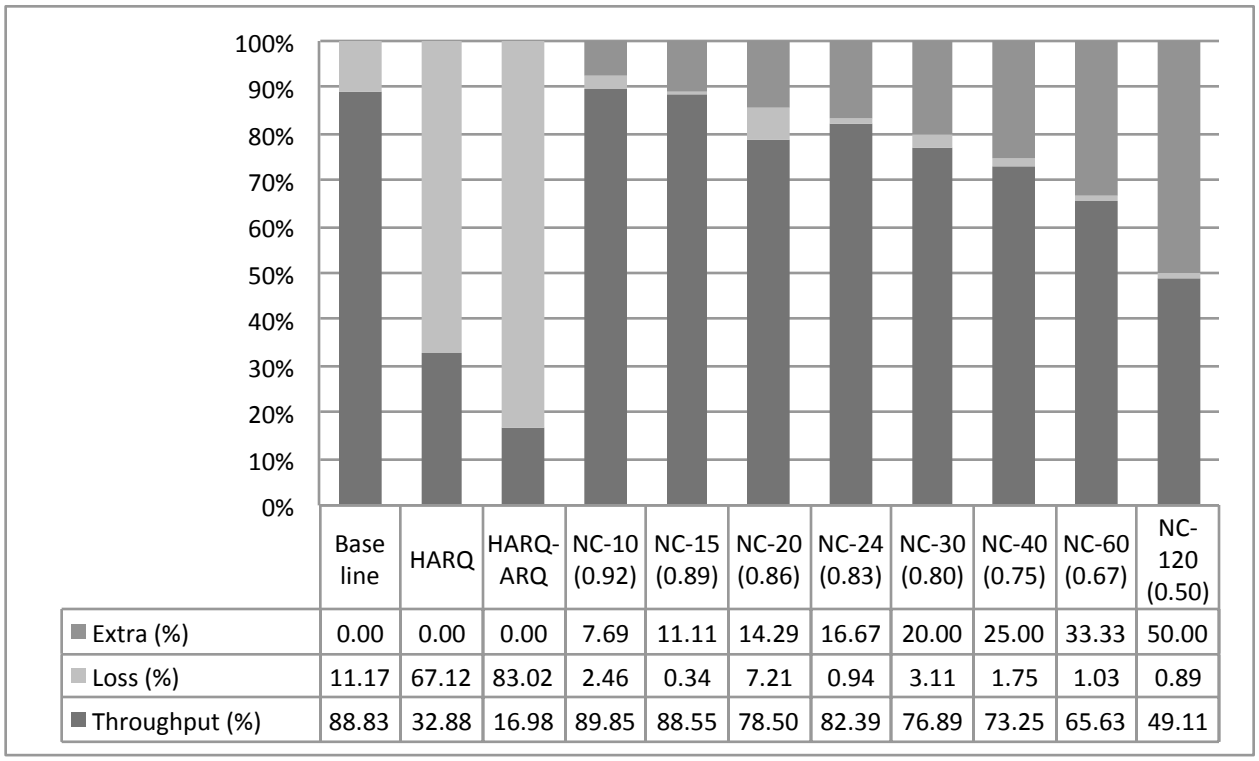


	Base line	HARQ	HARQ-ARQ	NC-10	NC-15	NC-20	NC-24	NC-30	NC-40	NC-60	NC-120
Extra (Mbps)	0.00	0.00	0.00	0.50	0.75	1.00	1.20	1.50	2.00	3.00	6.00
Loss (Mbps)	0.67	4.03	4.98	0.16	0.02	0.51	0.07	0.23	0.14	0.09	0.11
Throughput (Mbps)	5.33	1.97	1.02	5.84	5.98	5.50	5.93	5.77	5.86	5.91	5.89

Figure 8-3: 64 QAM CTC 1/2 at 13 dBm. Shows 6 Mbps offered load downlink Iperf throughput, lost bandwidth and extra bandwidth.

Figure 8-3 shows the throughput, the lost bandwidth and extra bandwidth. The NC-15 results are the best in terms of the throughput. Figure 8-4 alternatively shows the throughput, lost bandwidth and extra bandwidth on a 100% scale. On one hand, the throughput percentage of NC-15 matches that of Baseline (around 89%) and the loss percentage is reduced to nearly 0%. Thus, m greater than 15 may not be necessary. On the other hand, the throughput percentages of HARQ and HARQ-ARQ are around 33% and 17% and the loss percentages are around 67% and 83% respectively. NC-10 is the best in terms of the throughput percentage. NC-15 has a CR of 0.89, which roughly matches the throughput percentage of Baseline (88.83%); it has the lowest lost bandwidth.

Notice that NC-10 has a higher throughput percentage than that of Baseline while in theory, NC-10 should not have a higher throughput percentage than that of Baseline. This effect may be caused by 1) the experimental variations and 2) the under-approximate extra bandwidth.



	Base line	HARQ	HARQ-ARQ	NC-10 (0.92)	NC-15 (0.89)	NC-20 (0.86)	NC-24 (0.83)	NC-30 (0.80)	NC-40 (0.75)	NC-60 (0.67)	NC-120 (0.50)
Extra (%)	0.00	0.00	0.00	7.69	11.11	14.29	16.67	20.00	25.00	33.33	50.00
Loss (%)	11.17	67.12	83.02	2.46	0.34	7.21	0.94	3.11	1.75	1.03	0.89
Throughput (%)	88.83	32.88	16.98	89.85	88.55	78.50	82.39	76.89	73.25	65.63	49.11

Figure 8-4: 64 QAM CTC 1/2 at 13 dBm. Shows 6 Mbps offered load downlink Iperf throughput, lost bandwidth and extra bandwidth on a 100% scale. For each NC configuration, its Code Rate (CR) is noted in parentheses.

Figure 8-5 shows the TLER. Notice that both HARQ and HARQ-ARQ have very low TLERs. The TLERs for the NC configurations decrease as m increases. All the NC configurations have significantly higher TLERs than those of HARQ and HARQ-ARQ. NC-10 has the highest TLER, which is higher than that of Baseline; it is considered the most efficient configuration.

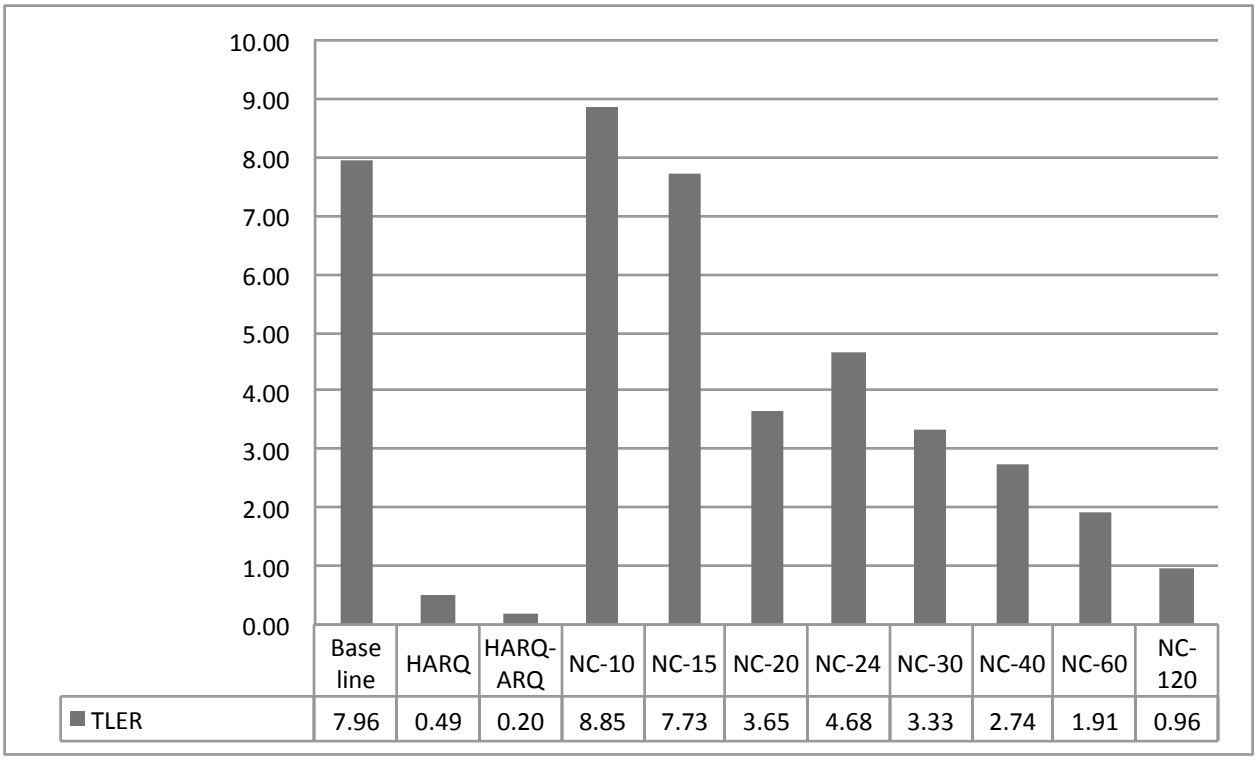


Figure 8-5: 64 QAM CTC 1/2 at 13 dBm. Shows 6 Mbps offered load downlink Iperf Throughput to Loss plus Extra Ratio (TLER).

Figure 8-6 shows the file transfer delay. HARQ and HARQ-ARQ have very high file transfer delays. That of HARQ-ARQ is around twice as much as that of HARQ and that of HARQ is around twice as much as that of Baseline. The delays of Baseline and the NC configurations are comparable. NC-40 and NC-60 give the best performance in terms of the delay, i.e., a 12% reduction from that of Baseline.

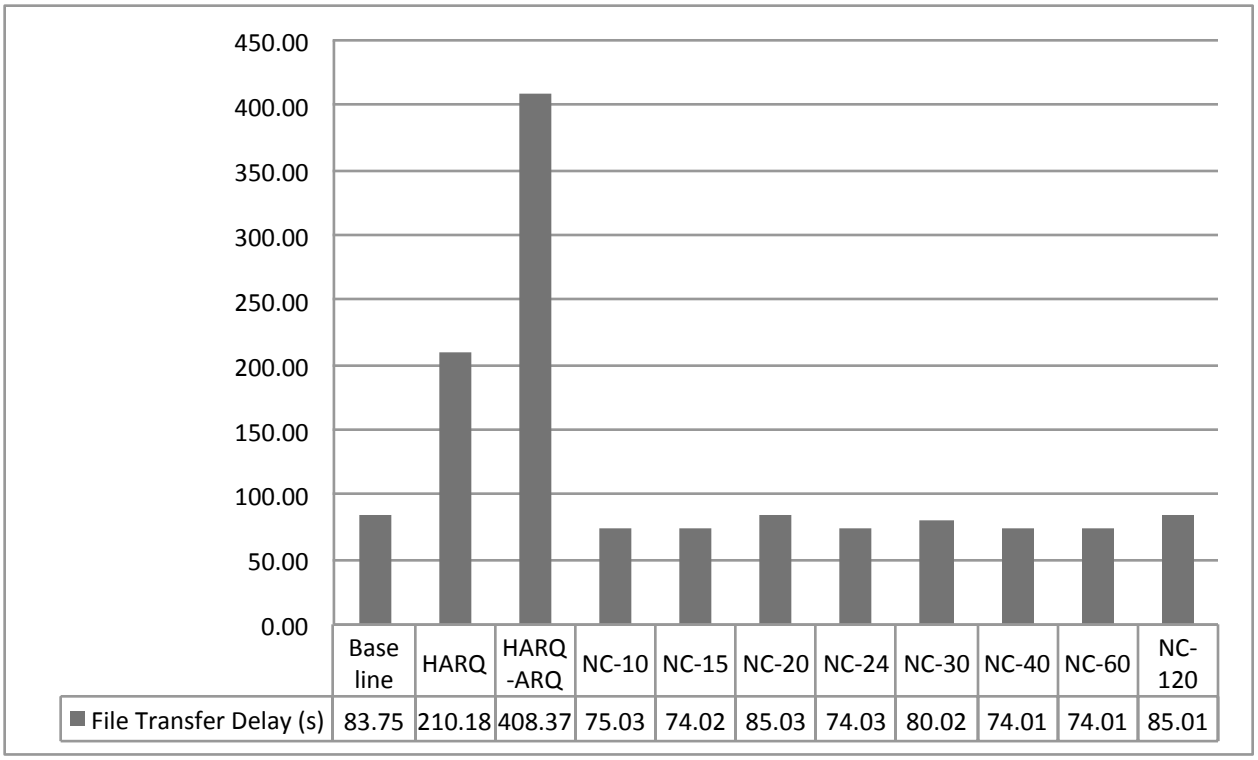


Figure 8-6: 64 QAM CTC 1/2 at 13 dBm. Shows 6 Mbps offered load downlink UFTP 50 MB file transfer delay.

8.3.2 64 QAM CTC 2/3 at 17 dBm

The MCS and power level considered in this section is 64 QAM CTC 2/3 at 17 dBm. The measured CINR, RSSI and Average Tx Power at the SS are shown in Table 8.9.

CINR	17 dB
RSSI	-76 dBm
Average Tx Power	-63 dBm

Table 8.9: 64 QAM CTC 2/3 at 17 dBm. Shows Carrier to Interference plus Noise Ratio (CINR), Received Signal Strength Indication (RSSI) and Average Tx Power measured at the SS.

Figure 8-7 shows the loss percentage. Notice that the loss percentages of all NC configurations except NC-10 are nearly 0%, while those of HARQ and HARQ-ARQ are around 50-80%, a much higher percentage than that of Baseline (around 10%).

