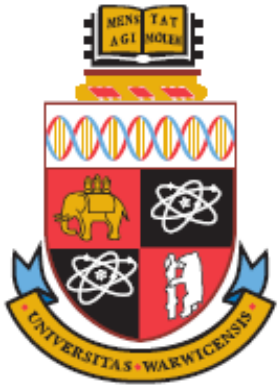


Network Coding for Computer Networking

by

Alaa Alsebae

A thesis submitted in partial fulfilment of the requirements for

the degree of Doctor of Philosophy

School of Engineering

November 2014

To my beloved country, Syria

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to my supervisors Dr. Mark Leeson and Prof. Roger Green without whom this work would have not been completed. Through continuous support, patience, motivation and immense knowledge, their guidance enlightened my research and writing up journey.

I am indebted to my parents who have shown a full support during this work despite the suffering they endured during the hard time in Syria. Special thanks go to my brother, Dr. Belal, who spared no efforts to strengthen my morals and motivate me to complete this work.

I am also very thankful to all my friends who kindly provided their best advice throughout this work especially Ms. Zeina Rihawi, School of Engineering and Mr. Karim El Haloui, the Warwick Mathematics Institute, for their invaluable support and discussions.

Finally, I would like to thank Mr. Abdulqader Alyasin who kindly supported and helped me to improve my English skills and make this work presentable.

Declaration

I herewith declare that this thesis contains my own research performed under the supervision of Dr. Mark Leeson and Prof. Roger Green, without assistance of third parties, unless stated otherwise. No part of this thesis was previously published or submitted for a degree at any other university.

Abstract

Conventional communication networks route data packets in a *store-and-forward* mode. A router buffers received packets and forwards them intact towards their intended destination. Network Coding (NC), however, generalises this method by allowing the router to perform algebraic operations on the packets before forwarding them. The purpose of NC is to improve the network performance to achieve its maximum capacity also known as *max-flow min-cut* bound. NC has become very well established in the field of information theory, however, practical implementations in real-world networks is yet to be explored. In this thesis, new implementations of NC are brought forward. The effect of NC on flow error control protocols and queuing over computer networks is investigated by establishing and designing a mathematical and simulation framework. One goal of such investigation is to understand how NC technique can reduce the number of packets required to acknowledge the reception of those sent over the network while error-control schemes are employed. Another goal is to control the network queuing stability by reducing the number of packets required to convey a set of information. A custom-built simulator based on SimEvents® has been developed in order to model several scenarios within this approach. The work in this thesis is divided into two key parts.

The objective of the first part is to study the performance of communication networks employing error control protocols when NC is adopted. In particular, two main Automatic Repeat reQuest (ARQ) schemes are invoked, namely the Stop-and-Wait (SW) and Selective Repeat (SR) ARQ. Results show that in unicast point-to-point communication, the proposed NC scheme offers an increase in the throughput over traditional SW ARQ between 2.5% and 50.5% at each link, with negligible decoding delay. Additionally, in a *Butterfly* network, SR ARQ employing NC achieves a throughput gain between 22% and 44% over traditional SR ARQ when the number of incoming links to the intermediate node varies between 2 and 5. Moreover, in an extended *Butterfly* network, NC offered a throughput increase of up to 48% under an error-free scenario and 50% in the presence of errors.

Despite the extensive research on synchronous NC performance in various fields, little has been said about its queuing behaviour. One assumption is that packets are served following a Poisson distribution. The packets from different streams are coded prior to being served and then exit through only one stream. This study determines the arrival distribution that coded packets follow at the serving node. In general this leads to study general queuing systems of type G/M/1. Hence, the objective of the second part of this study is twofold. The study aims to determine the distribution of the coded packets and estimate the waiting time faced by coded packets before their complete serving process. Results show that NC brings a new solution for queuing stability as evidenced by the small waiting time the coded packets spend in the intermediate node queue before serving. This work is further enhanced by studying the server utilization in traditional routing and NC scenarios. NC-based M/M/1 with finite capacity K is also analysed to investigate packet loss probability for both scenarios.

Based on the results achieved, the utilization of NC in error-prone and long propagation delay networks is recommended. Additionally, since the work provides an insightful prediction of particular networks queuing behaviour, employing

synchronous NC can bring a solution for systems' stability with packet-controlled sources and limited input buffers.

List of Abbreviations

ARQ	Automatic Repeat reQuest
AWGN	Additive white Gaussian Noise
BER/ PER	Bit/ Packet Error Rate
BPSK	Binary Phase Shift Keying
CRC	Cyclic Redundancy Check
FEC	Forward Error Correction
FIFO	First-In-First-Out
GBN	Go-Back-N
MWT	Mean Waiting Time
NC	Network Coding
RLC	Random Linear Coding
SR ARQ	Selective Repeat ARQ
SW ARQ	Stop-and-Wait ARQ
TR	Traditional Routing
ONC	Opportunistic NC
NP-hard	Non-deterministic Polynomial-time hard
M/M/1	Poisson arrival and departure queue with one server and infinite buffer
M/M/1/K	Poisson arrival and departure queue with one server and finite buffer of size K
M/D/1	Poisson arrival and deterministic departure queue with one server.
G/G/1	General arrival and departure queue with one server and infinite buffer

Publications Associated with This Research Work

Published Papers

1. A. Alsebae, M. S. Leeson and R. J. Green, “On modelling network coded ARQ-Based channels," *International Journal of Space-Based and Situated Computing (IJSSC), June, 2014*.
2. A. Alsebae, M. S. Leeson and R. J. Green, "The Throughput Benefits of Network Coding for SW-ARQ Communication," *The 27th International Conference on Advanced Information Networking and Applications (AINA 2013)*, 25-28 March 2013.
3. A. Alsebae, M. S. Leeson and R. J. Green, "SimEvents-based Modeling and Simulation Study of Stop-and-Wait Protocol," *The 5th International Conference on Modeling, Identification and Control (ICMIC 2013),* Aug31-1-2 Sept 2013.
4. A. Alsebae, M. S. Leeson and R. J. Green, "The Throughput Benefits of Network Coding for SR ARQ Communication," the *5th Computer Science and Electronic Engineering Conference (CEEC 2013)*, 17 -18 Sept 2013.
5. A. Alsebae, M. S. Leeson and R. J. Green, “Performance of a Network Coding Queuing Model with Deterministic Service," *The 9th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP), 23-25 July 2014.*

Submitted Papers

1. A. Alsebae, M. S. Leeson and R. J. Green, “SimEvents-based analysis for an improved selective repeat protocol in network coding scheme," *Journal of Performance Evaluation*. (under review)
2. A. Alsebae, M. S. Leeson and R. J. Green, “Synchronous network coding queuing analysis," *Journal of Performance Evaluation.* (under review)

Chapter 1

Introduction

Since the advent of data communication systems, conventional communication networks have been operating in a *store-and-forward* mode; a router accepts incoming packets, stores, and forwards them intact towards their ultimate destination. In 2000 [1], a new concept, *network coding,* fundamentally changed the 'received wisdom' which states that information transmission in a point-to-point network is equivalent to mail delivery in a postal system.

Network coding (NC) generalizes classical routing by assuming that routers are allowed to modify incoming packets prior to forwarding them. The reason for this assumption is that - when there is more than one destination node - it is not sufficient to replicate information at certain intermediate nodes, as illustrated in the following example.

Consider a communication system with two wireless nodes S_1 and S_2 that generate packets *a* and *b*, respectively. The packets are to be exchanged through a relay node, W, assuming that the wireless nodes cannot either transmit and receive simultaneously, or receive a transmission from more than one neighbouring node at a time, as shown in Figure 1.1.

(a) (b)

Figure 1.1: (a) Traditional store-and-forward network and (b) Network coding based network.

S_1 and S_2 send packets *a* and *b* to W, respectively. Without NC, node W simply relays packet *a* from S_1 to S_2 and vice-versa for packet *b*; as a result, this scheme will take a total of four time units to complete. On the other hand, adopting NC allows the intermediate node W to generate a new packet based on the packets it receives from S_1 and S_2. For example, applying modulo 2 addition or performing an Xor operation on packets *a* and *b* results in the transmission of a new packet a⊕b to S_1 and S_2. Since S_1 and S_2 already have packets *a* and *b* respectively, S_1 can recover packet *b* by applying a⊕(a⊕b), and S_2 can recover packet *a* by performing b⊕(a⊕b). Clearly, using NC, only three transmissions are required for S_1 and S_2 to exchange their packets.

Network coding can be performed in both a synchronous and an asynchronous fashion. In synchronous NC, a coding node matches incoming packets from different streams by holding previously arrived packets until all required packets are present. However, with random packet arrival, packet delay will occur, because packets are required to wait for the other packets to undergo NC. Hence, packets arriving at finite buffers will be lost or dropped if synchronous NC is adopted, since packets that wait to be network-coded can exceed the buffer capacity. On the other hand, coding functions can also be bounded by time; a coding node may transmit packets without coding, if all streams are yet to have their packets ready within a predefined time. This is referred to as asynchronous NC, which was first implemented in wireless networks as described in [2, 3].

Improving the network performance using NC depends on the number of incoming packets to be combined, the way in which they are combined, and the information which the coding node requires about the other coding nodes and the network

topology. NC has shown its greatest impact in the fields of mathematics, information theory and computer and communication networks.