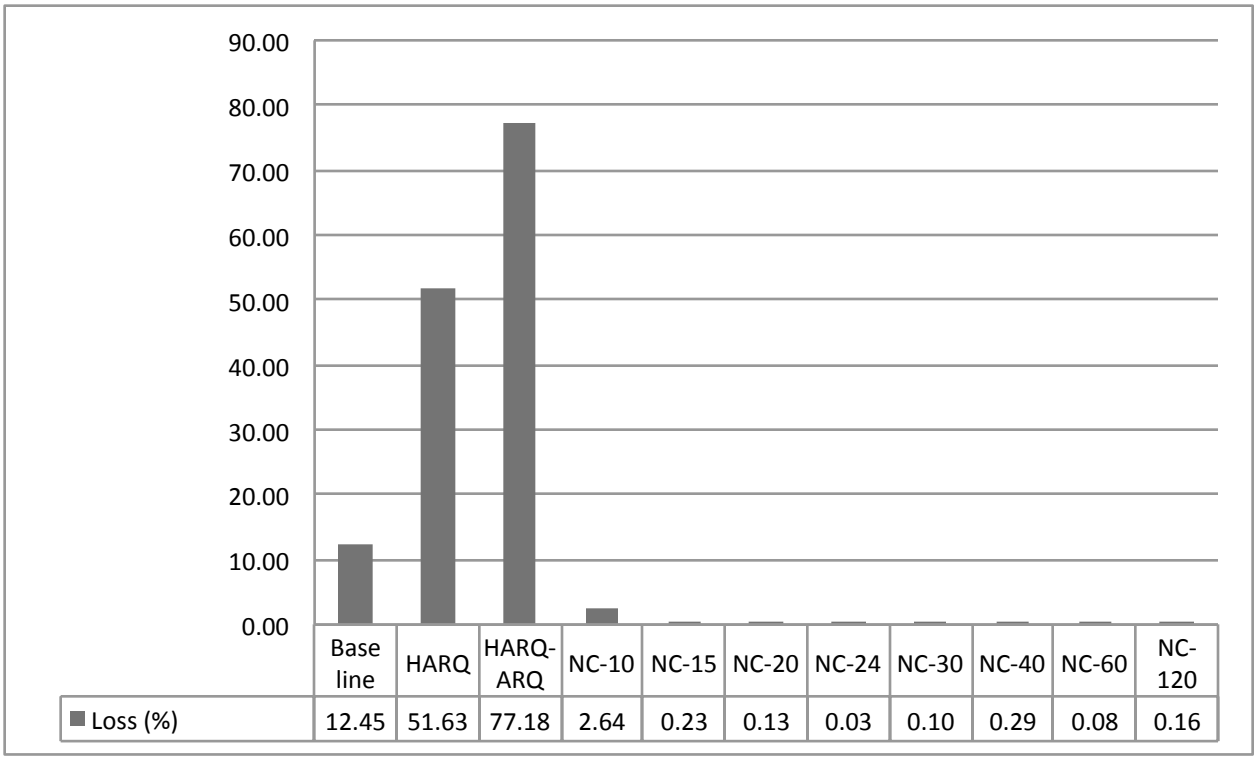
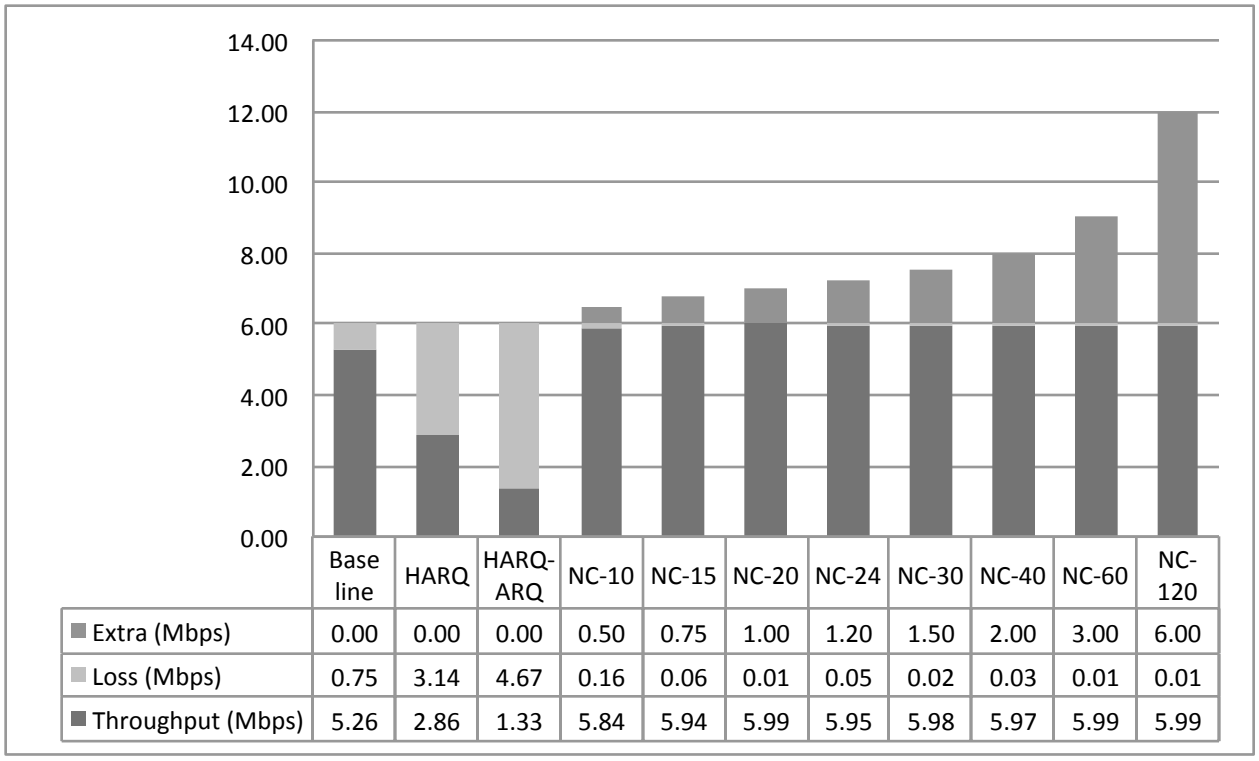


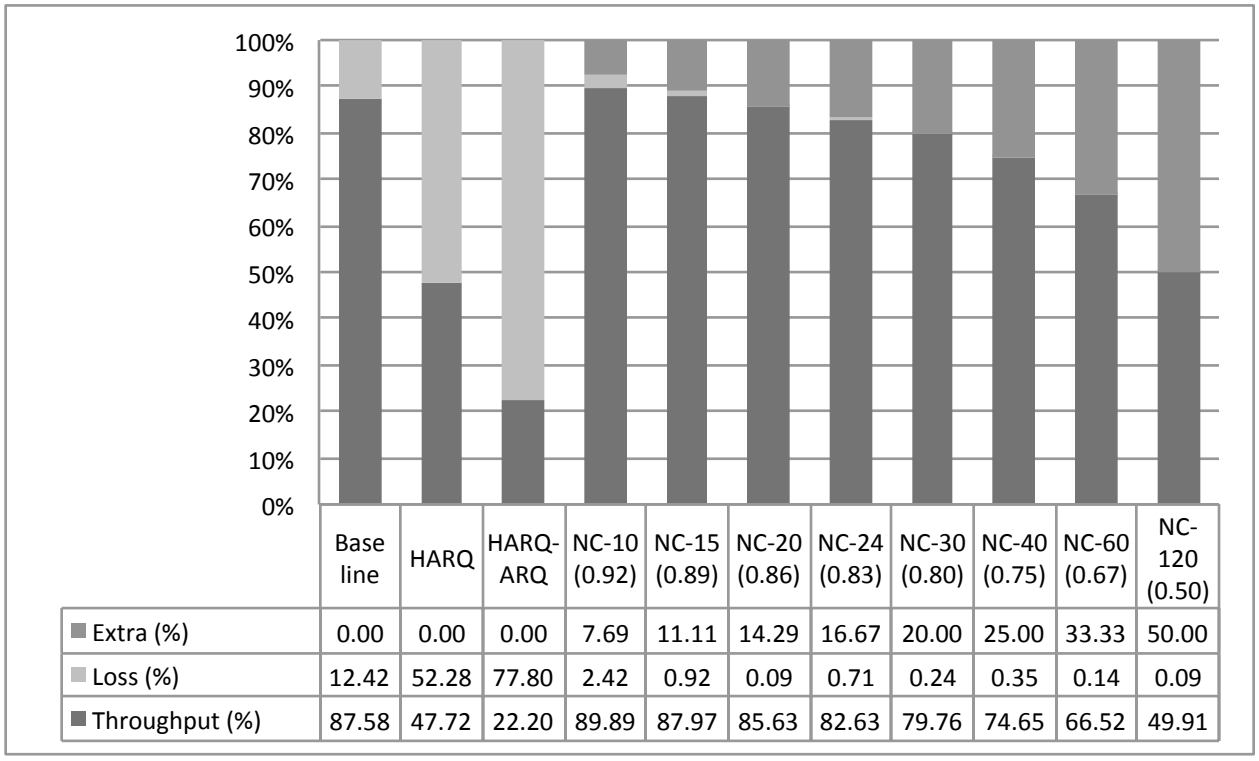
Figure 8-7: 64 QAM CTC 2/3 at 17 dBm. Shows 6 Mbps offered load downlink Iperf loss percentage.

Figure 8-8 shows the throughput, lost bandwidth and extra bandwidth. Figure 8-9 alternatively shows the throughput, lost bandwidth and extra bandwidth on a 100% scale. Notice that on one hand, the throughput percentage of NC-15 matches that of Baseline (around 88%) and the loss percentage is reduced to nearly 0%. Thus, m greater than 15 may not be necessary. The throughput percentages of HARQ and HARQ-ARQ, on the other hand, are around 50% and 20% and the loss percentages are around 50% and 80% respectively. NC-20 has a CR of 0.86, which is slightly below the throughput percentage of Baseline (87.58%); it is one of the configurations that has the lowest lost bandwidth.



	Base line	HARQ	HARQ-ARQ	NC-10	NC-15	NC-20	NC-24	NC-30	NC-40	NC-60	NC-120
Extra (Mbps)	0.00	0.00	0.00	0.50	0.75	1.00	1.20	1.50	2.00	3.00	6.00
Loss (Mbps)	0.75	3.14	4.67	0.16	0.06	0.01	0.05	0.02	0.03	0.01	0.01
Throughput (Mbps)	5.26	2.86	1.33	5.84	5.94	5.99	5.95	5.98	5.97	5.99	5.99

Figure 8-8: 64 QAM CTC 2/3 at 17 dBm. Shows 6 Mbps offered load downlink Iperf throughput, lost bandwidth and extra bandwidth.



	Base line	HARQ	HARQ-ARQ	NC-10 (0.92)	NC-15 (0.89)	NC-20 (0.86)	NC-24 (0.83)	NC-30 (0.80)	NC-40 (0.75)	NC-60 (0.67)	NC-120 (0.50)
Extra (%)	0.00	0.00	0.00	7.69	11.11	14.29	16.67	20.00	25.00	33.33	50.00
Loss (%)	12.42	52.28	77.80	2.42	0.92	0.09	0.71	0.24	0.35	0.14	0.09
Throughput (%)	87.58	47.72	22.20	89.89	87.97	85.63	82.63	79.76	74.65	66.52	49.91

Figure 8-9: 64 QAM CTC 2/3 at 17 dBm. Shows 6 Mbps offered load downlink Iperf throughput, lost bandwidth and extra bandwidth on a 100% scale. For each NC configuration, its Code Rate (CR) is noted in parentheses.

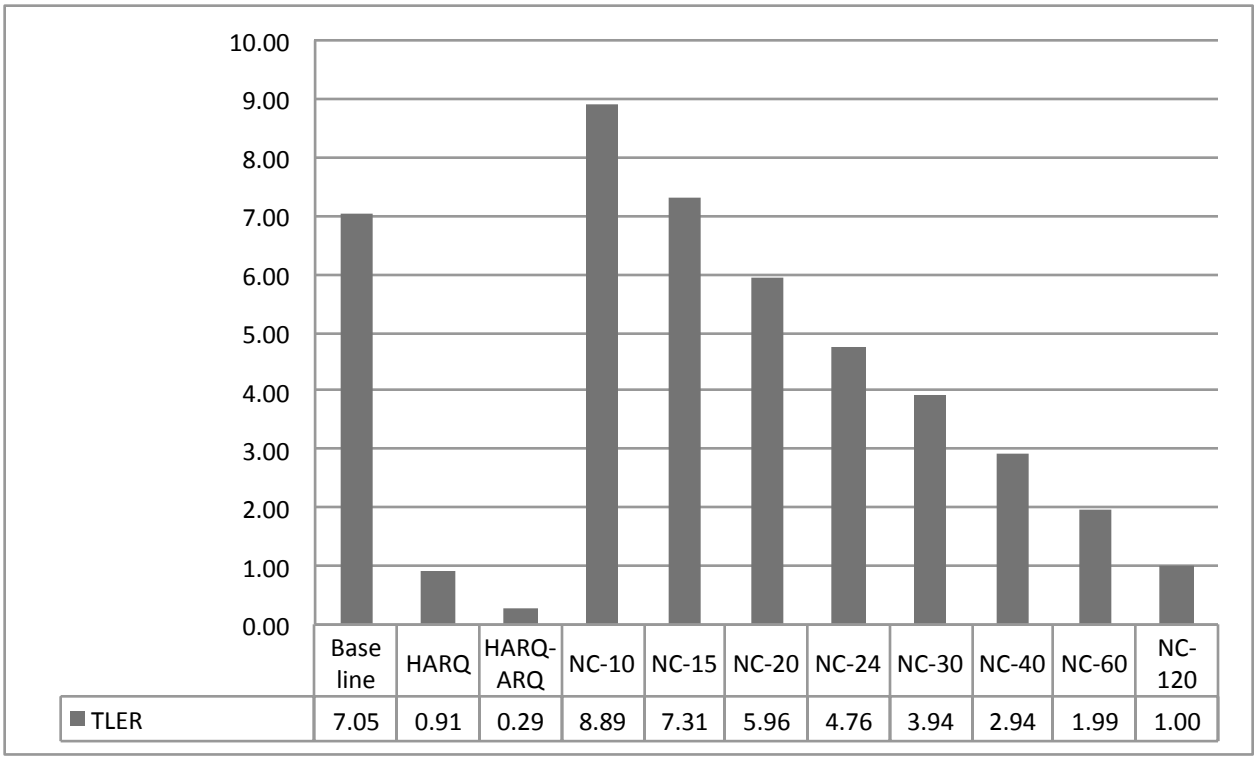


Figure 8-10: 64 QAM CTC 2/3 at 17 dBm. Shows 6 Mbps offered load downlink Iperf Throughput to Loss plus Extra Ratio (TLER).