Aims and Motivations

Reduced to its basic form, a computer or communication network consists of nodes and communication channels. Therefore, one of the most important issues in communication networks is ensuring that messages at the sender nodes are ultimately delivered to the destination correctly, and without duplication. Additionally, packets flowing from node-to-node should not form undesirable queues when the demand for a service is larger than the capacity for the service available at the node. Hence, in computer networks, flow error control protocols are adopted to regulate data flow and compensate for transmission node errors [4]. In common error control techniques, the receiver performs error detection and returns a positive acknowledgment (ACK) for error-free transmission, or a negative acknowledgment (NAK) when an error is detected. Additionally, the source retransmits any packet that has not been acknowledged within a predetermined amount of time. This procedure is referred to as Automatic Repeat ReQuest (ARQ) [4, 5].

The general principle of ARQ protocols is to request the transmitter to repeat the sending of erroneously-received packets; this is done after checking the validity of each packet using an error-detection technique. ARQ mainly operates according to three protocols, listed in the order of an increasing complexity: Stop-and-Wait (SW), Go-Back-N (GBN), and Selective Repeat (SR).

Within the same context, in computer networks, a queuing delay may occur at an intermediate node if packets of k incoming links with arrival rates λ_k address one output port of the node - that is, the server must serve packets with an arrival rate

equal to $\lambda = \sum_{i=1}^{k} \lambda_i$, which could exceed the service rate, μ. Hence, *flow control* and *queuing analysis* play an important role in computer and communication design, and their study gives the flexibility to adjust various parameters in the early stages of network planning rather than in the operational phase.

The advantages of NC, such as combining several packets into one, will allow the communication systems to send only one acknowledgment signal for several combined packets [6], and to reduce congestion at the intermediate node queues [7, 8]. This reduction will significantly improve the communication system throughput, transmission delay and energy consumption.

Motivated by the aforementioned NC characteristics, and the potential system performance improvements, the work in this thesis aims to provide SimEvents®-based models of six network scenarios that show the merit of using NC, which are presented in two parts: The first investigates the benefits of NC when employed with unicast SW and SR ARQ, multicast SR ARQ, and extended multicast SR ARQ networks. The second presents a network model of two types of queuing models, M/M/1 and M/D/1, residing at an intermediate node and employing synchronous NC defined in the following chapter.

SimEvents® is an event-driven software tool which simulates system behaviour using a hierarchical modelling strategy, working upward from the basic subsystems to represent and compile the complete network [9]. SimEvents, which is a library element in MATLAB® Simulink with a user-friendly Graphical User Interface (GUI), has the ability to utilize relevant library elements for realistic network modelling. It consists of predefined blocks, such as queues, servers and switches to enable the representation of communication systems and aggregation of statistics

such as delay, throughput, packet loss, average queue length, and other metrics. Paths can be established by connecting gate and switching blocks on which packets can travel in response to events and according to selective switching criteria and imposed delays. This allows the modelling of simple and complex networks of queues and servers. Therefore, SimEvents is suitable for the particular scenarios analysed in this thesis. Despite its features, there have been as yet few academic papers based on the use of SimEvents [10-12]. Most studies on computer and communication networks have been extensively utilizing the well-known network simulators NS-2 and OPNET [13-15].

The work will in general focus purely on the use of NC from the error-control schemes and the queuing system perspective. To the best of the author's knowledge, NC with these settings has not been applied in this way with this focus and it will be shown that its use will bring benefits to the research area. It should also be stated that the method and results presented here are not mutually exclusive, and this work is not shown to be a solution, but part of the solution.

It is hoped that the benefits of this work are taken further by others researchers, particularly the network optimisers who may be able to find the conditions for which the network have high throughput, and potentially fulfilling the system design requirements. It is also hoped that this thesis will as well act as a guideline on the usage of SimEvents as a tool for analytical validation.

The list of contributions drives the entire thesis

The following contributions have been produced as a result of the work contained within this thesis.

1. The work starts by constructing a control unit to be used for the first time in SimEvents®, enabling the simulation of traditional SW ARQ. This control unit solves the problem where, in SW ARQ, the sender cannot transmit a new packet until it receives an acknowledgment from the previous packet. Since there is no acknowledgment received prior to the first packet, a mechanism to solve this problem is proposed.
2. The work then exploits the control unit to model a communication system employing traditional SW protocol as an error-control scheme.
3. The modelling of the traditional SW protocol is then integrated to investigate the throughput benefits of NC when employed along with the SW ARQ protocol.
4. The promising results obtained by employing NC with SW ARQ extend the work to encompass the investigation of the throughput benefits of NC, when employed along with the SR ARQ protocol in both unicast and multicast butterfly networks.
5. The investigation of the throughput benefits of NC, when employed along with SR ARQ protocol in a simple butterfly network, allows the expansion of the work to investigate NC when employed in an extended butterfly network with two intermediate coding nodes and three sinks.
6. The introduction of synchronous NC, when studying its benefits in the extended butterfly network, highlights the need to study the effects of synchronous NC on an intermediate node queuing system. This is manifested by analysing the performance of an NC-based queuing model with exponential inter-arrival and an exponential service, M/M/1. A theoretical construction of the coded packets distribution behaviour under this arrangement is provided and validated via

simulation; this is in addition to the investigation of the mean waiting time and server utilization of the system.

7. Under the same arrangement of the M/M/1 system, the performance analysis of a NC-based queuing model with exponential arrival and deterministic service, M/D/1, is carried out by exploiting the theoretical results of the coded packets behaviour previously obtained. This is further validated via simulation under the new arrangement where the mean waiting time and the server utilization are also provided.

Thesis Outline

The thesis is organised into 9 chapters, of which Chapters 3, 4, 5 and 6 constitute the first part, and Chapters 7 and 8 represent the second.

Chapter 2 discusses the theory and practice of the NC technique, and the literature relevant to the study. Specifically, it explores the fundamental concepts of NC, and also the various sub-types of NC such as Linear NC (LNC), random and deterministic NC. Synchronous and asynchronous NC and their applications in both error control schemes and queuing theory are also explained.

Chapter 3 elaborates the three main types of ARQ scheme. The main focus will be on SW and SR ARQ. Within this chapter, a simulation-based analysis of the traditional SW ARQ protocol performance is provided. A transmitter control unit is proposed to tackle the issue of sending the first packet without a previous acknowledgment (ACK) signal being required from the receiver; this enables the implementation of NC in the SW ARQ scheme proposed in the following chapter.

Chapter 4 illustrates the usefulness of adopting a modified NC scheme in a SW ARQ based point-to-point communication system employed at an intermediate link

in a multicast network. A Vandermonde coding matrix is utilised so that the receiver will be able to send one ACK signal for several linearly-coded packets. For various packet error rates and propagations times, it is shown that the link throughput can be greatly improved, depending on the number of incoming links to which the intermediate nodes are connected.

Chapter 5 investigates the throughput gain that NC can offer to both unicast and multicast scenarios, adopting SR ARQ as an error-control scheme. In both unicast and multicast scenarios, k of n-dimensional packets are firstly coded into only one packet of size n, and then advanced into the channel, which makes it less prone to errors and consume less transmission time. The throughput performance of SR and SR NC-based systems is compared under various packet error rates and propagation delays.

Chapter 6 extends the work carried out in **Chapter 5** by utilizing NC in an extended *butterfly* scenario, where a network has two encoding nodes and three sinks. The introduction of a synchronization scheme - where a destination node holds the decoding operation until all required coding packets from all flows arrive - highlights the importance of studying the queuing behaviour of a coding node, since its input buffers become intractable when the required packets in one flow take a long time to arrive, which is the topic discussed in both **Chapter 7** and **Chapter 8**.

Chapter 7 explores the consequences of applying synchronous NC on an intermediate queuing system. It starts with providing a mathematical derivation and simulation of the behaviour of coded packets in a Poisson arrival and exponential service queuing model, M/M/1. Theoretical and simulation results show that the Poisson-distributed packets coming from different streams follow a general random

distribution after coding. Additionally, NC is shown to achieve significant queuing stability in terms of waiting time and packet loss probability, compared to the case of traditional routing. Server utilization of both traditional routing and NC is also provided. Furthermore, a study of packet loss probability is investigated for an M/M/1 queuing model with limited capacity *K*.

Chapter 8 exploits the results obtained from **Chapter 7**, and studies the behaviour of coded packets in a Poisson-arrival and deterministic service, M/D/1 queuing model. The coded packets distribution expression is again stated and verified via simulation. Simulation-based results of performance measures including the comparison of the FIFO queue mean waiting time and server utilization for merged and coded packets are also presented.

Chapter 9 concludes the research presented throughout the thesis, and summarizes the major results achieved from the previous chapters. This is followed by suggestions for further research that may be considered given the results presented here.

Chapter 2

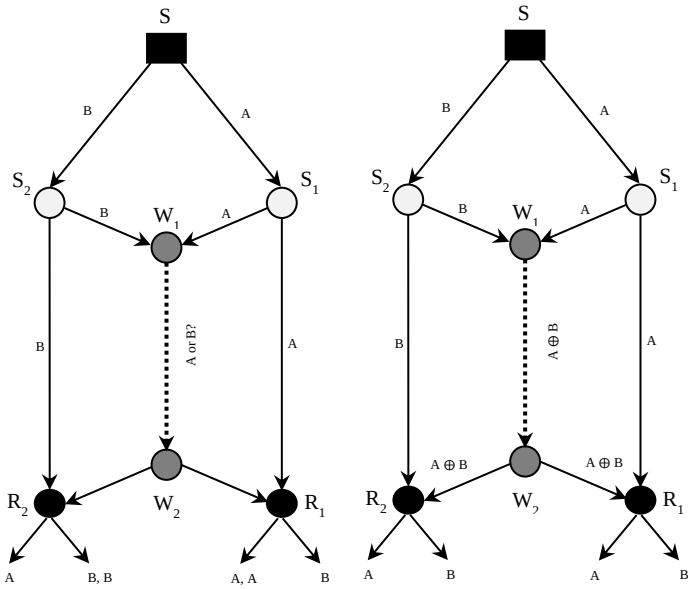
Network Coding

2.1 Introduction

Chapter 1 has provided a brief introduction to the concept of network coding (NC). The goal of this chapter is to detail the theory pertaining NC and its application in the subject areas of error control protocols and queuing theory. Introduced in [1], NC can achieve the multicast capacity between the sending node and each receiving node. Unlike conventional *store-and-forward* routing [16], NC allows the intermediate nodes to perform mathematical operations on data received at their incoming links when information is multicast in a network. The concept of NC is best illustrated by the *butterfly* network presented in Figure 2.1, which is considered as the starting point of NC.

The butterfly model is a directed graph $G = (V, E)$, where V is the set of nodes that encompasses hosts and routers, and E is the set of edges that connects these routers together. The source node $S \subset V$ generates information bits A and B at a rate of two per time unit, and wishes to convey these bits to both sinks R_1 and $R_2 \subset V$. Each link in the network has the capacity to transmit one single bit at a time. Nodes W_1 and W_2 are the intermediate nodes, which act as typical switching or relay stations. A key feature of this network is that it contains a bottleneck link (W_1, W_2).

When the network is operated based on a traditional routing (TR) scheme, node W_1 can only advance one bit downward through the bottleneck link (W_1, W_2) upon receiving A and B, causing a congestion problem.



(a) Traditional routing (b) Network coding

Figure 2.1: A network that requires coding to achieve multicast

If A is forwarded through W_1, sink R_1 will, in a single time slot, receive A twice but not B; the same applies for R_2 when B is forwarded. Nevertheless, by allowing W_1 to combine A and B simply using the modulo 2 sum (XOR function), R_1 and R_2 receive pairs (A, A⊕B) and (B, A⊕B), respectively. This allows destination R_1 to recover B= A⊕(A⊕B), and R_2 to recover A= B⊕(A⊕B). Therefore, R_1 and R_2 can easily recover the original packet bits by performing a simple algebraic operation, but at the expense of an encoding and decoding operation cost at the intermediate and destination nodes, respectively. The transmission procedure is summarised in Table 2.1 and Table 2.2 for traditional routing and network coding, respectively.

Table 2.1: Data transmission in a *butterfly* network using traditional routing.

Time slot	A	B
0	$S \rightarrow S_1$	$S \rightarrow S_2$
1	$S_1 \rightarrow W_1, S_1 \rightarrow R_1$	$S_2 \rightarrow W_1, S_2 \rightarrow R_2$
2	$W_1 \rightarrow W_2$	W_1
3	$W_2 \rightarrow R_2$	$W_1 \rightarrow W_2$
4	R_2	$W_2 \rightarrow R_1$

Table 2.2: Data transmission in a *butterfly* network using network coding.

Time slot	A	B	$A \oplus B$
0	$S \rightarrow S_1$	$S \rightarrow S_2$	
1	$S_1 \rightarrow W_1, S_1 \rightarrow R_1$	$S_2 \rightarrow W_1, S_2 \rightarrow R_2$	
2	R_1	R_2	$W_1 \rightarrow W_2$
3	R_1	R_2	$W_2 \rightarrow R_1$, $W_2 \rightarrow R_2$

It can be seen that traditional routing consumes a total of four time-slots to transmit data bits A and B; however, NC is capable of conveying A and B in three time-slots, which results in 2/3 = 0.667 bits per time-slot compared to 0.5 bits per time-slot in TR. NC can therefore help in better-sharing the available network resources, and

thus offers numerous advantages over the conventional routing in multicast throughput [1, 17, 18].

It has been proved [19] that in an acyclic coding networks, the number of encoding nodes required to achieve the network capacity is bounded by h^3k^3, where h is the number of packets needed to be delivered to a set of k sinks. It is been also observed that the number of encoding nodes is independent of the size of the network and is bounded by h^3k^3.

It is important to determine the location and number of nodes which should employ NC, however, finding a minimal set of encoding nodes where coding is required is NP-hard [19]. Several papers [17, 20-25] have addressed coding and decoding options, and have established the theoretical ability of NC to improve network throughput. An algebraic framework for NC with an investigation of linear network codes for directed graphs with cycles was developed in [23]. This scheme was later used by [20] to show that random NC coefficients can be efficiently constructed. An efficient construction of deterministic network codes - when the topology is known - was proposed in [22]. This is manifested by providing deterministic polynomial time algorithms for designing linear codes, given a directed acyclic graph with an edge, of unit capacity, that is tolerant to edge failure.

The idea of random linear network coding was first implemented by Chou et al. [26]; the authors proposed a practical real-world NC scheme by designing a packet format that removes the need for any centralized knowledge of the graph topology and the encoding or decoding functions. A stream of information is divided into generations, and random NC is applied within each generation. Coding coefficients are carried by the coded packets before transmission. Practical NC can achieve throughput close to

capacity with low delay [26]. Based upon this practical NC scheme, a commercial application for file content distribution (named Avalanche) is proposed in [27]. NC has diverse applications in wired and wireless networks, such as peer-to-peer networks, sensor, *ad hoc* and cellular networks [28].

2.2 Network Coding in Wireless Networks

The use of NC has also demonstrated advantages in increasing the throughput and bandwidth efficiency of wireless networks, and in saving wireless resources [3, 21, 29-31]. As an illustration, consider a communication system shown in Figure 2.2, with two wireless nodes A and C, that wish to exchange their bits through an intermediate relay node B. It is assumed that the wireless node cannot either receive a transmission from more than one neighbouring node, or simultaneously transmit and receive data.

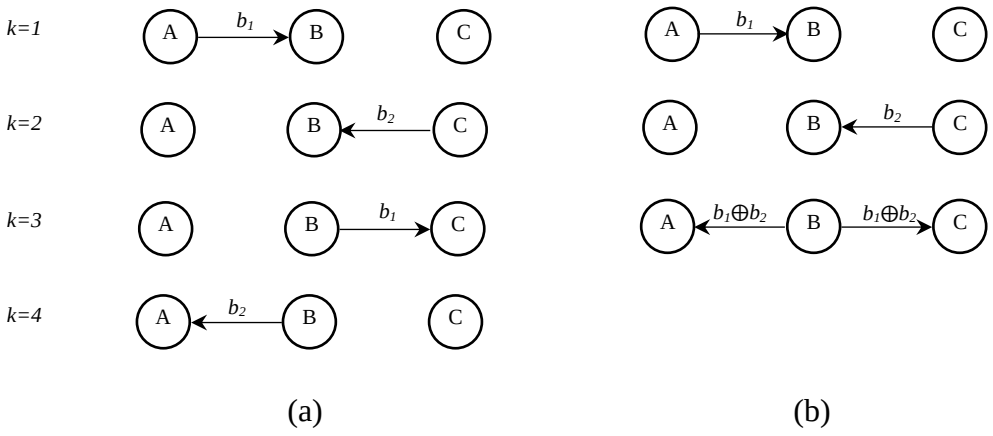


Figure 2.2: A transmission scenario in (a) store-and-forward and (b) NC wireless communication systems

Figure 2.2 (a) shows that a traditional routing scheme takes a total of k = 4 time-slots to complete the exchange procedure. With the help of NC, the transmission time can be reduced to three time-slots, as shown in Figure 2.2 (b). Hence, in comparison with the TR scheme, NC can save one time slot, and hence save transmission bandwidth.

Furthermore, NC can offer significant benefits by saving transmission energy when deployed [32]. The first implementation of NC in a real-world wireless environment was presented in [2] using an opportunistic approach to NC called COPE, where each node intercepts data on the medium, and detects coding opportunities after learning the status of the neighbouring nodes. This is then implemented in real-world wired networks in [26, 27]. The scenario presented in Figure 2.2 can also be considered as a model of a satellite communication system, where A and C correspond to ground stations that communicate with each other through node B, which corresponds to the satellite. This concept was first introduced in [33], before it was fully developed by [1]; in the latter the term "network coding" was coined.

2.3 Synchronous and Asynchronous Network Coding

In the early stages of the development of NC, the maximum network capacity achieved by NC implicitly assumed that packets of different flows are well synchronised. In synchronous NC, a coding node holds its coding operation until all required packets from different flows arrive. However, since real-world networks present stochastic packet arrivals and unpredictable delays, the size of the coding node buffers may increase to infinity [34]. This led to the proposal of the asynchronous NC approach, where the node calculates the probability that the packets it is waiting to combine are decoded by the receiver. If the probability is greater than a certain threshold, the node sends a packet combination, otherwise the packet is sent without coding. Additionally, a coding node may transit un-coded packets if no coding opportunities occur within a predefined time or pre-set conditions. This concept was fist implemented and evaluated by [2, 3] and attracted substantial research efforts.

2.4 Linear Network Coding (LNC)

The encoding functions f_e for $e \in E$ can be very complicated, and the role of NC is to design a function which ensures the reception of coded blocks that are represented by a set of equations that have a unique solution at the receivers.