Figure 8-10 shows the TLER. Notice that both HARQ and HARQ-ARQ have very low TLERs. The TLERs for the NC configurations decrease as m increases. All the NC configurations have significantly higher TLERs than those of HARQ and HARQ-ARQ. NC-10 has the highest TLER, which is higher than that of Baseline; it is considered the most efficient configuration. NC-15 also has a higher TLER than that of Baseline; however, NC-15 is not as efficient as NC-10.

Figure 8-11 shows the file transfer delay. HARQ and HARQ-ARQ have very high file transfer delays. The delays of Baseline and the NC configurations are comparable. NC-60 gives the best performance in terms of the delay, i.e., an 11% reduction from that of Baseline.

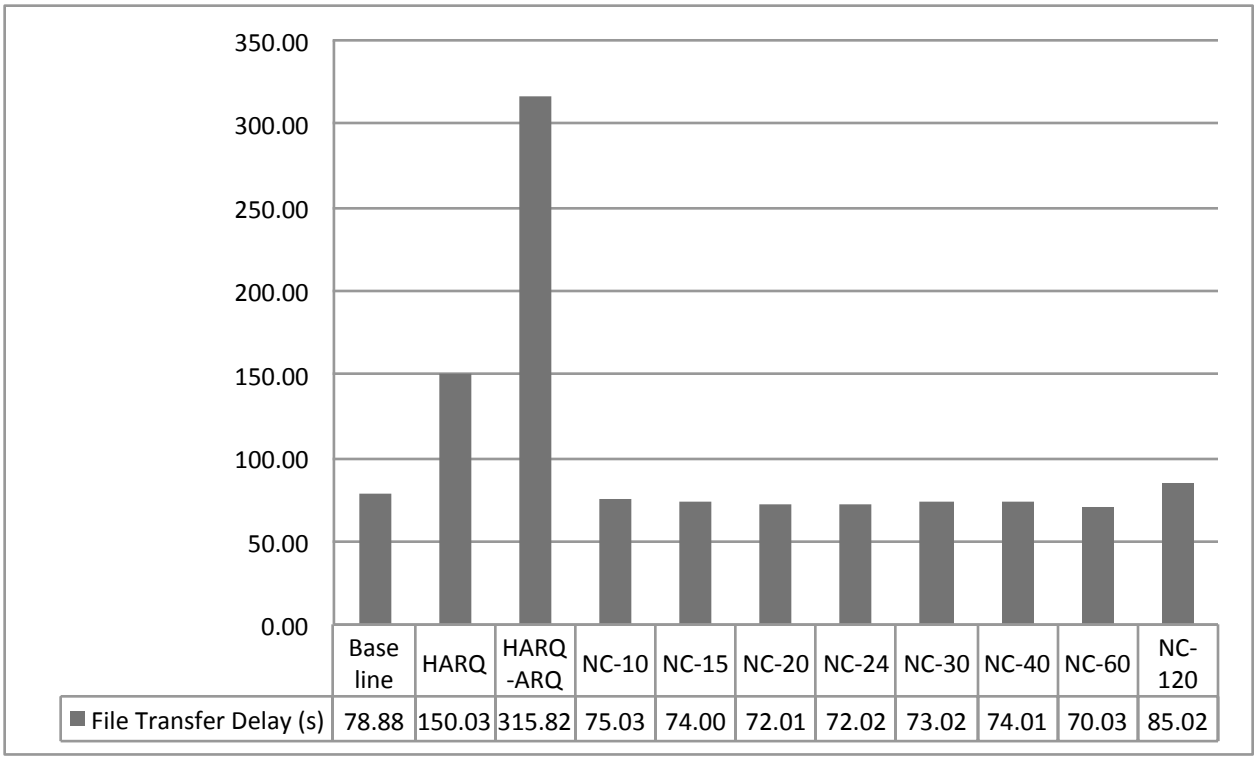


Figure 8-11: 64 QAM CTC 2/3 at 17 dBm. Shows 6 Mbps offered load downlink UFTP 50 MB file transfer delay.

8.3.3 64 QAM CTC 3/4 at 18 dBm

The MCS and power level considered in this section is 64 QAM CTC 3/4 at 18 dBm. The measured CINR, RSSI and Average Tx Power at the SS are shown in Table 8.10.

CINR	18 dB
RSSI	-75 dBm
Average Tx Power	-63 dBm

Table 8.10: 64 QAM CTC 3/4 at 18 dBm. Shows Carrier to Interference plus Noise Ratio (CINR), Received Signal Strength Indication (RSSI) and Average Tx Power measured at the SS.

Figure 8-12 shows the loss percentage. Notice that all the NC configurations reduce the loss percentage, while HARQ and HARQ-ARQ increase it. In terms of the loss percentage, while NC-30 is the best, it is comparable to NC-15, NC-20, NC-40, NC-60 and NC-120.

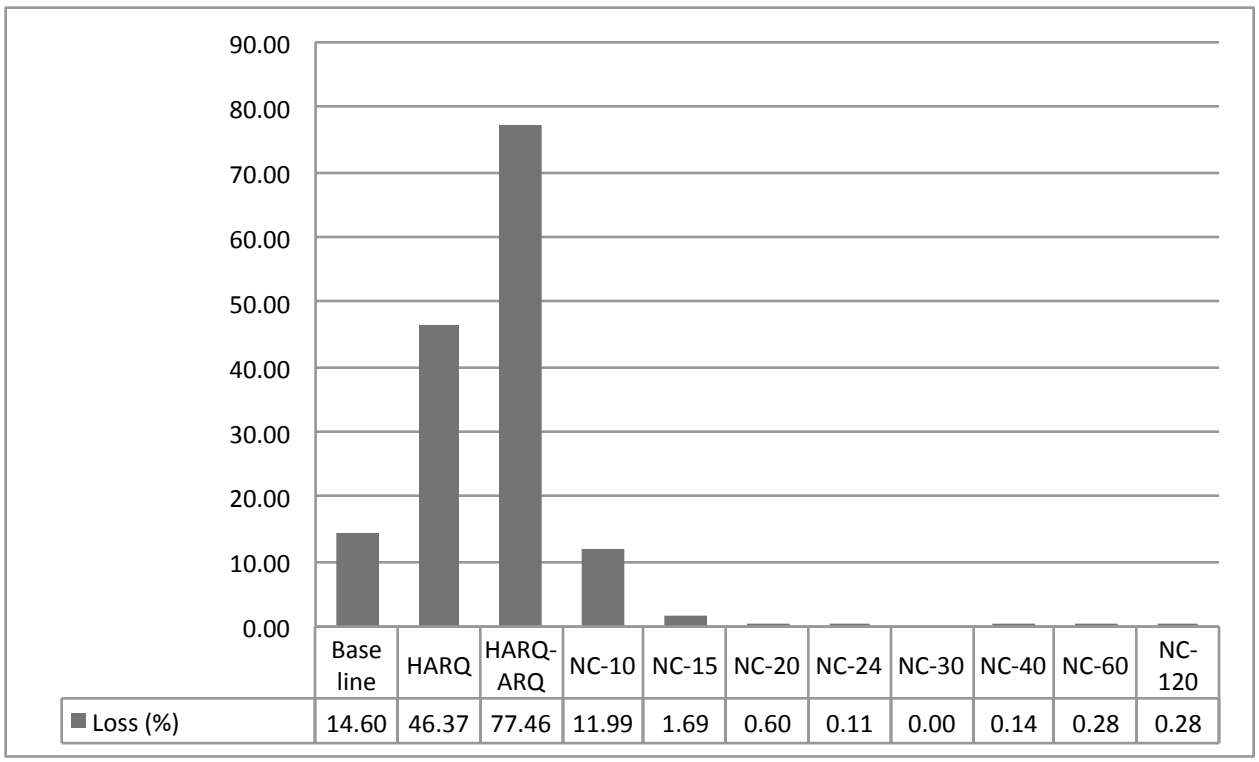
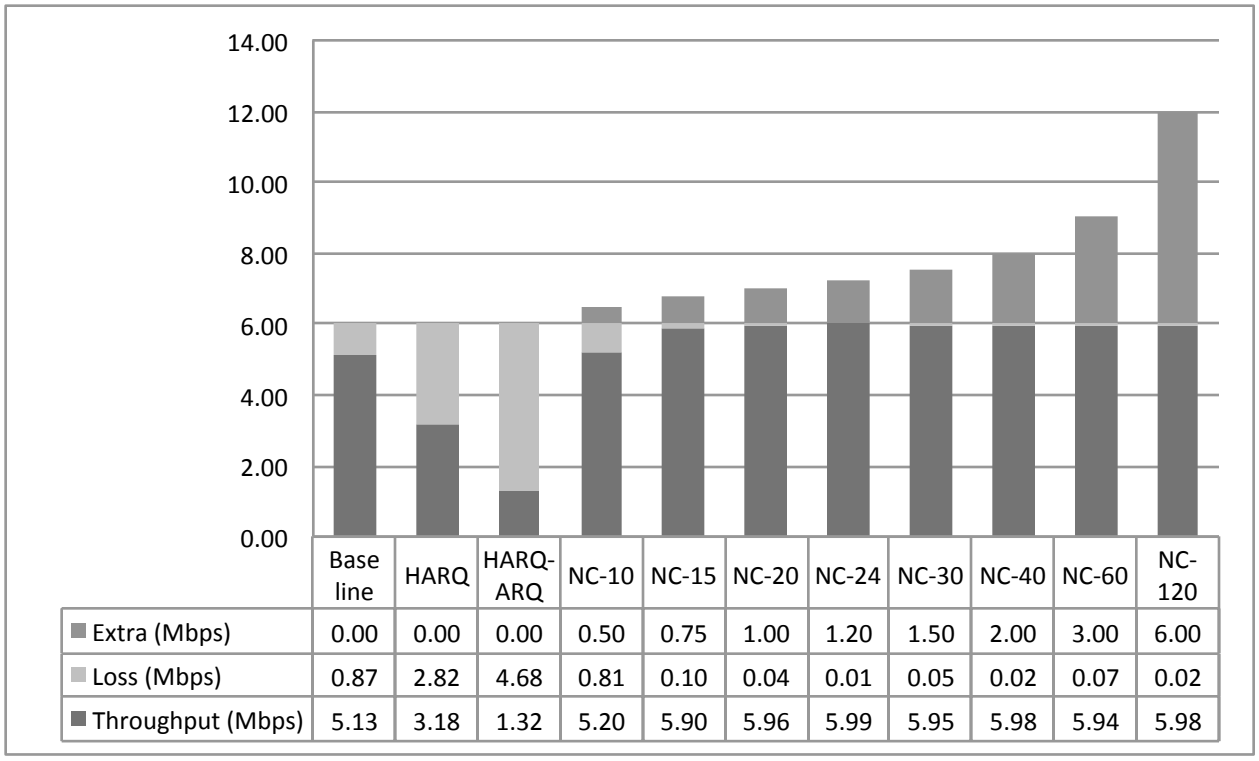


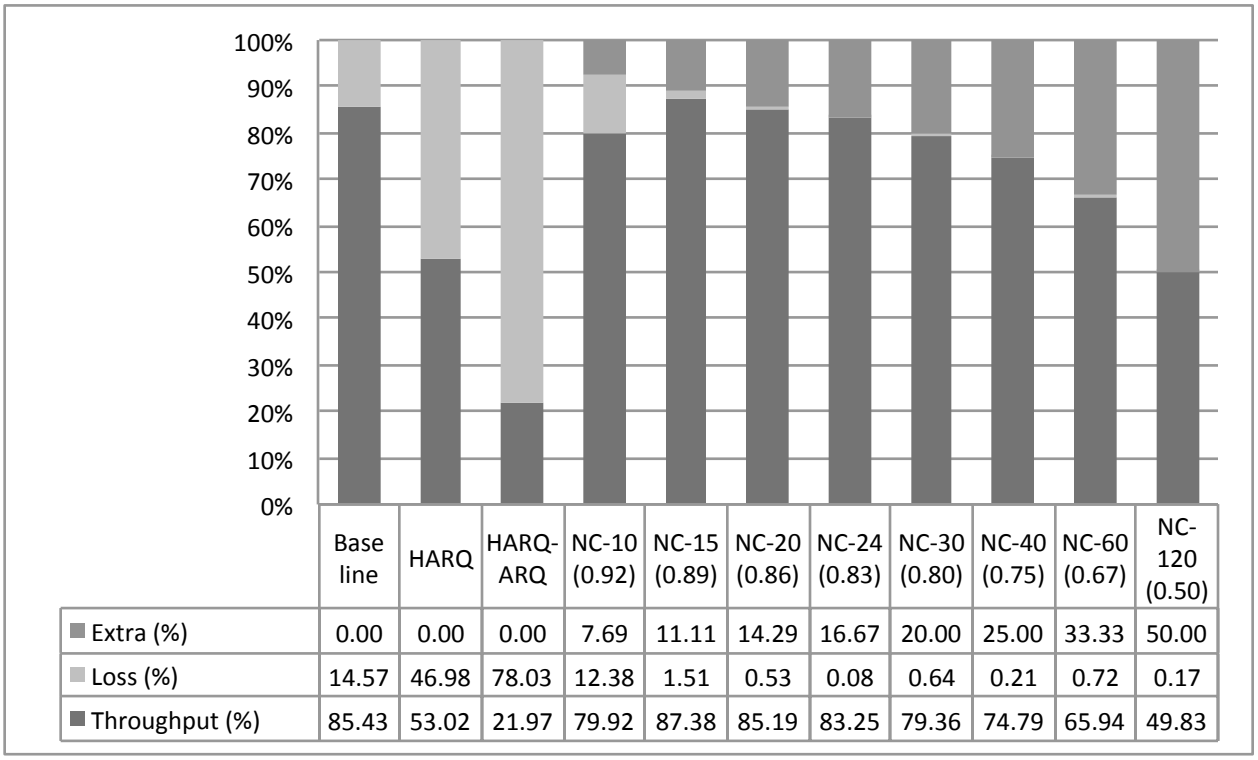
Figure 8-12: 64 QAM CTC 3/4 at 18 dBm. Shows 6 Mbps offered load downlink Iperf loss percentage.

Figure 8-13 shows the throughput, lost bandwidth and extra bandwidth. Figure 8-14 alternatively shows the throughput, lost bandwidth and extra bandwidth on a 100% scale. Notice that on one hand, the throughput percentage of NC-20 matches that of Baseline (around 85%) and the loss percentage is reduced to nearly 0%. Thus, m greater than 20 may not be necessary. The throughput percentages of HARQ and HARQ-ARQ, on the other hand, are around 52% and 21% and the loss percentages are around 48% and 79% respectively. NC-24 has a CR of 0.83, which slightly below the throughput percentage of Baseline (85.43%); it has the lowest lost bandwidth.



	Base line	HARQ	HARQ-ARQ	NC-10	NC-15	NC-20	NC-24	NC-30	NC-40	NC-60	NC-120
Extra (Mbps)	0.00	0.00	0.00	0.50	0.75	1.00	1.20	1.50	2.00	3.00	6.00
Loss (Mbps)	0.87	2.82	4.68	0.81	0.10	0.04	0.01	0.05	0.02	0.07	0.02
Throughput (Mbps)	5.13	3.18	1.32	5.20	5.90	5.96	5.99	5.95	5.98	5.94	5.98

Figure 8-13: 64 QAM CTC 3/4 at 18 dBm. Shows 6 Mbps offered load downlink Iperf throughput, lost bandwidth and extra bandwidth.



	Base line	HARQ	HARQ-ARQ	NC-10 (0.92)	NC-15 (0.89)	NC-20 (0.86)	NC-24 (0.83)	NC-30 (0.80)	NC-40 (0.75)	NC-60 (0.67)	NC-120 (0.50)
Extra (%)	0.00	0.00	0.00	7.69	11.11	14.29	16.67	20.00	25.00	33.33	50.00
Loss (%)	14.57	46.98	78.03	12.38	1.51	0.53	0.08	0.64	0.21	0.72	0.17
Throughput (%)	85.43	53.02	21.97	79.92	87.38	85.19	83.25	79.36	74.79	65.94	49.83

Figure 8-14: 64 QAM CTC 3/4 at 18 dBm. Shows 6 Mbps offered load downlink Iperf throughput, lost bandwidth and extra bandwidth on a 100% scale. For each NC configuration, its Code Rate (CR) is noted in parentheses.

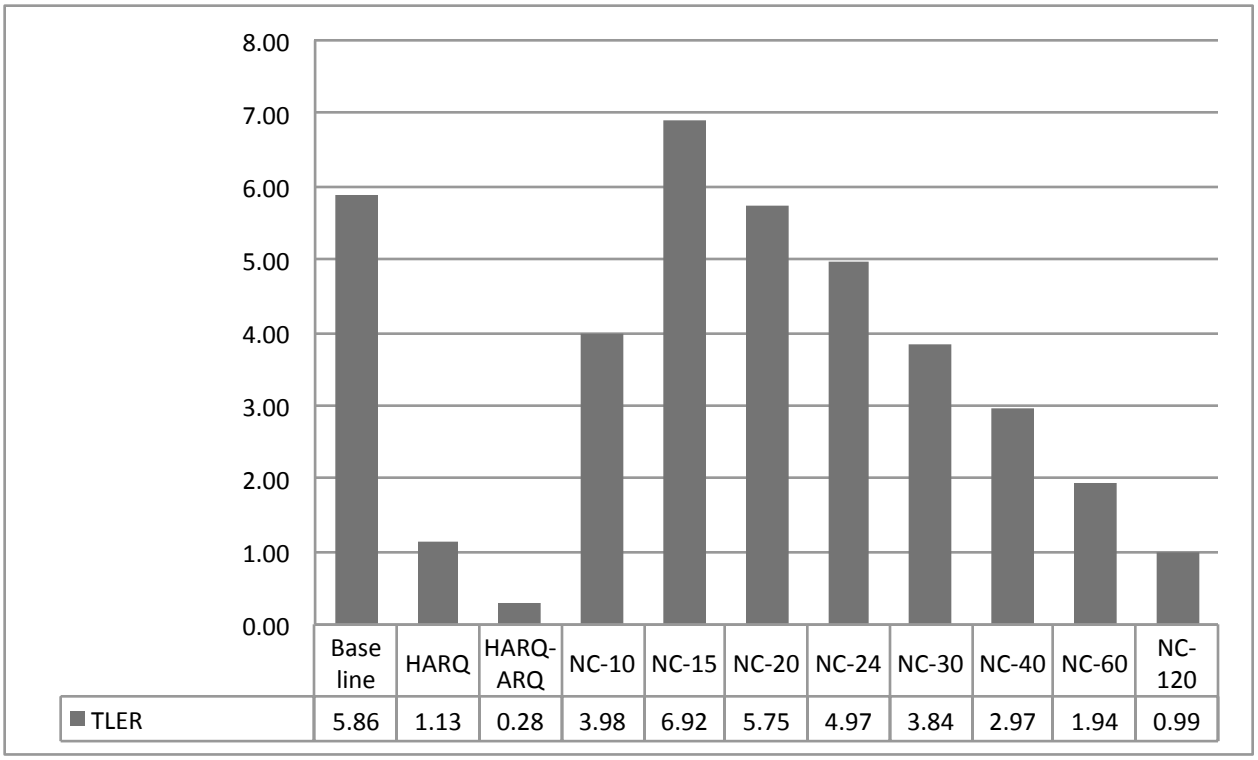


Figure 8-15: 64 QAM CTC 3/4 at 18 dBm. Shows 6 Mbps offered load downlink Iperf Throughput to Loss plus Extra Ratio (TLER).

Figure 8-15 shows the TLER. Notice that both HARQ and HARQ-ARQ have very low TLERs. The TLER for the NC configurations decreases as m increases with the exception of NC-10. All the NC configurations have significantly higher TLERs than those of HARQ and HARQ-ARQ. NC-15 has the highest TLER, which is higher than that of Baseline; it is considered the most efficient configuration.

Figure 8-16 shows the file transfer delay. Notice that HARQ-ARQ has a very high file transfer delay. NC-30 gives the best performance in terms of the delay, i.e., a 39% reduction from that of Baseline.

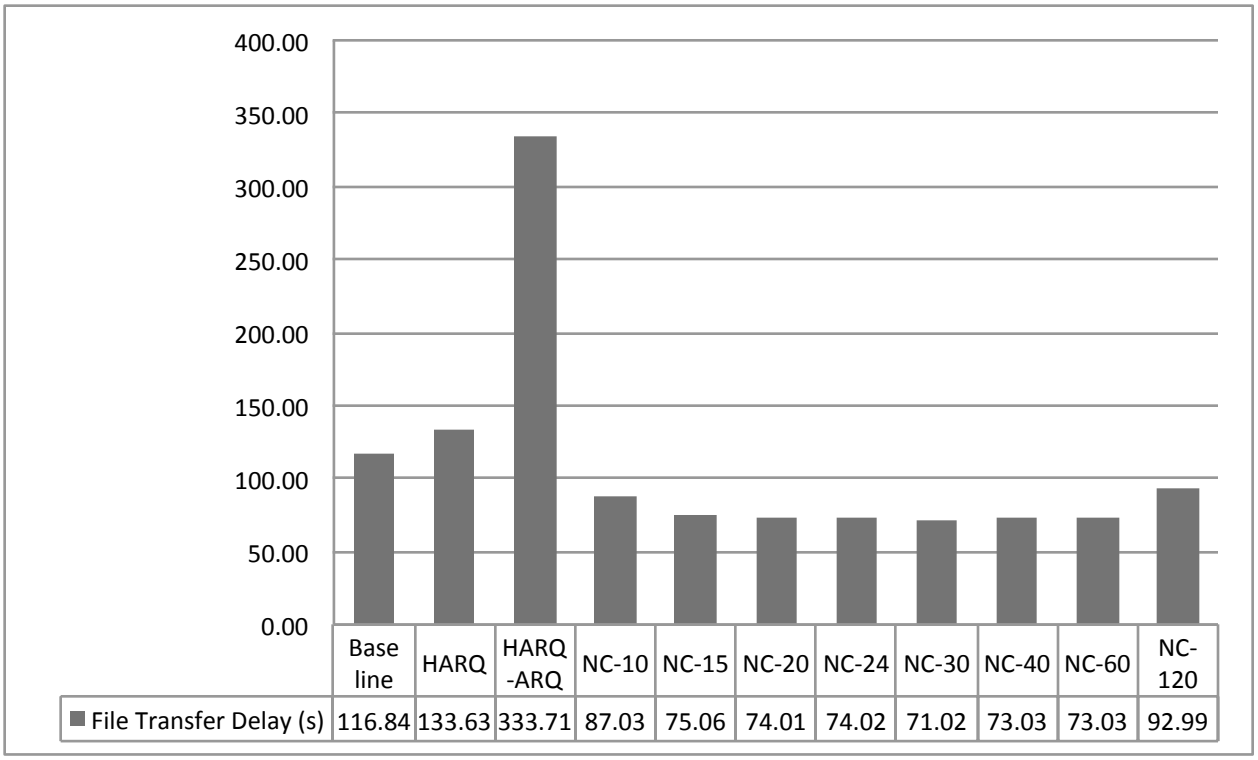


Figure 8-16: 64 QAM CTC 3/4 at 18 dBm. Shows 6 Mbps offered load downlink UFTP 50 MB file transfer delay.

8.3.4 64 QAM CTC 5/6 at 20 dBm

The MCS and power level considered in this section is 64 QAM CTC 5/6 at 20 dBm. The measured CINR, RSSI and Average Tx Power at the SS are shown in Table 8.11.

CINR	18 dB
RSSI	-73 dBm
Average Tx Power	-63 dBm

Table 8.11: 64 QAM CTC 5/6 at 20 dBm. Shows Carrier to Interference plus Noise Ratio (CINR), Received Signal Strength Indication (RSSI) and Average Tx Power measured at the SS.

Figure 8-17 shows the loss percentage. Notice that all the NC configurations reduce the loss percentage while HARQ and HARQ-ARQ increase it. However, HARQ-ARQ has a lower loss percentage than that of HARQ. NC-10 has the highest loss percentage while NC-40 has the lowest one.