Li et al. [17] concluded that a linear encoding process is sufficient to achieve the max-flow bound in a multicast network, between a source and each receiver; a detailed algebraic description of such a function in NC is presented in [23]. The use of linear coding makes the coding and decoding processes faster, and easier to implement in practice [35].

Let $b_1, b_2, \ldots b_k \in \mathbb{F}_q$ be the message packets arrive at the intermediate node W_1.

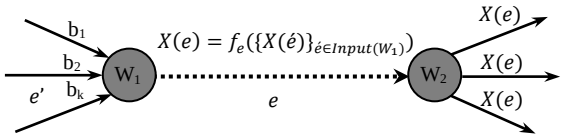


Figure 2.3: A intermediate link employing LNC

The transmission on the link is assumed to be synchronized in such a way that several packets are present at W_1 at the same time. The transmitted packets $X(e)$ denote the symbol that is being sent on link $e \in E$, and are related to the received packets by the linear equation, which is expressed as a combination of the packets that enter at node W_1:

$$X(e) = f_e(\{X(é)\}_{é \in Input(W_1)}) = L_w . X(é)$$

where L_w is the local transfer matrix at W_1. Each outgoing packet $X(e)$ is a linear combination of the incoming packets at the edge $é$ entering node W_1. The intermediate node W_2 relays the coded blocks $X(e)$ to its outgoing ports toward the

sink nodes $t_i \in T \subset V$, which should be able to retrieve the original packets by solving the set of equations

$$\{f_e(b_1, b_2, \dots b_k) = X(e)\}_{e \in input\,(t_i)}.$$

Since only linear operations are performed in the network, every packet transmitted along an edge is a linear combination of the source packets, $b_1, b_2, \dots b_k$. That is, for every node $v \in V$, X(e) can be given by

$$X_v(\acute{e}) = G_v \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}$$

where G_v is the global transfer matrix at v. In order for a sink node $t \in V$ to recover the source packets, it is necessary for G_t at t to have a rank k [26], since only then does G_t have an inverse G_t^{-1} satisfying $G_t^{-1} G_t = I_k$, where I_k is the identity matrix.

The result of encoding a set of packets that satisfies the coding conditions is called a *coded block*. The encoding function coefficients can be chosen in a deterministic or random way, highlighted in the following sections.

2.4.1 Deterministic linear network coding

When the data segment blocks are about to transmit, the source S chooses a set of coding coefficients C to produce a linear combination of the original data blocks. In deterministic NC [22, 36], C is chosen according to a predefined set of coefficients, nodes use the same encoding function at each run, and the encoding coefficients are known to every node. Hence, a full knowledge of the network topology is required, and nodes should be informed about the encoding design; this eliminates the need for overheads, and the operational cost of decoding will be low [37]. However, packets in practice are subject to link and node losses, and centralised knowledge of the network

topology is difficult to obtain. These constraints are overcome by random NC addressed in the following section.

2.4.2 Random linear network coding

In random NC [20, 38, 39], coding coefficients are not carefully designed, but simply chosen at random from a finite field. Hence, no previous knowledge of the network topology is required, which makes it suitable for large and dynamic networks. This property provides the flexibility to cope with arbitrary network topologies, and the ability to be applied in wireless networks [3, 40], P2P networks [27, 41], network security [42] and optical networks [43]. However, this randomized method requires a certain amount of overhead, such as the calculation of coding coefficients which are attached and transmitted in each coded block. Moreover, the higher operational cost of such a random coding scheme cannot be avoided, since the receivers have to solve a new system of equations for each transmission [37]. It can be concluded that random NC is resistant to packet loss, delays, different link capacities and variations in network topology [44].

2.4.3 Full-rank Vandermonde Matrix

As already mentioned, coding coefficients can be randomly generated and embedded in each coded block header [26]; this makes the approach suitable for large and dynamic networks. An alternative approach is for the sender and receiver to agree on one deterministic invertible encoding matrix, **V**. This method requires prior knowledge of the network topology, but it reduces the operational cost of the decoding process, as the receivers solve the same system of equations for each transmission. In Chapter 4, a ($k \times k$) Vandermonde matrix [45] is used, where $\alpha_i \neq \alpha_j$ for $i \neq j$ to ensure $\det(\mathbf{V}) \neq 0$:

$$V = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_k \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \dots & \alpha_k^{k-1} \end{pmatrix} \tag{2.1}$$

The row elements of V ensure a non-linear relationship between the encoded blocks, which allows the receiver to perform a decoding operation once *k* blocks have been received.

Figure 2.4 gives an illustrative example of a source *S*, multicasting blocks B_1 and B_2 to both receivers R_1 and R_2. Although both nodes W_1 and W_2 can serve node R_1, they may end up transmitting the same block B_2 to node R_1, since there is no connection between W_1 and W_2. In this case, the bandwidth of W_1 is wasted. Similarly, W_1 may transmit block B_1 to node R_2, which has already received B_1 through S_1.

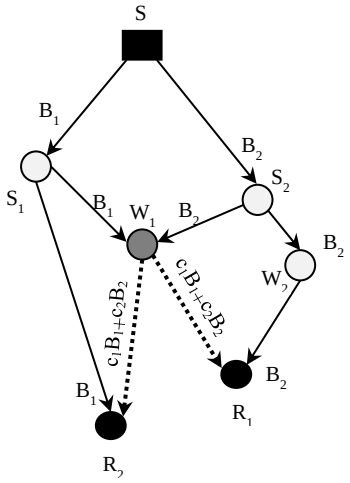


Figure 2.4: An example of the distribution of two blocks B1 and B2 with NC [44]

With NC, however, W_1 can transmit a coded block $Y=c_1B_1+c_2B_2$, with randomly chosen coefficients c_1 and c_2, to both R_1 and R_2. In this case R_1 can retrieve the original blocks B_1 and B_2 by solving the following set of simultaneous equations:

$$\begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ c_1 & c_2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} B_2 \\ Y \end{bmatrix} \quad (2.2)$$

The computational complexity of NC escalates with an increasing number of blocks; to reduce the effect of this, blocks in a file can be divided into multiple generations, with NC only being applied within the same generation [26].

2.5 Network Coding in Computer Networks

2.5.1 Network coding based error control techniques

The use of NC has been explored extensively in order to exploit its usefulness in improving multicast capacity. However, NC is still emerging from the fields of mathematics and information theory, therefore its practical application in real-world data communications systems has not been fully explored in existing research work. Usually, the performance of communication systems employing NC is investigated under the assumption that packets travel in error-free networks [1, 17]. Since transmission errors are unavoidable, communication networks adopt error control techniques to ensure reliable packet delivery. Packets which are received incorrectly must be retransmitted according to one of the most commonly used Automatic Repeat reQuest (ARQ) protocols: Stop-and-Wait (SW), Go BACK N (GBN) and Selective Repeat (SR). These protocols are based on the same mechanisms: the sender starts a countdown timer whenever it transmits a packet; the receiver transmits an acknowledgment for every correct reception; and the sender assumes that a packet has been lost when its acknowledgment does not arrive before the timer expires [46, 47].

The benefits of NC, as described in Chapters 4, 5 and 6, can be combined with ARQ schemes to improve network performance. Employing NC in a network allows the

next-hop destination to send one feedback message (ACK signal) for either *k* linearly coded packets of size *kn*, or one combination of size *n*, depending on the coding scheme utilised, as depicted in Figure 2.5. This yields better network performance in terms of throughput, transmission bandwidth and energy consumption.

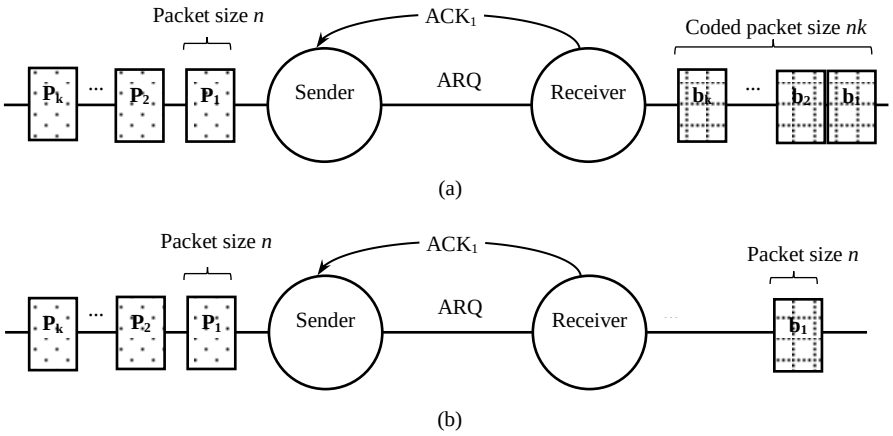


Figure 2.5: NC-based scenario where one ACK signal is sufficient for (a) coded packets of size *nk*, (b) coded packets of size *n*

Network coding with feedback was first attempted by Fragouli et al. [6]; the authors present a number of simple observations that indicate the possible benefits of using feedback in the context of NC. In [48], the authors analyse the reliability gain of NC for reliable multicasting in wireless networks; they show that NC significantly reduces the number of retransmissions in error-prone networks compared with an end-to-end ARQ scheme.