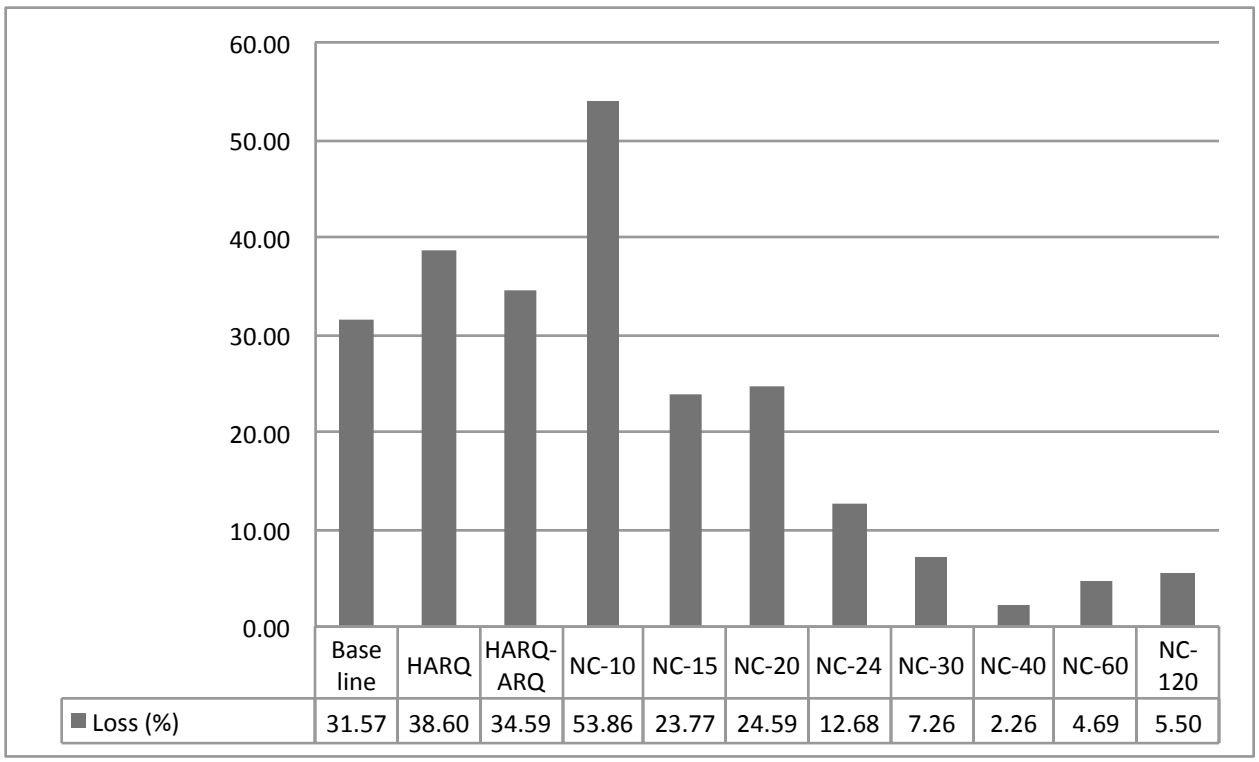
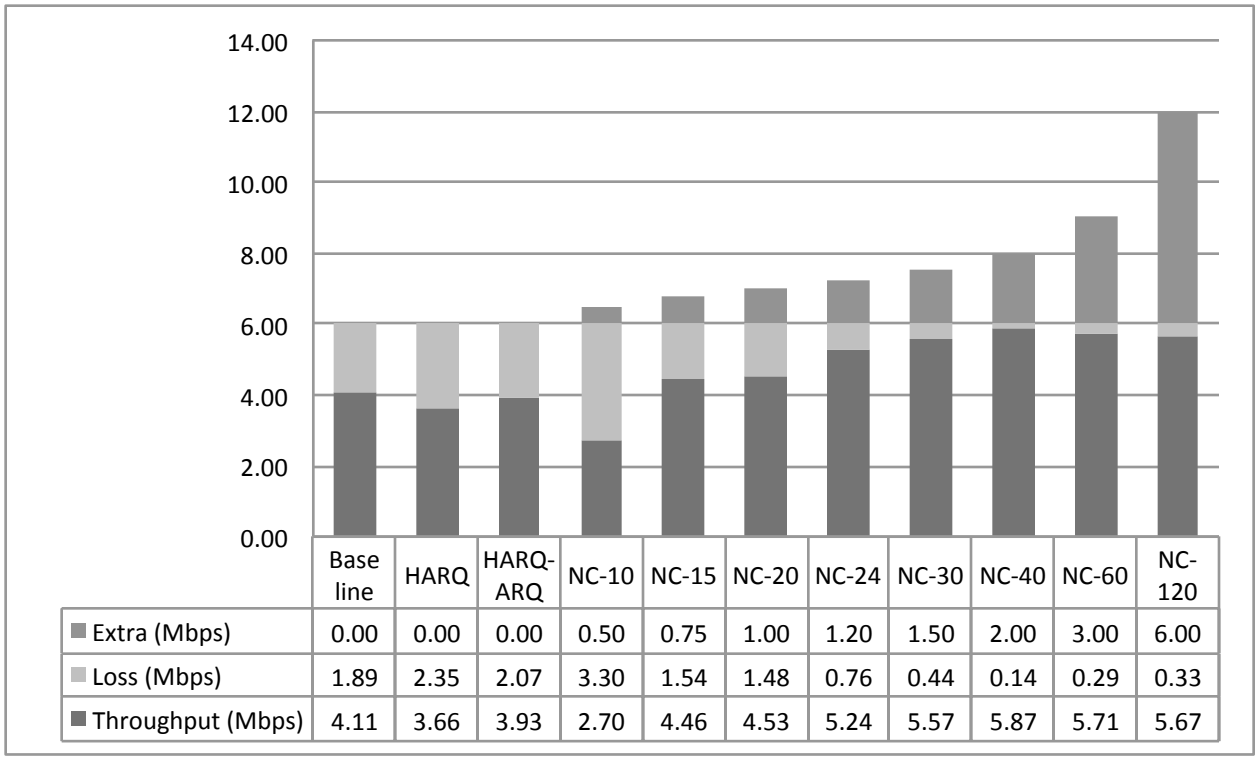


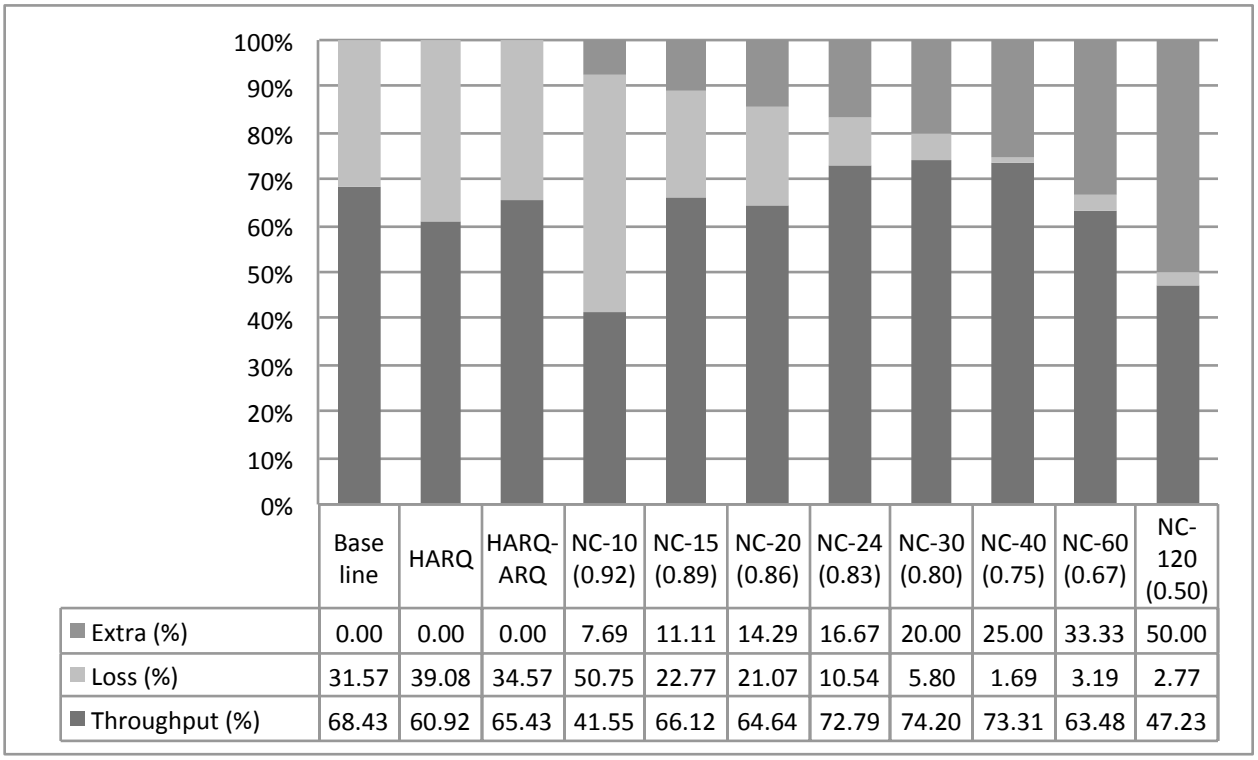
Figure 8-17: 64 QAM CTC 5/6 at 20 dBm. Shows 6 Mbps offered load downlink Iperf loss percentage.

Figure 8-18 shows the throughput, lost bandwidth and extra bandwidth. Figure 8-19 alternatively shows the throughput, lost bandwidth and extra bandwidth on a 100% scale. In terms of the throughput, NC-40 is the best while NC-10 is the worst. The performance among the NC configurations varies. The higher the m, the lower the loss percentage. In terms of the throughput percentage, the performance of HARQ and HARQ-ARQ is comparable to that of Baseline, NC-15, NC-20 and NC-60. NC-40 has a CR of 0.75, which is slightly above the throughput percentage of Baseline (68.43%); it has the lowest lost bandwidth.



	Base line	HARQ	HARQ-ARQ	NC-10	NC-15	NC-20	NC-24	NC-30	NC-40	NC-60	NC-120
Extra (Mbps)	0.00	0.00	0.00	0.50	0.75	1.00	1.20	1.50	2.00	3.00	6.00
Loss (Mbps)	1.89	2.35	2.07	3.30	1.54	1.48	0.76	0.44	0.14	0.29	0.33
Throughput (Mbps)	4.11	3.66	3.93	2.70	4.46	4.53	5.24	5.57	5.87	5.71	5.67

Figure 8-18: 64 QAM CTC 5/6 at 20 dBm. Shows 6 Mbps offered load downlink Iperf throughput, lost bandwidth and extra bandwidth.



	Base line	HARQ	HARQ-ARQ	NC-10 (0.92)	NC-15 (0.89)	NC-20 (0.86)	NC-24 (0.83)	NC-30 (0.80)	NC-40 (0.75)	NC-60 (0.67)	NC-120 (0.50)
Extra (%)	0.00	0.00	0.00	7.69	11.11	14.29	16.67	20.00	25.00	33.33	50.00
Loss (%)	31.57	39.08	34.57	50.75	22.77	21.07	10.54	5.80	1.69	3.19	2.77
Throughput (%)	68.43	60.92	65.43	41.55	66.12	64.64	72.79	74.20	73.31	63.48	47.23

Figure 8-19: 64 QAM CTC 5/6 at 20 dBm. Shows 6 Mbps offered load downlink Iperf throughput, lost bandwidth and extra bandwidth on a 100% scale. For each NC configuration, its Code Rate (CR) is noted in parentheses.

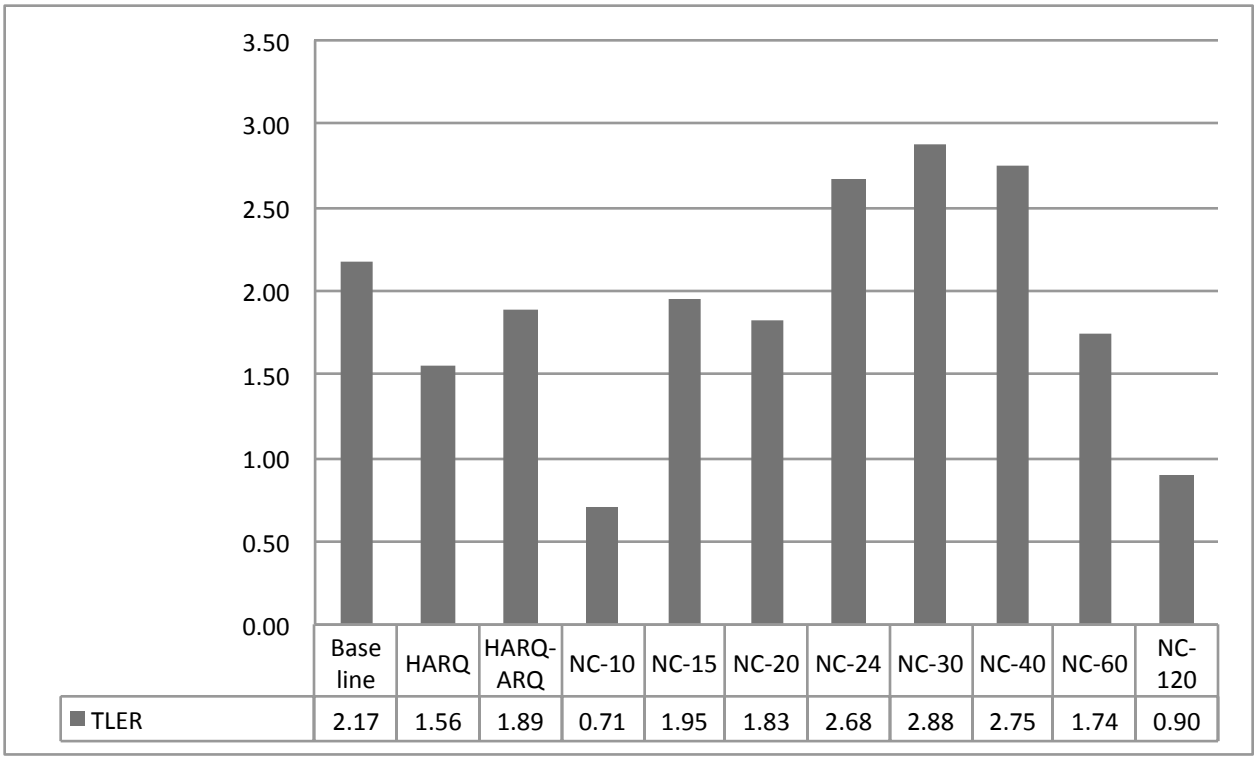


Figure 8-20: 64 QAM CTC 5/6 at 20 dBm. Shows 6 Mbps offered load downlink Iperf Throughput to Loss plus Extra Ratio (TLER).

Figure 8-20 shows the TLER. Notice that the TLERs of HARQ and HARQ-ARQ are on par with and sometimes better than some of the NC configurations. The TLERs of high-m NC configurations, i.e., NC-30, NC-40, NC-60 and NC-120, decreases as m increases. NC-30 has the highest TLER, which is higher than that of Baseline; it is considered the most efficient configuration.

Figure 8-21 shows the file transfer delay. NC-40 is the best in terms of the delay, i.e., a 46% reduction from that of Baseline.

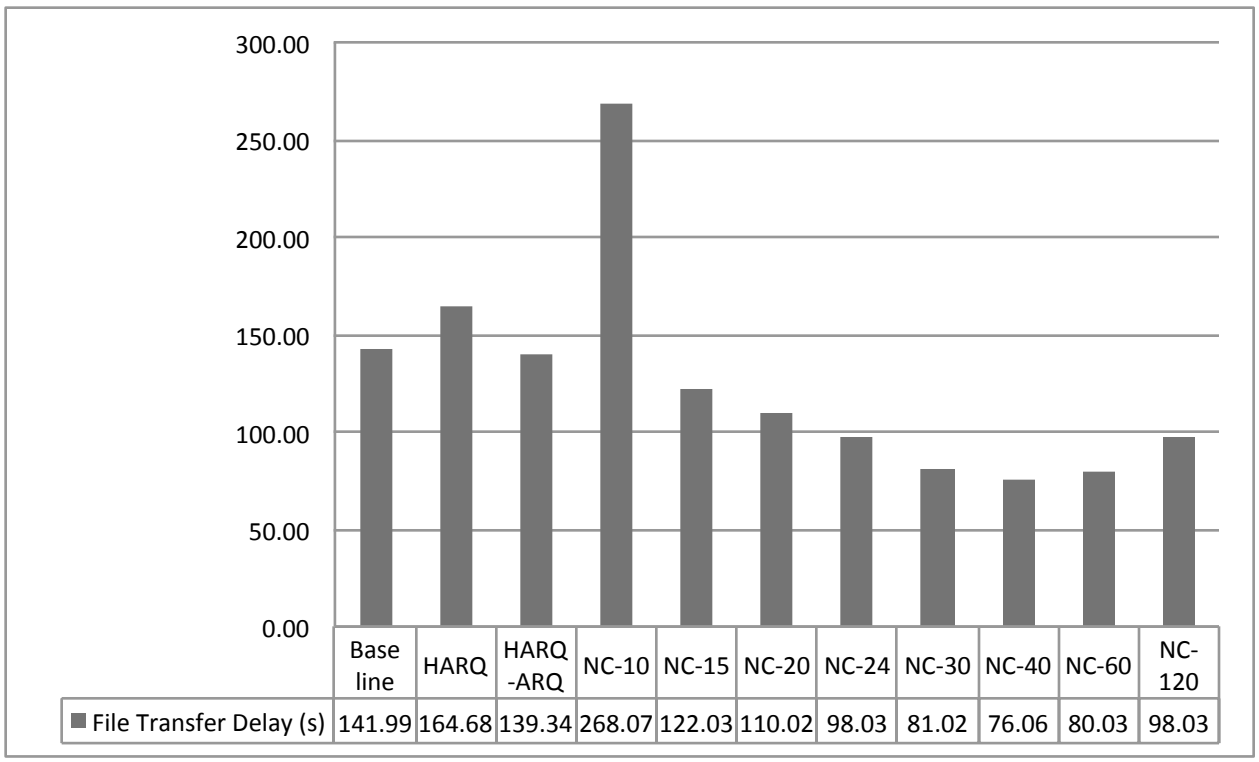
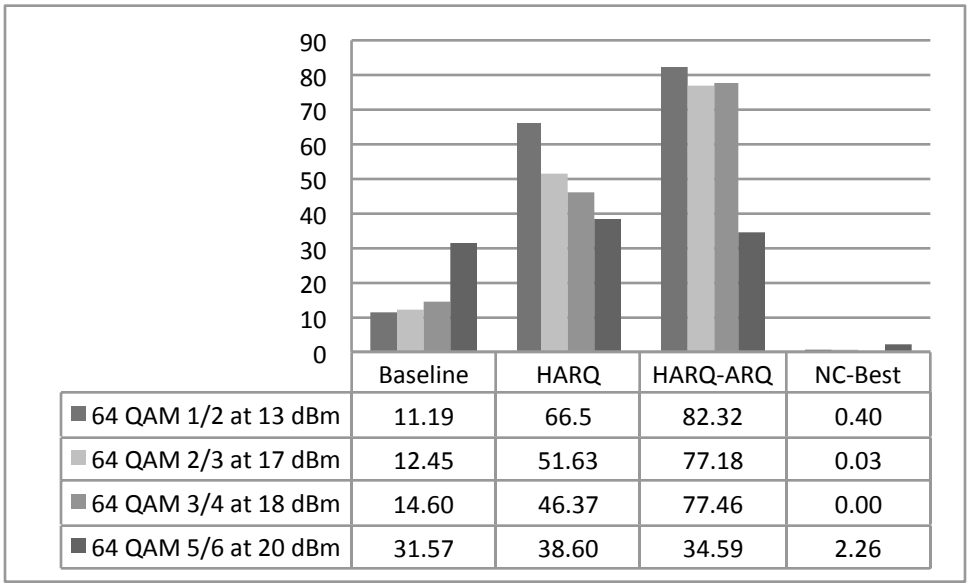


Figure 8-21: 64 QAM CTC 5/6 at 20 dBm. Shows 6 Mbps offered load downlink UFTP 50 MB file transfer delay.

8.3.5 Summary

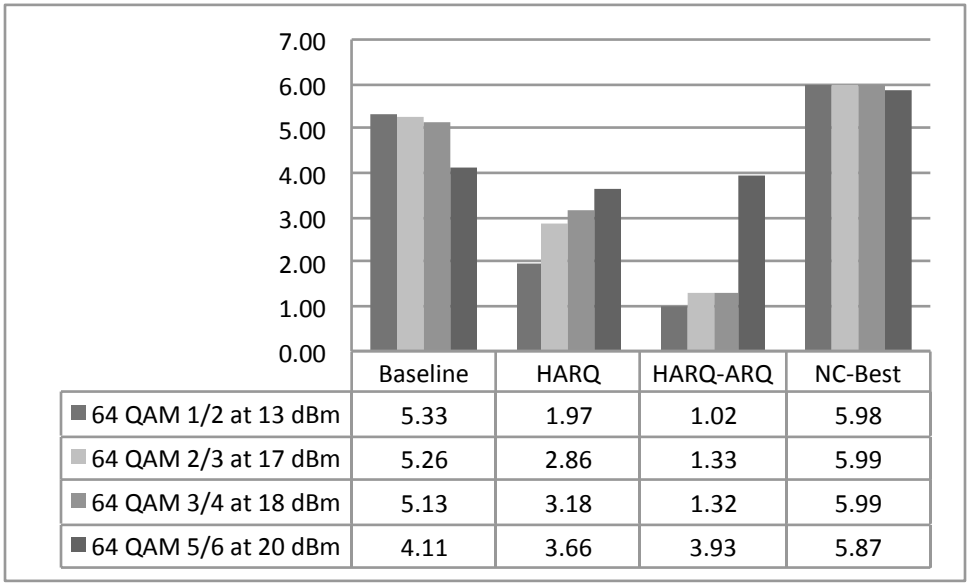
In this section, we compare all 4 MCSs and power levels and 4 configurations: Baseline, HARQ, HARQ-ARQ and NC-Best (the best configuration of all the NC configurations).

Figure 8-22 shows the 6 Mbps offered load downlink Iperf loss percentages for all four modulation and coding schemes, power levels and configurations. It is worth noticing that as the code rate increases, in the baseline, the loss percentage increases. In HARQ and HARQ-ARQ, the loss percentage decreases. In NC-Best, the loss percentage stays close to 0%.



	Baseline	HARQ	HARQ-ARQ	NC-Best
64 QAM 1/2 at 13 dBm	11.19	66.5	82.32	0.40
64 QAM 2/3 at 17 dBm	12.45	51.63	77.18	0.03
64 QAM 3/4 at 18 dBm	14.60	46.37	77.46	0.00
64 QAM 5/6 at 20 dBm	31.57	38.60	34.59	2.26

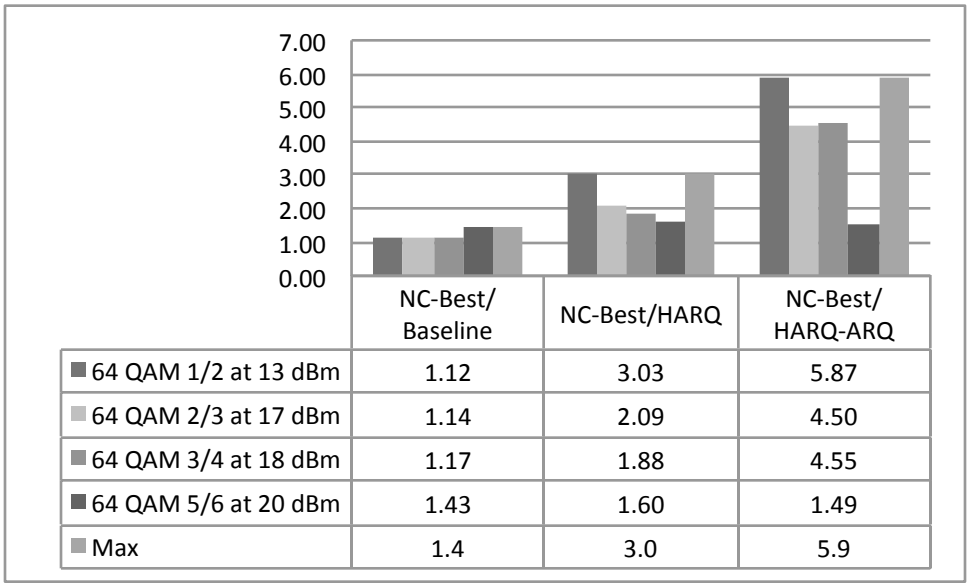
Figure 8-22: Loss (%) Comparison. Shows 6 Mbps offered load downlink Iperf loss percentages for all 4 Modulation and Coding Schemes and power levels and 4 configurations. NC-Best is the best configuration of all the NC configurations.



	Baseline	HARQ	HARQ-ARQ	NC-Best
64 QAM 1/2 at 13 dBm	5.33	1.97	1.02	5.98
64 QAM 2/3 at 17 dBm	5.26	2.86	1.33	5.99
64 QAM 3/4 at 18 dBm	5.13	3.18	1.32	5.99
64 QAM 5/6 at 20 dBm	4.11	3.66	3.93	5.87

Figure 8-23: Throughput (Mbps) Comparison. Shows 6 Mbps offered load downlink Iperf throughputs for all 4 Modulation and Coding Schemes (MCSs) and power levels and 4 configurations. NC-Best is the best configuration of all the NC configurations.

Figure 8-23 shows 6 Mbps offered load downlink Iperf throughputs for all 4 Modulation and Coding Schemes (MCSs) and power levels and 4 configurations. Notice that as the

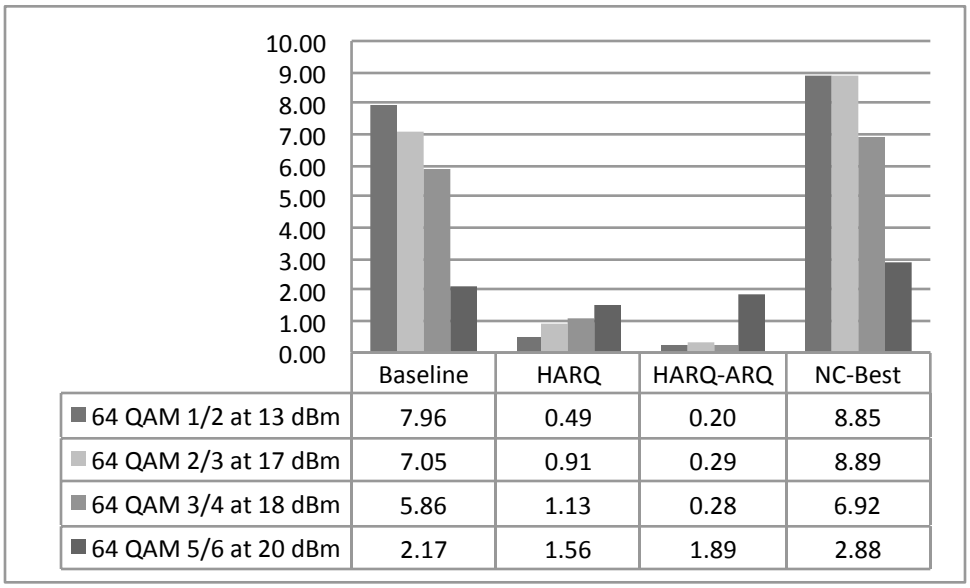


	NC-Best/ Baseline	NC-Best/HARQ	NC-Best/ HARQ-ARQ
64 QAM 1/2 at 13 dBm	1.12	3.03	5.87
64 QAM 2/3 at 17 dBm	1.14	2.09	4.50
64 QAM 3/4 at 18 dBm	1.17	1.88	4.55
64 QAM 5/6 at 20 dBm	1.43	1.60	1.49
Max	1.4	3.0	5.9

Figure 8-24: Throughput Ratio. Shows the throughput ratios of NC-Best/Baseline, NC-Best/HARQ and NC-Best/HARQ-ARQ for all 4 Modulation and Coding Schemes (MCSs) and power levels.

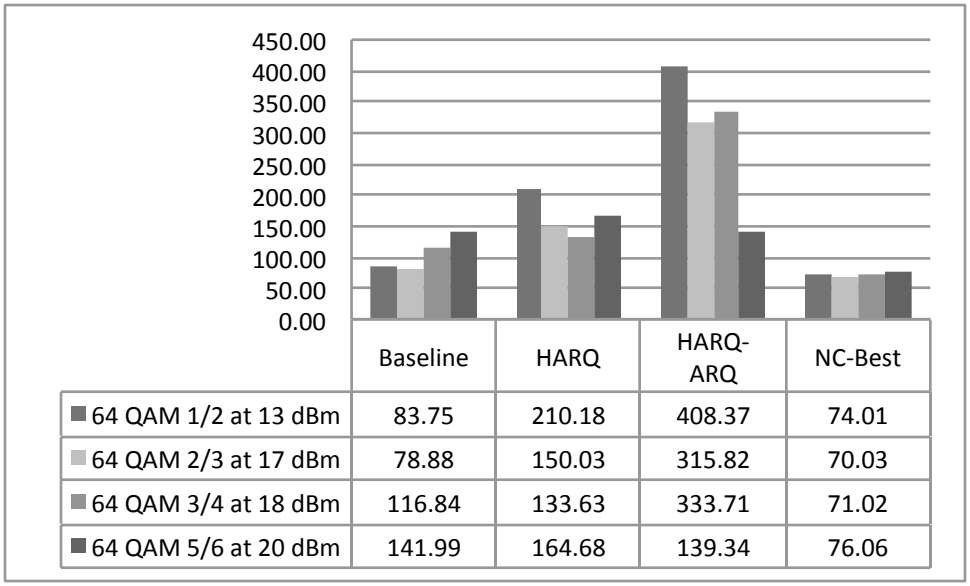
code rate increases, in Baseline, the throughput decreases. In HARQ and HARQ-ARQ, the throughput increases. In NC-Best, the throughput stays close to 6 Mbps. Figure 8-24 compares the throughput of NC-Best to that of Baseline, HARQ and HARQ-ARQ. NC-Best can deliver the throughput up to 1.4 times that of Baseline, 3.0 times that of HARQ and 5.9 times that of HARQ-ARQ.