As a natural extension of ARQ schemes, [49] proposes a new coding and queuing management algorithm for communication networks, which employs linear NC. This scheme combines the benefits of NC and ARQ, enabling the sender to control its queued packets upon receiving acknowledgments based on the receivers' states of knowledge. In a single-hop wireless network [50], a hybrid ARQ-random NC framework for real-time media broadcast is proposed, which aims to minimise the packet erasure rate in order to enhance the media quality at receivers. Within the

same context [51], some NC schemes have been proposed to reduce the number of broadcast transmissions from the sender to multiple receivers, compared to ARQ schemes. This is carried out by maintaining and then XORing a list of *N* lost packets at the sender, before transmitting them to all receivers. A receiver is able to recover its lost packet by XORing the successfully-received combined packet with an appropriate set of previous correctly received packets. By integrating NC and ARQ schemes in [52], the end-to-end delay performance of reliable communications over a lossy wireless network was studied, based on the model presented in [48].

The steady-state throughput of general NC nodes is studied in [53]. In this study, a NC node with *H* incoming links perform coding, and transmit the coded packets via one outgoing port, through Stop-and-Wait ARQ protected links. The authors assume that the system operates in a synchronous manner, and that the incoming node links are provided with a FIFO queue to store packets, which are assumed to be ready to send. The authors conclude that the throughput of a NC node decreases as the number of incoming links increases and hence an increasing buffer size of the involved links is suggested. In the context of the three types of ARQ schemes, the authors in [54] investigated the maximum achievable throughput of the Butterfly networks. Figure 2.6 shows that in an error-free environment, the system attains the maximum possible throughput that the ARQ scheme can achieve. The work presented in this thesis, however, compares the performance of the three types of ARQ with and without employing NC. This makes the effect of NC improvement to be more evident.

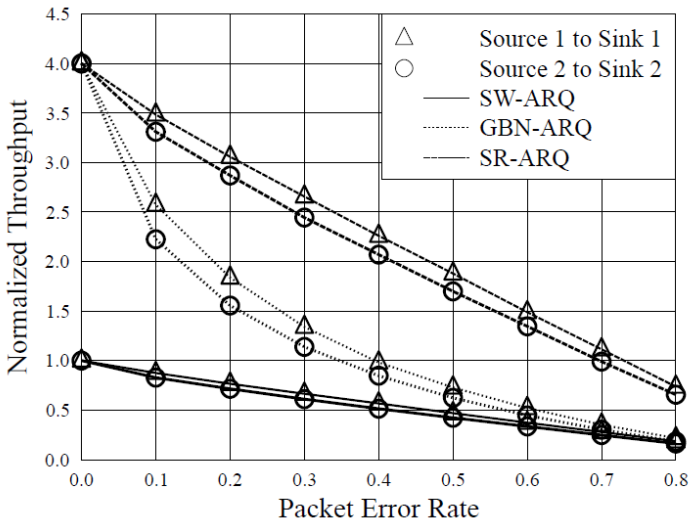


Figure 2.6: Maximum throughput versus PER of the butterfly network [54]

The authors in [55] present an NC-based cooperative ARQ scheme, which improves the network's bandwidth by minimizing the number of total transmissions, while significantly reducing the average time to transmit data packets. Among ARQ schemes, the Stop-and-Wait protocol is the most suitable one that can be used in half-duplex underwater networks [56]. However, the low bandwidth, long propagation delays and high error probability makes other ARQ techniques unsuitable for such environments [57]. The authors in [57] propose an error recovery scheme that carefully couples NC and multiple paths in underwater sensor networks. A number of downstream neighbour nodes are utilized to improve the efficiency of NC through adjusting the routing paths and the amount of redundancy on each node.

2.5.2 Network coding based queuing theory

The use of NC can achieve a significant capacity gain when the incoming data are properly synchronised. Hence, it is important to investigate the impact of employing synchronised NC on the behaviour of an encoding node. If synchronized NC is adopted over an intermediate encoding node with K incoming links, coding will not be performed unless at least one packet on each link is present. This means that

buffers attached to incoming links with high data rates will suffer from queuing problems, which leads to their buffer size approaching infinity [34]. Although the performance of communication systems employing asynchronous NC has been the subject of substantial interest in recent years, very little work exists that considers the queuing behaviour when synchronous NC is adopted. Figure 2.7 shows a general queuing system with two incoming links and input buffers.

In Chapters 7 and 8, an intermediate node with *K* input links is considered, as depicted in Figure 2.7.

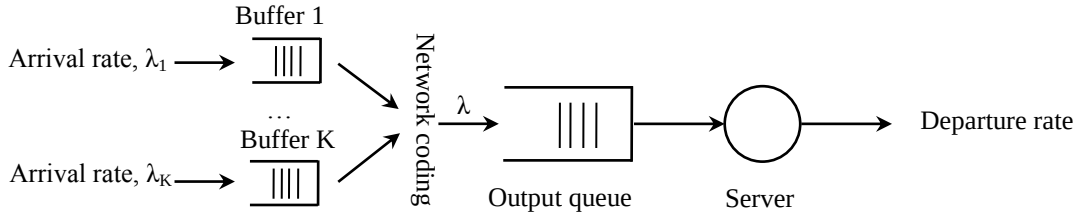


Figure 2.7: A typical queuing system with K incoming links

Each link is connected to a source that generates packets independently according to a Poisson process with rate λ_K. In the case of traditional routing, the total incoming traffic rate with two sources at the intermediate node queue is $\lambda = \lambda_1 + \lambda_2$. The employment of a synchronous NC scheme forces packets to wait for the arrival of other packets with which they will be combined, incurring a potential synchronization delay. In order to tackle this problem, several strategies have been proposed based on the concept of sending un-coded packets should packets of other sources not arrive within a predefined time. This may cause transmission inefficiency, since un-coded packet streams can only communicate one packet at a time, whereas coded packet streams can communicate two or more packets during the same duration of transmission. Therefore, introducing asynchronous NC may offer an advantage in terms of increasing coding opportunities, as a coding node may

wait some time for a packet to be received from the other node. However, system performance may degrade if this waiting time is prolonged.

In the literature, various approaches have been taken to tackle the problem of synchronization, and to quantify the benefits of NC from the queuing perspective. The opportunistic NC (ONC) method has been proposed where the coding decision depends on certain conditions [2, 3]. Within the context of queuing analysis, an opportunistic NC approach has been proposed by [58] where the decision of coding depends on the buffer's state at a given node. In the developed queuing model, the decision to transmit un-coded packets depends on the number of packets in the buffers. The impact of this decision on the energy-delay trade-offs is analysed in [59]. The authors introduced energy-efficient transmission strategies based on the instantaneous queue contents for stable operation, and highlight the cross-layer optimization trade-offs between different measures of throughput and energy efficiency. ONC is also employed in [60] to solve the problem of synchronization where a use of a dynamic buffer allocation at the relay station is proposed. The authors proposed a discrete-time Markov queuing model for a wireless lossy butterfly network, where opportunistic NC scheduling protocol [2, 7, 58] is employed at the relay node. In [61], the authors model an energy-delay trade-off problem as a continuous Markov chain with exponential arrival rates. The method proposes a threshold for the buffer used to accommodate waiting packets – when this is exceeded, packets are transmitted un-coded. This work has been extended in [62], a study which captures the different functionality of NC, including packet classification, route processing, coding and transmission. It is assumed that the relay's buffers have unlimited capacity; therefore, loss of packets due to buffer overflow cannot occur. An NC aware queue management scheme [63] is employed

at intermediate nodes, to fully exploit the coding opportunities. The authors in [64] analysed the behaviour of a system employing ONC for the case of general arrival rates. The analysis studies the trade-off between the delay due to waiting for coding opportunities and the increased efficiency of spectrum access due to NC.

Through solving queuing systems for the buffer behaviour at a relay node, [65] derives closed-form expressions for the throughput and packet delay of NC in the slotted ALOHA protocol. In [66], the authors present a queuing model for RLC for time-division-duplex (TDD) channels in a bulk-service queuing system; numerical results for the mean number of packets in the queue are presented, and an optimal choice for the bulk-size that minimizes the mean number of packets in the queue is shown.

Presented in Figure 2.8, the model proposed by [34], concludes that Poisson distributed incoming packets results in coded packets asymptotically follow a Poisson distribution as shown in Figure 2.9. It also concludes that due to synchronization, the queue size of the encoding node will approach infinity, as shown in Figure 2.10. This observation has led to the proposal of an opportunistic scheduling strategy, Combined Opportunistic Scheduling and Encoding (COSE), in which a clearing stage is associated with an encoding stage to transmit unmatched packets to clear non-empty queues. This scheme has come to be regarded as an asynchronous NC approach, since some packets are transmitted without coding.

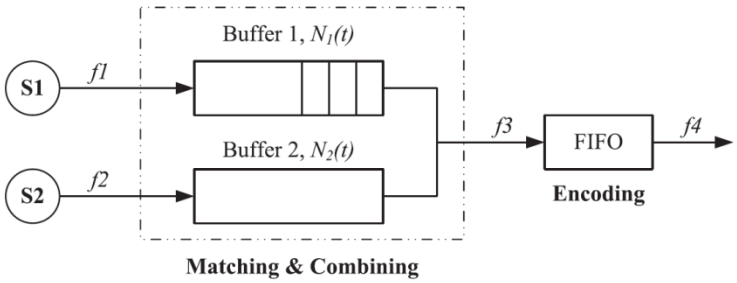


Figure 2.8: Queuing model of the encoding node with two inputs and one output [34]

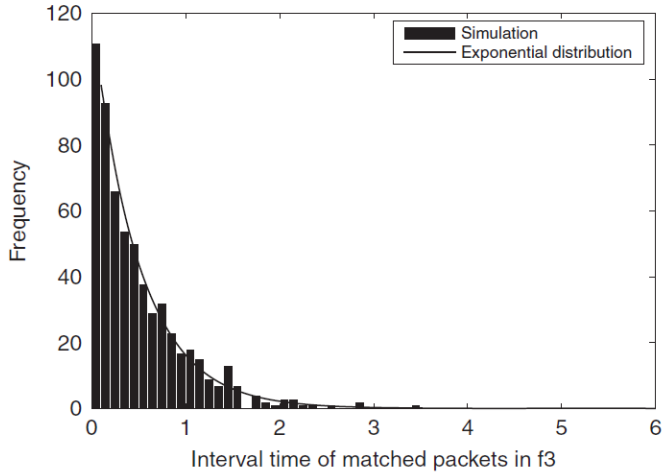


Figure 2.9: Histogram of the interval time in *f*3 [34]

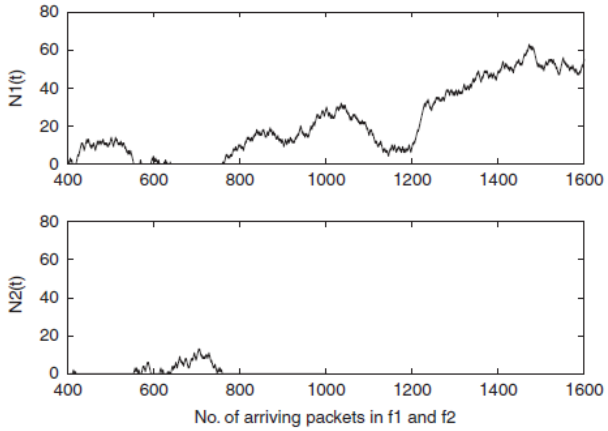


Figure 2.10: Queue length variation via time in f1 and f2 [34]

In [67], the additional delay at the intermediate nodes caused by synchronous and asynchronous partial coding is evaluated. In a wireless *Butterfly* network [7], the authors compare the classical routing with NC in delay-sensitive applications. Two queues in tandem were considered to observe the equilibrium queue-length build-up

of the second queue in the presence of stochastic arrivals and limited service rates. In the wireless broadcast medium, RLC is also studied in [8]. The authors present a queuing model for RLC and explore the delay performance of RLC that adapts to the stochastic arrivals of traffic at source node, which is analysed as a discrete-time bulk-service queue.

In further research, the authors in [68] built a generic queuing model under asynchronous NC, based on the Internet core scenario. They studied the stability of the system, and the maximum achievable coding gain in a stable system. In addition, a quantitative relation between different metrics such as packet delay, traffic arrival rate and service rate under general traffic arrival is described.

In contrast to previous work, **Chapter 7** extends the results obtained by [34] and introduces a new analytical model for an M/M/1 system, with two and three source nodes addressing one intermediate node, but without input buffers. The source nodes are assumed to hold their packets until all packets from other sources have their packets ready for coding simultaneously. The study aims to investigate the coded packet distribution prior to their entry to the intermediate node queue, and also their waiting times before they are served. The results show that the resulting coded packets follow a random general distribution. The M/M/1 queue has a Poisson distribution for arrivals and departures and has infinite buffer size with one server. If the queue has a finite buffer of K then it will be referred to as M/M/1/K. Later in Chapters 7 and 8, the terminologies of newly formed queues will be identified and defined accordingly.

Based on the analytical results obtained in Chapter 7, **Chapter 8** studies the behaviour of an M/D/1 queuing system under a synchronous NC-based scenario. The

study shows that the M/D/1 queuing system, in a NC-based system with two inputs, leads to a general input queuing system, which consequently leads to a G/D/1 queuing system. The M/D/1 queue is a single-server system with Poisson-distributed arrivals, with a mean value λ, and is a special case of the M/G/1 queue with a constant service time distribution μ. An example of the application of this model is a situation where two packets from different streams arrive at a router; here they are formed into a single packet with a fixed length - hence the service time is always the same. The characteristics of this queue are originated from the Pollaczek-Khintchin formula [69]. Since the service time of the studied system is constant, the variance is zero, i.e. $\sigma_2 = 0$. The statistical measures are thereby developed and listed below:

$$L_q = \rho^2/[2(1-\rho)] \tag{2.3}$$

$$W_q = \frac{L_q}{\lambda} = \frac{1}{2(\mu-\lambda)} - \frac{1}{2\mu} \tag{2.4}$$

where ρ is the utilization ratio, and is equal to λ/μ, and L_q and W_q are the queue length and the mean waiting time in the queue, respectively.

There have been several studies that investigate the G/D/1 system but out of the scope of NC. For example, the relationship between the probability generating function (pgf) of the buffer content, and the pgf of the packet delay, in a discrete-time G/D/1, is derived in [70]. The G/D/1 can be used to model heavy traffic networks using α-stable self-similar processes [71]. Based on an embedded Markov chain, simple recursive formulae of the queue length and waiting time for the case of discrete autoregressive arrivals can be derived [72]. Wireless nodes are modelled as a G/D/1 queue for capacity planning using a Cross-Layer approach [73]. In cell-based telecommunication systems, with a constant cell transmission time, such as ATM, G/D/1 has potential applications since it allows arbitrary inter-arrival time

distributions, whilst maintaining a constant cell service time [74]. The authors in [75] propose methods to estimate the parameters of arrival processes of a G/D/1 queuing system, based only on observed departures from the system. The proposed estimation algorithm can be further utilized in a network of an arbitrary number of G/D/1 queues in sequence.

2.6 Conclusions

In this chapter, an overview of NC is provided as background knowledge for aiding the understanding of the following chapters. The NC technique proposes that an intermediate node can forward any function of incoming packets, not only limited to making copies of these packets. Some examples of the general concept of NC are stated in order to elaborate the advantages of NC over the TR scheme in both wired and wireless environments. Additionally, related work concerning the employment of NC along with error control schemes and queuing theory is also presented. The following chapter introduces three flow error control mechanisms, namely Stop-and-Wait (SW), Go-back-N (GBN) and Selective Repeat (SR); a simulation-based study of SW ARQ is also described.

Chapter 3

Computer Network Flow Control

3.1 Introduction

In the previous chapter, a brief introduction to the concept of network coding (NC) was provided. In this chapter, three flow error control mechanisms, namely Stop-and-Wait (SW), Go-back-N (GBN) and Selective Repeat (SR) are introduced. Communication and computer networks adopt a flow control mechanism to maintain the packet transmission reliability between the sender and receiver; in addition this ensures that the receiver is not overwhelmed by the sender and that the network does not suffer from the effect of bottlenecks.

Most previous studies on the benefits of NC have been carried out under the assumption that the packets travel over error free channels [17, 23, 26, 76]; however, errors may be introduced during transmission from a source to a receiver since communication channels are subject to channel noise [4, 77]. The transmitter and receiver must adopt a set of rules to ensure the reliable transfer of messages. To accomplish this reliable transfer, the receiver performs an error detection procedure and informs the source that an error has been detected, and hence that another copy of the corrupted packet should to be retransmitted. The process of error detection and packet retransmission is called Automatic Repeat reQuest (ARQ). ARQ protocols are one of the flow control mechanisms used to ensure high-reliability communication in noisy channels [4]. An ARQ protocol builds an algorithm to allow the sender to have a full knowledge about the transmitted packets at the receiver.

This chapter seeks to give an overview of the different types of ARQ protocol, and examines the overall throughput performance of a Stop-and-Wait (SW) ARQ scheme experiencing different Packet Error Rates (PERs), using both simulation and numerical approaches. Furthermore, it provides a brief introduction to the Selective Repeat (SR) ARQ protocol and its requirements, and this is to be utilised in Chapters 5 and 6 when examining the benefits of Network Coding (NC) schemes employing this protocol.

3.2 Data Link Flow Error Control

In computer networks, the data link layer regulates the amount of data that the sender entity can transmit before overwhelming the receiver entity, using a given flow control technique [78-80]. The receiving entity has limited processing speed and queue capacity; therefore - in the absence of flow control - the receiver's queue may gradually build up and overflow while processing the old data. Additionally, a typical flow control method also involves a process for informing the sender of any packet loss or damage during transmission, thereby managing to perform a retransmission, or a data correction at the receiver. In the following sections, a brief overview of three ARQ transmission schemes is first considered, and then a range of performance results related to the SW ARQ protocol is shown.

3.2.1 Stop-and-Wait ARQ

The Stop-and-Wait (SW) ARQ data transfer protocol is the simplest type of ARQ scheme [81]. Fundamentally, packets are sent sequentially, each containing different data, and each packet has to be received correctly before the system initiates transmission of the next packet. In a communication network employing SW ARQ, the transmitter stops after sending one packet, and waits for its delivery status. If the

packet reaches the receiver correctly, this sends an Acknowledgment (ACK) signal; otherwise, a negative acknowledgment (NAK) is sent back to the transmitter upon detecting an error using an error-detection technique. The transmitter commences transmission of the next packet upon receiving an ACK signal, but it fetches and resends the same packet from the forward queue when a NAK signal is received. The transmitter usually implements a timer whose expiry, without any response from the receiver, triggers a retransmission of the unacknowledged packet. Figure 3.1 shows a source, A, transmitting only a single packet to a destination, B, and then waiting for its acknowledgment. Each packet takes one propagation time delay (T_p) to reach the receiver B, in addition to the time required to emit all packet bits onto the transmission link (T_r); this means that the transmitter idle time is equal to $2T_p + T_r$. Correctly received packets are aggregated by B in the same order in which they were sent.