Figure 8-25 shows 6 Mbps offered load downlink Iperf Throughput to Loss plus Extra Ratio (TLER) for all 4 Modulation and Coding Schemes (MCSs) and power levels and 4 configurations. Notice that as the code rate increases, in Baseline, the TLER decreases. In HARQ and HARQ-ARQ, the TLER increases. In NC-Best, TLER decreases.



	Baseline	HARQ	HARQ-ARQ	NC-Best
64 QAM 1/2 at 13 dBm	7.96	0.49	0.20	8.85
64 QAM 2/3 at 17 dBm	7.05	0.91	0.29	8.89
64 QAM 3/4 at 18 dBm	5.86	1.13	0.28	6.92
64 QAM 5/6 at 20 dBm	2.17	1.56	1.89	2.88

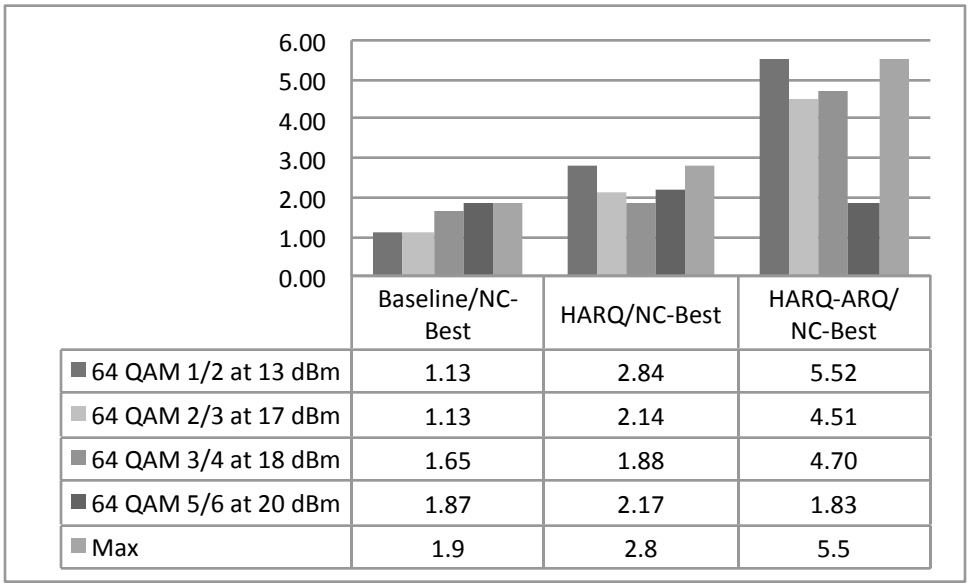
Figure 8-25: TLER Comparison. Shows 6 Mbps offered load downlink Iperf Throughput to Loss plus Extra Ratio (TLER) for all 4 Modulation and Coding Schemes (MCSs) and power levels and 4 configurations. NC-Best is the best configuration of all the NC configurations.



	Baseline	HARQ	HARQ-ARQ	NC-Best
64 QAM 1/2 at 13 dBm	83.75	210.18	408.37	74.01
64 QAM 2/3 at 17 dBm	78.88	150.03	315.82	70.03
64 QAM 3/4 at 18 dBm	116.84	133.63	333.71	71.02
64 QAM 5/6 at 20 dBm	141.99	164.68	139.34	76.06

Figure 8-26: File Transfer Delay (s) Comparison. Shows 6 Mbps offered load downlink UFTP 50 MB file transfer delays for all 4 Modulation and Coding Schemes (MCSs) and power levels and 4 configurations. NC-Best is the best configuration of all the NC configurations.

Figure 8-26 shows the 6 Mbps offered load downlink UFTP 50 MB file transfer delays for all 4 modulation and coding schemes (MCSs), power levels and configurations. Notice that



	Baseline/NC-Best	HARQ/NC-Best	HARQ-ARQ/ NC-Best
64 QAM 1/2 at 13 dBm	1.13	2.84	5.52
64 QAM 2/3 at 17 dBm	1.13	2.14	4.51
64 QAM 3/4 at 18 dBm	1.65	1.88	4.70
64 QAM 5/6 at 20 dBm	1.87	2.17	1.83
Max	1.9	2.8	5.5

Figure 8-27: File Transfer Delay Ratio. Shows the delay ratios of Baseline/NC-Best, HARQ/NC-Best and HARQ-ARQ/NC-Best for all 4 Modulation and Coding Schemes (MCSs) and power levels.

as the code rate increases, in Baseline, the delay tends to increase. In HARQ and HARQ-ARQ, the delay tends to decrease. In NC-Best, the delay remains close to 70 s. Figure 8-27 compares the delay of NC-Best to that of Baseline, the HARQ and the HARQ-ARQ. NC-Best can reduce the delay up to 1.9 times that of Baseline, 2.8 times that of the HARQ and 5.5 times that of the HARQ-ARQ.

8.4 Discussion

The trend across different MCSs and power levels seems consistent: NC configurations use the extra bandwidth to compensate for the lost bandwidth, increase the throughput and reduce the loss percentage significantly while HARQ and HARQ-ARQ greatly reduce the throughput and increase the lost bandwidth. The loss percentage graphs show that NC works well as a packet erasure code since the loss percentage of most NC configurations is less than that of Baseline and most NC configurations have a higher throughput than that of Baseline. How much loss percentage each NC configuration can reduce or how much

throughput it can gain depend on the number of redundancy packets per round (m). In most cases, we only need to configure m such that the resulting Code Rate (CR) is about the same as the throughput percentage of Baseline. Indeed, Figure 8-28 shows that the throughput percentage of Baseline closely matches the CR of the NC configuration with the highest throughput.

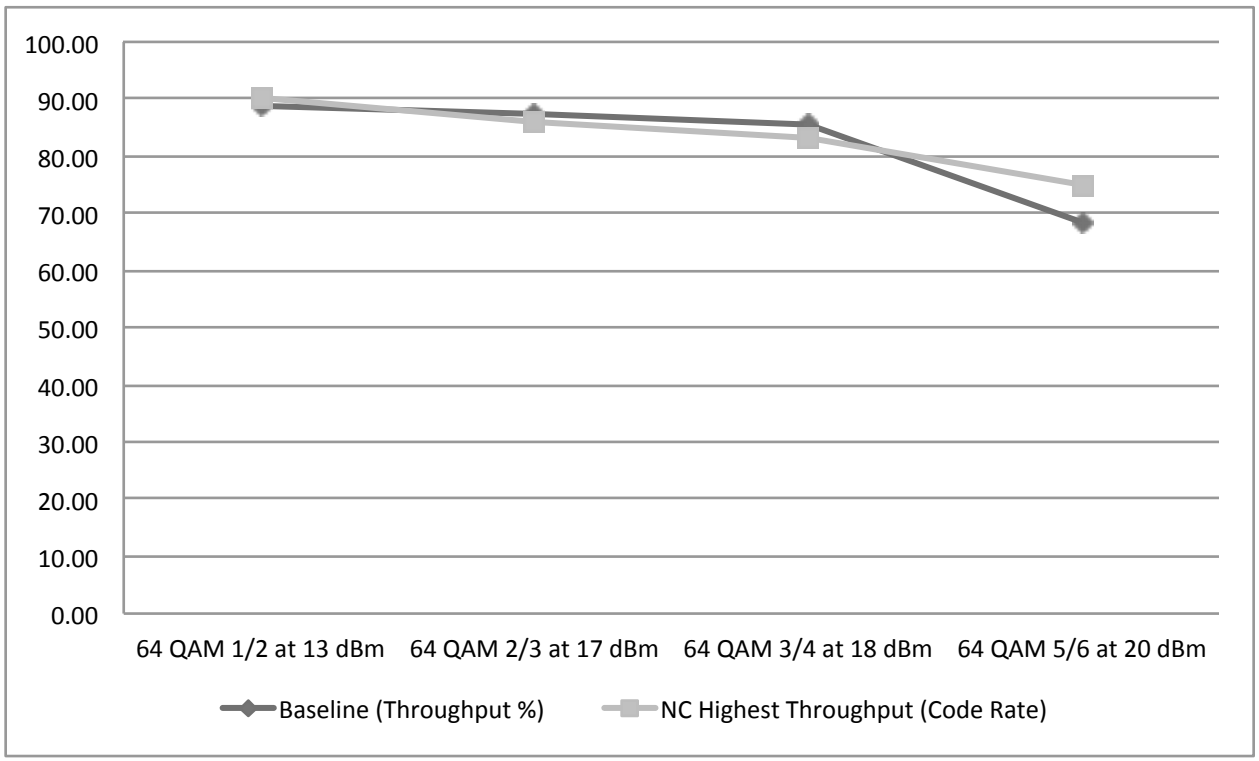


Figure 8-28: The throughput percentage of Baseline compared to the CR of the NC configuration with the highest throughput.

Now, a higher m may not be necessary but a lesser m may not be sufficient. When m is too low, most coded blocks cannot be recovered, thus incurring additional loss percentage, reducing the throughput and increasing file transfer delay. When m is too high, there are more redundant packets and more overhead, which may lead to buffer overflow. The delay tends to increase as the loss percentage of Baseline increases, except for the NC configurations. Thus, the higher the loss, the greater the benefit of NC.

At 6 Mbps of offered load, HARQ and ARQ do not perform well. Compared to Baseline, HARQ and HARQ-ARQ incur additional loss percentage, reduce throughput and increase file

transfer delay. The performance of HARQ and ARQ increases as the code rate of the MCS increases. Nevertheless, the performance of HARQ and HARQ-ARQ does not match that of the NC configurations. The low performance of HARQ and ARQ may be due to the following reasons. First, HARQ and ARQ may be poorly implemented. Second, the parameters of HARQ and ARQ, such as delay timeouts and the maximum number of retransmissions, used in the experiments may not be optimal and may need tuning. Third, HARQ and ARQ may just be inefficient as illustrated by the delay issue mentioned in Section 3.2.2, for instance. These conclusions, however, require further verification.

The results suggest that NC has a potential to replace HARQ and ARQ in future wireless network design. We infer that there are three main reasons why NC outperforms HARQ and ARQ. First, our NC protocol does not necessarily rely on an ACK or NACK packet, unlike HARQ and ARQ. In HARQ and ARQ, since the transmitter has to wait for an ACK or NACK packet in each transmission, the performance of HARQ and ARQ depends on the Round Trip Time (RTT). The longer the RTT, the lower the expected overall performance. In other words, the RTT limits the throughput of HARQ and ARQ. On the contrary, in NC, additional degrees of freedom (coded packets) can be sent ahead of time and lost packets can be recovered from the additionally received coded packets. Hence, the RTT does not limit the overall performance.

Second, the low throughput of HARQ and HARQ-ARQ suggests that HARQ and ARQ generate a large amount of redundancy. In our experiments, the redundancy for NC reaches 100% ($n = 120$) whereas for HARQ, each packet may be retransmitted up to 4 times (HARQ_MAX_RETRANSMISSION), thus incurring a potential redundancy of 400%. Such high levels of redundancy incur load increases that may reach or exceed the supported PHY-layer data rate. The excess load may cause buffer overflows at the MAC layer, resulting in a drop in throughput. In contrast, NC generates a reasonable amount of redundancy and thus offers higher throughput than HARQ and HARQ-ARQ.

Third, in HARQ and ARQ, each additional redundant packet can only compensate for a particular lost packet while in NC, each additional coded packet can potentially compensate

for any lost packet. In HARQ and ARQ, a particular packet is deemed lost if it is not successfully received after a fixed number of retransmissions. In NC, however, any lost packet can be replaced by the next coded packet. Therefore, NC is more robust to lost packets and also less sensitive to lost ACK packets than HARQ and ARQ, since it requires only one ACK packet per block.

As a conclusion, in this chapter, we presented and discussed the results of the network coding experiments. The experiments show that NC can outperform HARQ and ARQ, offering up to 5.9 times gain in throughput and 5.5 times reduction in file transfer delay. Thus, NC should be considered as a valid replacement for MAC layer HARQ and ARQ in order to maximize performance in 4G networks.

Chapter 9

Conclusion

In this chapter, we summarize the contributions of this thesis and list a number of important future work topics.

9.1 Contributions

This thesis addressed the design and implementation of a network-coding-enhanced network architecture for next generation wireless networks. In this design, network coding is used as a packet erasure code to provide resilience for remaining errors at the MAC layer. Using WiMAX as a case study, a number of experiments were conducted to validate the design decision. The results of the new design and of its implementation were compared to those of the HARQ and ARQ mechanisms in terms of packet loss percentage, throughput and file transfer delay. We demonstrated that network coding works very well as a packet erasure code. Compared to HARQ and ARQ mechanisms, network coding can offer up to 5.9 times gain in the throughput and 5.5 times reduction in the file transfer delay. We believe that the presented architecture, design and implementation will be instrumental in providing faster and more efficient next generation wireless network services at low cost.

9.2 Future Work

With the newly accessible GENI platforms, experimental studies have become easier and more accessible. Owing to the limited knowledge about the GENI WiMAX BS behavior, one direction of future work would be to experimentally evaluate and characterize the performance of the GENI WiMAX BSs, specifically the performance of the HARQ and ARQ mechanisms in different conditions.

Another direction is to experimentally study the sensitivity of the performance of the proposed design to different offered loads. The experimentation could also be extended to investigate various parameters of the proposed design. In our experiments, we only considered a single round of redundancy transmission (k equal to 1). For further study, we can try experimenting with multiple rounds. The number of concurrent encoder-decoder thread pairs (p) is another interesting parameter to vary in order to study its effect in more detail. We also recommend extending the design with an adaptive scheme to dynamically adjust various design parameters as briefly mentioned in Section 5.5. After tackling static Subscriber Stations (SSs), single-hop topologies and single-interface settings, the study may be broadened to mobile Subscriber Stations (SSs), multiple-hop topologies and multiple-interface settings.

The optimization of the decoding time is also an interesting direction to pursue further. Different decoding algorithms such as the Jacobi iterative method for finite field matrix inversion as suggested in Chapter 6 may be considered. Streaming SIMD Extensions (SSE) instruction set and Graphics Processing Unit (GPU) acceleration may also be considered.

The proposed design may be modified to work with Relay Stations (RSs) and in multiple unicast, multicast and broadcast scenarios. A handover scenario may also be interesting to investigate as it would provide a validation of the approach in a mobile environment. Furthermore, other network coding schemes such as N-in-1 NC [40] could be considered and provide performance comparisons.

It is clear that the design can be applied at different layers of the OSI model; direct implementation in industrial-grade WiMAX/LTE systems is a natural next step. Last but not least, the validation of our results through simulation and theoretical analysis is also an

important direction to explore.

Appendix A

Finite Field

In this section, we introduce finite field and its arithmetic. For more information, please refer to textbooks that treat finite fields [41].

Finite field or Galois field contains a finite number of elements. In this thesis, we use $\mathbf{GF}(p^n)$ to represent Galois field containing p^n number of elements, where p is a prime number, and n is a positive integer. Arithmetic operations in a finite field are different from standard integer arithmetic operations. All arithmetic operations performed in the finite field result in an element within that field.

Elements of $\mathbf{GF}(p^n)$ can be represented as polynomials of degree strictly less than n over $\mathbf{GF}(p)$. Arithmetic operations are performed modulo R, where R is an irreducible polynomial of degree n over $\mathbf{GF}(p)$, using polynomial long division. For example, when p is 2 and n is greater than 6, 89 can be represented as $x^6 + x^4 + x^3 + 1$.

In computer science applications, the operations are simplified when p is 2, making $\mathbf{GF}(2^n)$ popular choices for practical applications. In fact, we will be using $\mathbf{GF}(2^8)$ or Rijndael's finite field. Site [64] contains detailed description of Rijndael's finite field. Practical implementations such as the use of log table to speed up the calculation are also discussed there.

Bibliography

[1] GENI WiMAX platforms. http://wimax.orbit-lab.org/.

[2] Iperf. http://sourceforge.net/projects/iperf/.

[3] The netfilter.org project. http://www.netfilter.org.

[4] Padding (cryptography): Byte padding. http://en.wikipedia.org/wiki/Padding_(cryptography).

[5] Difference between 3G and 4G network technology. http://www.differencebetween.com/difference-between-3g-and-4g-network-technology/, February 26 2011.

[6] M.S. Abuzeid, Y.A. Fahmy, and M.M.S. El-Soudani. IR-HARQ vs. joint channel-network coding for cooperative wireless communication. *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, August 2011.

[7] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216, 2002.

[8] J.G. Andrews, A. Ghosh, and R. Muhamed. *Fundamentals of WiMAX: understanding broadband wireless networking*. Prentice Hall PTR, 2007.

[9] J.Y. Baek and Y.J. Suh. An adaptive arq–harq interworking scheme in wimax systems. *Computer Networks*, 2012.

[10] V.K. Bhargava and I.J. Fair. *Forward Error Correction Coding*. CRC Press LLC, 1999.

[11] D. Bush. UFTP. http://www.tcnj.edu/~bush/uftp.html.

[12] M.J. Chang, Z. Abichar, and C.Y. Hsu. WiMAX or LTE: who will lead the broadband mobile internet? *IT professional*, 12(3):26–32, 2010.

[13] P.A. Chou, Y. Wu, and K. Jain. Practical network coding. 2003.

[14] A.F. Dana, R. Gowaikar, R. Palanki, B. Hassibi, and M. Effros. Capacity of wireless erasure networks. *Information Theory, IEEE Transactions on*, 52(3):789–804, 2006.

[15] R.V. Dias, J.P. Leite, R.M. Amorim, A.N. Barreto, P.H.P. De Carvalho, and R.D. Vieira. Performance analysis of HARQ in WiMAX networks considering imperfect channel estimation. *Electrical Engineering*, 2010.

[16] M.W. El-Bahri, H. Boujernaa, and M. Siala. Performance comparison of type I, II and III hybrid ARQ schemes over AWGN channels. In *Industrial Technology, 2004. IEEE ICIT'04. 2004 IEEE International Conference on*, volume 3, pages 1417–1421. IEEE, 2004.

[17] A. Eryilmaz, A. Ozdaglar, and M. Médard. On delay performance gains from network coding. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 864–870. IEEE, 2006.

[18] P. Fan, C. Zhi, C. Wei, and K. Ben Letaief. Reliable relay assisted wireless multicast using network coding. *Selected Areas in Communications, IEEE Journal on*, 27(5):749–762, 2009.

[19] M. Ghaderi, D. Towsley, and J. Kurose. Reliability gain of network coding in lossy wireless networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 2171–2179. IEEE, 2008.

[20] A. Ghosh, J. Zhang, R. Muhamed, and J.G. Andrews. *Fundamentals of LTE*. Prentice Hall, 2010.

[21] IEEE 802.16 Working Group. IEEE 802.16: Broadband wireless metropolitan area networks (MANs). `http://standards.ieee.org/about/get/802/802.16.html`, 2005.

[22] IEEE 802.16 Working Group. IEEE 802.16 e–2005 IEEE standard for local and metropolitan area networks part 16: Air interface for fixed and mobile broadband wireless access systems amendment for physical and medium access control layers for combined fixed and mobile operation in licensed bands. `http://standards.ieee.org/getieee802/download/802.16e-2005.pdf`, 2005.

[23] Network Working Group. Computing the internet checksum. `http://tools.ietf.org/html/rfc1071`, 1988.

[24] D. Haley and A. Grant. Reversible low-density parity-check codes. *Information Theory, IEEE Transactions on*, 55(5):2016–2036, 2009.

[25] L. Hardesty. The power of random. `http://www.drdobbs.com/parallel/222700498`, 2010.

[26] L. Hardesty. Rethinking networking. `http://www.drdobbs.com/architecture-and-design/222900190`, 2010.

[27] T. Ho, R. Koetter, M. Medard, D.R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Information Theory, 2003. Proceedings. IEEE International Symposium on*, page 442. IEEE, 2003.

[28] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *Information Theory, IEEE Transactions on*, 52(10):4413–4430, 2006.

[29] S.K. Hong and J.M. Chung. Network-coding-based hybrid ARQ scheme for mobile relay networks. *Electronics letters*, 46(7):539–541, 2010.

[30] C.V.N. Index. Forecast and methodology, 2009–2014. *White paper, CISCO, June*, 2, 2010.

[31] S. Jakubczak, H. Rahul, and D. Katabi. One-size-fits-all wireless video. In *ACM SIGCOMM HotNets*, 2009.

[32] J. Jin and B. Li. Adaptive random network coding in WiMAX. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 2576–2580. IEEE, 2008.

[33] J. Jin, B. Li, and T. Kong. Is random network coding helpful in WiMAX? In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 2162–2170. IEEE, 2008.

[34] M. Jolfaei, S. Martin, and J. Mattfeldt. A new efficient selective repeat protocol for point-to-multipoint communication. In *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, volume 2, pages 1113–1117. IEEE, 1993.

[35] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: practical wireless network coding. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 243–254. ACM, 2006.

[36] P. Larsson. Analysis of multi-user ARQ with multiple unicast flows under non-iid reception probabilities. In *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*, pages 384–388. IEEE, 2007.

[37] P. Larsson. Multicast multiuser ARQ. In *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pages 1985–1990. IEEE, 2008.

[38] P. Larsson and N. Johansson. Multi-user ARQ. In *Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd*, volume 4, pages 2052–2057. IEEE, 2006.

[39] P. Larsson, B. Smida, T. Koike-Akino, and V. Tarokh. Analysis of network coded HARQ for multiple unicast flows. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.

[40] Z. Li, Q. Luo, and W. Featherstone. N-in-1 retransmission with network coding. *Wireless Communications, IEEE Transactions on*, 9(9):2689–2694, 2010.

[41] R. Lidl and H. Niederreiter. Finite fields, volume 20 of encyclopedia of mathematics and its applications, 1983.

[42] J. Lu, C.K. Wu, S. Xiao, and J.C. Du. A network coding based hybrid ARQ algorithm for wireless video broadcast. *SCIENCE CHINA Information Sciences*, pages 1–6, 2011.

[43] D. E. Lucani, F. H. P. Fitzek, M. Médard, and M. Stojanovic. Network coding for data dissemination: it is not what you know, but what your neighbors don't know. In *Proceedings of the 7th international conference on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 407–414. IEEE Press, 2009.

[44] D.E. Lucani, M. Médard, and M. Stojanovic. Systematic network coding for time-division duplexing. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pages 2403–2407. IEEE, 2010.

[45] D.S. Lun, M. Médard, R. Koetter, and M. Effros. Further results on coding for reliable communication over packet networks. In *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pages 1848–1852. IEEE, 2005.

[46] D.S. Lun, M. Médard, R. Koetter, and M. Effros. On coding for reliable communication over packet networks. *Physical Communication*, 1(1):3–20, 2008.

[47] J. Manssour, A. Osseiran, and B. Slimane. A unicast retransmission scheme based on network coding. *Vehicular Technology, IEEE Transactions on*, pages 1–1, 2011.

[48] M. Medard and A. Sprintson. *Network Coding: Fundamentals and Applications*. Academic Press, 2011.

[49] J. Metzner. An improved broadcast retransmission protocol. *Communications, IEEE Transactions on*, 32(6):679–683, 1984.

[50] D. Nguyen, T. Tran, T. Nguyen, and B. Bose. Wireless broadcast using network coding. *Vehicular Technology, IEEE Transactions on*, 58(2):914–925, 2009.

[51] Q. Peng, T. Zhang, and L. Cuthbert. Research on network coding based hybrid-ARQ scheme for wireless networks. In *Communication Systems (ICCS), 2010 IEEE International Conference on*, pages 218–222. IEEE, 2010.

[52] S. Pu, Z. He, X. Lin, and W. Wu. Performance analysis of joint chase combining and network coding in wireless broadcast retransmission. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*, pages 1–4. IEEE, 2008.

[53] J. Qureshi, C.H. Foh, and J. Cai. An efficient network coding based retransmission algorithm for wireless multicasts. *CoRR*, abs/1008.1842, 2010.

[54] B. Rohani and K. Homayounfar. Physical layer network coding for wireless applications: A survey. Technical report, IEICE, Tech. Rep. 445, 2009.

[55] A. Sayenko, V. Tykhomyrov, H. Martikainen, and O. Alanen. Performance analysis of the IEEE 802.16 ARQ mechanism. In *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, pages 314–322. ACM, 2007.

[56] G. Strang. *Introduction to linear algebra*. Wellesley Cambridge Pr, 2003.

[57] Y. Sun, Y. Li, and X. Wang. Cooperative hybrid-ARQ protocol with network coding. In *Communications and Networking in China, 2009. ChinaCOM 2009. Fourth International Conference on*, pages 1–5. IEEE, 2009.

[58] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros. Network coding meets TCP. In *INFOCOM 2009, IEEE*, pages 280–288. IEEE, 2009.

[59] J.K. Sundararajan, D. Shah, and M. Médard. ARQ for network coding. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 1651–1655. IEEE, 2008.

[60] J.K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros. Network coding meets TCP: Theory and implementation. *Proceedings of the IEEE*, 99(3):490–512, 2011.

[61] R. Thobaben. Joint network/channel coding for multi-user hybrid-ARQ. In *Source and Channel Coding (SCC), 2008 7th International ITG Conference on*, pages 1–6. VDE, 2008.

[62] T. Tran, T. Nguyen, and B. Bose. A joint network-channel coding technique for single-hop wireless networks. In *Network Coding, Theory and Applications, 2008. NetCod 2008. Fourth Workshop on*, pages 1–6. IEEE, 2008.

[63] T. Tran, T. Nguyen, B. Bose, and V. Gopal. A hybrid network coding technique for single-hop wireless networks. *Selected Areas in Communications, IEEE Journal on*, 27(5):685–698, 2009.

[64] S. Trenholme. Aes' galois field. `http://www.samiam.org/galois.html`.

[65] International Telecommunication Union. *Measuring the information society: The ICT development index*. International Telecommunication Union, 2009.

[66] J.H. Van Lint. *Introduction to coding theory*, volume 86. Springer Verlag, 1999.

[67] Q.T. Vien, H.X. Nguyen, J. Choi, B. Stewart, and H. Tianfield. Network coding-based block ACK for wireless relay networks. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1–5. IEEE, 2011.

[68] Q.T. Vien, L.N. Tran, and H.X. Nguyen. Network coding-based ARQ retransmission strategies for two-way wireless relay networks. In *Software, Telecommunications and Computer Networks (SoftCOM), 2010 International Conference on*, pages 180–184. IEEE, 2010.

[69] D. Wetteroth. *OSI reference model for telecommunications*. McGraw-Hill Professional, 2001.

[70] A.A. Yazdi, S. Sorour, S. Valaee, and R.Y. Kim. Optimum network coding for delay sensitive applications in WiMAX unicast. In *INFOCOM 2009, IEEE*, pages 2576–2580. IEEE, 2009.

[71] R. W. Yeung, S.Y. Li, and N. Cai. *Network coding theory*. Now Pub, 2006.

[72] S. Yong and L.B. Sung. XOR retransmission in multicast error recovery. In *Networks, 2000.(ICON 2000). Proceedings. IEEE International Conference on*, pages 336–340. IEEE, 2000.

[73] S. Yun, H. Kim, and K. Tan. Towards zero retransmission overhead: A symbol level network coding approach to retransmission. *Mobile Computing, IEEE Transactions on*, pages 1–1, 2011.

[74] Z. Zhang, T. Lv, X. Su, and H. Gao. Dual XOR in the air: a network coding based retransmission scheme for wireless broadcasting. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.