Figure 3.1 depicts a general SW ARQ packet transmission process with three main scenarios. In the first, $Packet_0$ transmitted by A is delivered successfully to B, which in turn, sends ACK_0 accordingly to A. In the second scenario, $Packet_1$ is sent to B but is lost. The sender A employs a countdown timer, which triggers a resend of the same packet upon expiry of the time-out interval. In the third scenario, A has to send the same packet again to B due to loss of the ACK, even if the sent packet was correctly received by B.

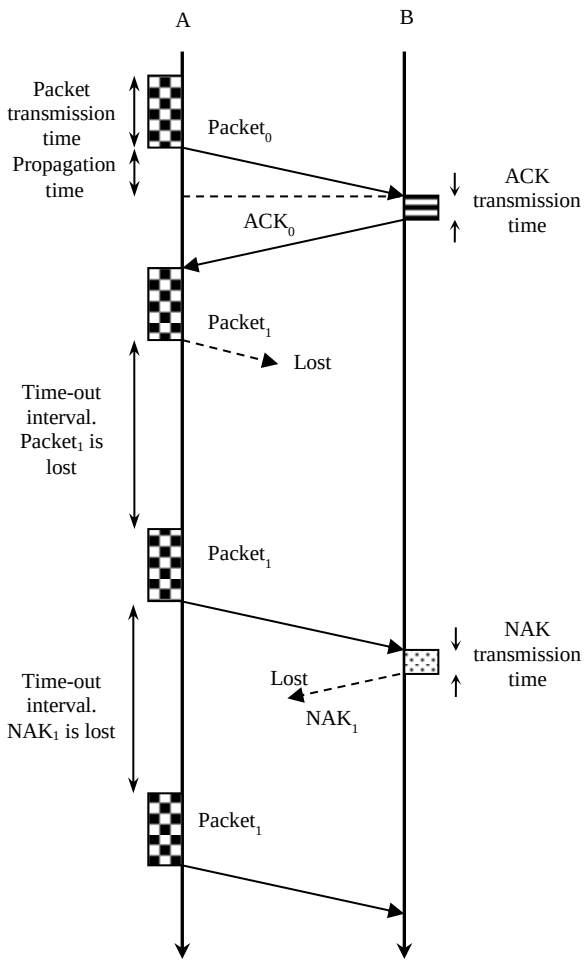


Figure 3.1: Stop-and-Wait ARQ protocol [82]

The time between two successive generated packets, T_{gen}, is called the 'intergeneration time'. The propagation time delay governs the throughput performance when it is larger than the intergeneration time. This is because the sender will not send any outstanding queued packet until it receives an ACK/NAK signal from the receiver after a delay of approximately $2T_p$. Figure 3.2 shows the general packet behaviour when transmission is error-free.

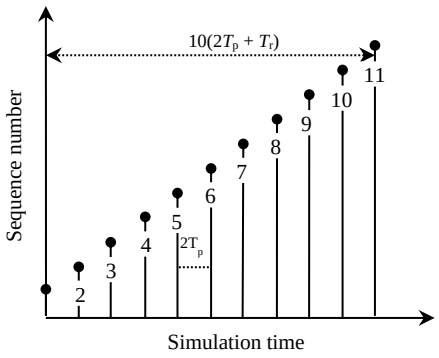


Figure 3.2: Stop-and-Wait protocol in error-free transmission

Figure 3.3 depicts how the SW ARQ method operates when packets are lost in the presence of errors. Packet numbers four and six are found to have been lost or corrupted during the transmission, which forces the sender to suspend the sending of packets number five and seven until it is ascertained that all previous packets have been submitted correctly and in sequence. If the frame or ACK/NAK signal is lost, the transmitter will time-out, and will re-send the previously sent packet.

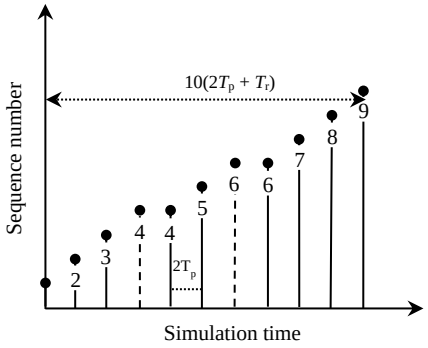


Figure 3.3: Stop-and-Wait protocol in an erroneous transmission

In the previous examples, the packets are assumed to be ready for transmission. Therefore, the time between packets in both Figures 3.2 and 3.3 is equal. However, later in Figure 3.8, the time between transmitted and retransmitted packets is not equal since some packets require some time to be generated and ready for transmission.

To mathematically analyse the SW ARQ throughput performance, two network stations are considered with a propagation delay T_p. For a packet size, *PS*, including headers and data fields, the time needed to transmit the packet into a transmission channel, of capacity *C* bps, is $T_r = PS/C$.

If the transmitter begins sending packets at time $t = 0$, then the last bit enters the channel at time T_r; the packet therefore reaches the receiver at T_p+T_r. The acknowledgment signal is usually assumed to be of a negligible size and the receiver sends it back immediately after receiving the transmitted packet. The ACK signal thus reaches the transmitter after a total transmission time of $T_{tot} = 2T_p + T_r$. This means that the efficiency of the SW ACK protocol will degrade as the ratio of the propagation time to transmission time becomes large, and this categorizes the maximum utilization performance, *U*, of the communication system studied [77]:

$$U = 1/(2\alpha + 1) \tag{3.1}$$

where α is the ratio T_p/ T_r. In practice, communication links have a non-zero Bit Error Rate (BER), hence, in order to transmit a packet successfully, an average N_r transmission attempts will be required. The probability that a transmitted packet of length *n* contains an error PER is given by [47]:

$$\text{PER} = 1 - (1 - \text{BER})^n \tag{3.2}$$

Therefore the probability of uncorrupted reception of a packet will be 1 – PER, hence $N_r = 1/ (1 - \text{PER})$. Thus, the utilization can be obtained by:

$$U = \frac{1 - PER}{2\alpha + 1} \tag{3.3}$$

In a noise-free channel, the BER is zero, and hence PER=0. Thus, link utilization takes the value given by (3.1). The throughput, η, of a SW ARQ scheme is given by

η = U*C [46].

In an SW ARQ scheme, if the intergeneration time between packets is smaller than ($2T_p + T_r$), it creates a queuing problem at the transmitter queue, which is analysed in the subsequent sections. For simplicity, the term 'packet' will - from this point onwards - be used to denote the resultant transmitted network entities rather than frame.

3.2.1.1 Stop-and-wait protocol architecture in SimEvents

Simulation analysis is a flexible method for evaluating and observing the behaviour of real-world communication networks over time; it allows modification of the simulated network attributes, in order to examine network performance under certain conditions. SimEvents® [10, 12, 83, 84] is a relatively recent Discrete Event Simulation (DES) tool [85, 86], which helps researchers to design models and estimate system performance using a powerful Graphical User Interface (GUI) console. It offers a platform for designing various computer and communication models, and for evaluating different network metrics, including congestion, throughput and end-to-end delay, using the supplied statistical aggregation blocks. SimEvents appears as a component library [9] for Simulink® in MATLAB, and uses items of interest called entities, which can be configured, via their attributes, to represent different network data packets and control signals. The library encompasses the fundamental elements that constitute basic and complex networks; including queues, switches, servers, timers, and sink blocks. It also offers blocks which allow the writing of user-defined programs, and collaborates with the main MATLAB command window.

a) Network deployment and planning in SimEvents

The SimEvents system incorporates discrete-event system modelling in the time-based framework of MATLAB Simulink, and is therefore suited to the modelling of continuous-time and periodic discrete-time communication systems. It is generally possible to construct a discrete-event system using SimEvents, by adding a variety of discrete items of interest suitable for producing and processing entities, such as generators, queues, and servers, from the SimEvents block library. A Simulink model can contain a purely discrete-event system with no time-based components when modelling event-based processes. However, using special gateway blocks, hybrid systems can be constructed, because one or more discrete-event systems can coexist with time-based systems in a Simulink model. Entities can pass through a network of queues, servers, gates, and switches during a simulation, and can also carry data that can be stored in one or more defined attributes of an entity. The values of attributes can be read or changed during the simulation. Therefore, SimEvents offers a convenient modelling method to represent a computer network using entity generators to generate packets and assign payloads using entity attributes.

b) Block libraries used for SW ARQ simulation

The necessary blocks in SimEvents for SW ARQ modelling include *inter alia*:

1. ***Time-based Entity Generator***: this block is used to generate entities in the form of packets. The intergeneration time is chosen in this chapter according to a constant and exponential distribution.

2. ***Set Attribute***: data and sequence number attributes are assigned to entities using this block. These attributes can be fetched back using the *get attribute* block.

3. ***Entity Departure Counter***: this block is used to compute the number of entities which have departed the source; it writes this number to a packet *attribute* field.

4. ***FIFO Queue***: the transmitter requires a queue of capacity 1 in the scheme modelled. However, this block can handle an infinite number of packets in a first-in, first-out (FIFO) fashion. The queue retains the entity if the OUT port is blocked, and attempts to output an entity if the path is available. This block can provide several metrics, such as: an average number of entities in the queue over time, a sample mean of the waiting times in the queue for all entities that have departed, *etc*.

5. ***Enabled Gate***: the system simulated uses this block to control the packet flow through its input ***en***. The gate opens when ***en*** is positive and closes when it is zero or negative.

6. ***Replicate***: for the sake of simulation, packets are required to follow two or more paths and therefore this block is used to output a specified number of copies of the arriving entity.

7. ***Path Combiner***: this block is used when the merging of entity or packet paths is necessary, for example in merging the paths of the new and retransmitted packets. It accepts packets through any input port and outputs them through a single output port.

8. ***Switch***: based on a specified switching criterion, this block is able to guide packets according to certain attributes, which can be specified by the user.

9. ***Server***: this block is used to serve one or more entities, or to delay them for a defined period of time.

10. ***Function-Call Subsystem***: function-call signals can conditionally invoke this block to execute another user-designed function within the block.

11. ***Data Store Memory***: this block is used to reserve a shared data store, which is usable by the *Data Store Read* and *Write* block which has the same specified data store name.

c) Communication Model

The simulated model used in this study is depicted in Figure 3.4; it consists of a packet generator, transmitter, receiver and the forward and reverse channels. The packet generator block is able to send packets at different data rates according to a constant, uniform, exponential and user-defined intergeneration time. Unless stated otherwise, all time intervals are expressed in arbitrary time units. The forward and reverse channels employ a *server,* which imposes a specified propagation time delay. Errors are injected using an attribute block that adds an ACK or NAK field to the passing packets. The error probability is chosen to vary according to a *Bernoulli* distribution; the receiver checks this field and accepts or rejects packets upon fetching an ACK or NAK accordingly. When an ACK signal reaches the transmitter, it triggers an output *switch* to empty its queue, which stores a copy of the previously sent packet, allowing the next packet to be advanced into it. However, if a NAK signal is received, the output *switch* passes the stored packet to be sent again into the transmission link.

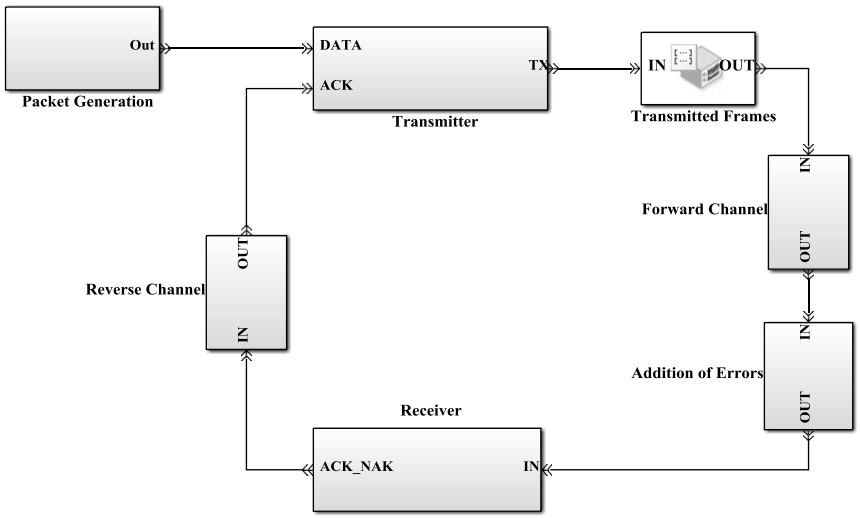


Figure 3.4: SimEvents-based point-to-point communication network

The transmitter components are shown in Figure 3.5; the key element of the operation is a control unit, which oversees the packets flow during transmission.

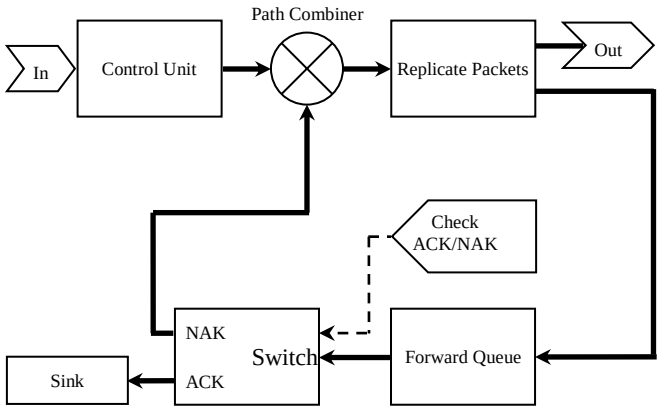


Figure 3.5: The transmitter main blocks

The advanced packet from the control unit is replicated into two copies, one of which is stored in the forward queue, and the other of which is pushed onto the transmission line. The acknowledgment signals generated at the receiver update the transmitter regarding the packet delivery status. This means that any packet being advanced to the transmission line or suspended is subject to the previously sent packet status.

Due to the fact that there is no ACK/NAK signal for the first packet, it is necessary to construct a *control unit* to tackle this problem. The control unit subsystem shown in Figure 3.6 consists of two paths. The lower path advances only the first packet to the transmission line, so that it can generate an ACK/NAK signal at the receiver side, which in turn can control the rest of packets. The upper path controls the remaining packets by reading the delivery status of the previously transmitted packet; this can be done by reserving a shared memory location using the *data store memory* block, which is accessible by the *data store read/write* block.

When a packet passes through the *enabled gate*, it calls a function that nullifies the *store memory* location using a *write store memory* block, which stops any subsequent packet from going through. A successfully transmitted packet writes a positive value in the memory location, indicating that a new packet is allowed to pass. However, in the case of unsuccessful transmission, the value of the memory location is kept at zero until a successful transmission of this packet occurs.

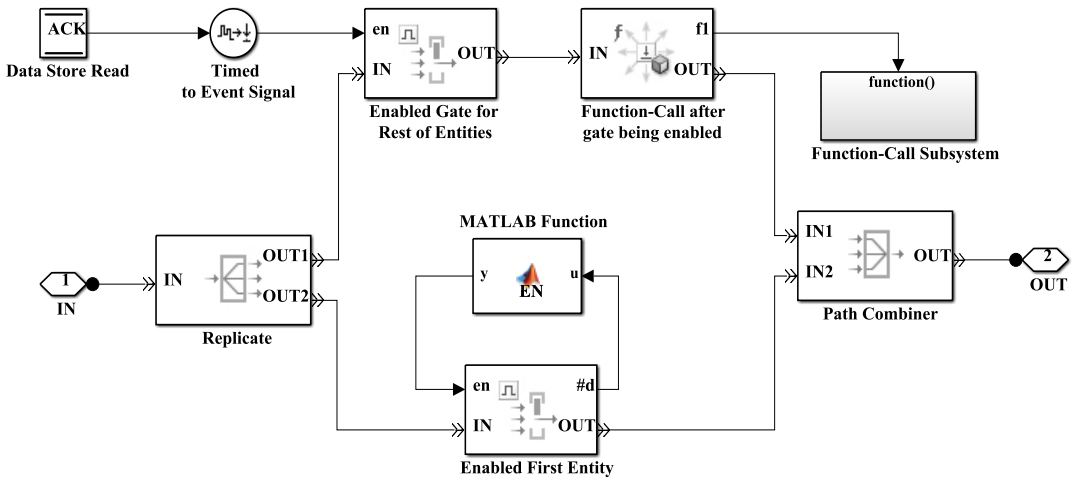


Figure 3.6: The transmitter control unit components

As mentioned previously, the packets can be generated according to several distributions. In Figure 3.7, packets are produced exponentially in an error-free environment, with a mean of one and propagation time T_p=0.015.

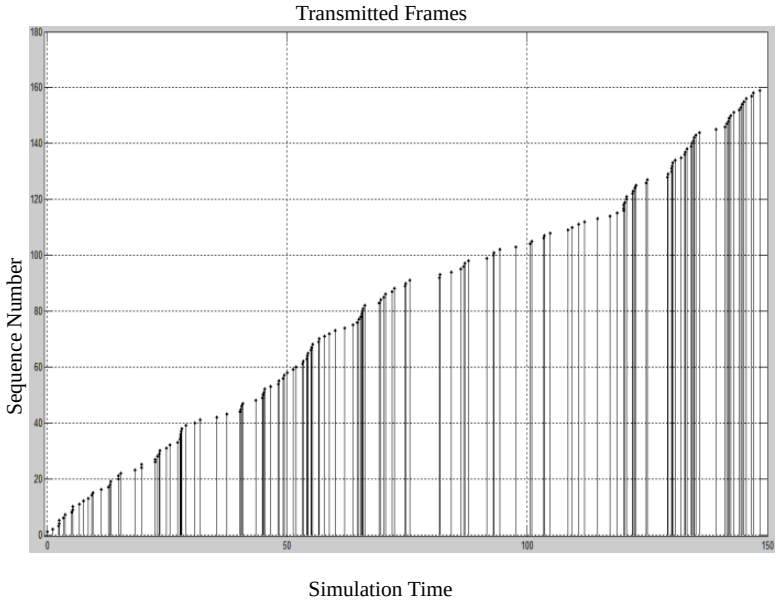


Figure 3.7: Exponential packets intergeneration

Another example is shown in Figure 3.8; this clarifies the behaviour of the packet transmission sequence presented in Figure 3.3. Where packets are generated with, for example, PER=0.1, T_p=1 and at a constant T_{gen}=3, it can be seen how erroneously received packets are sequentially retransmitted.

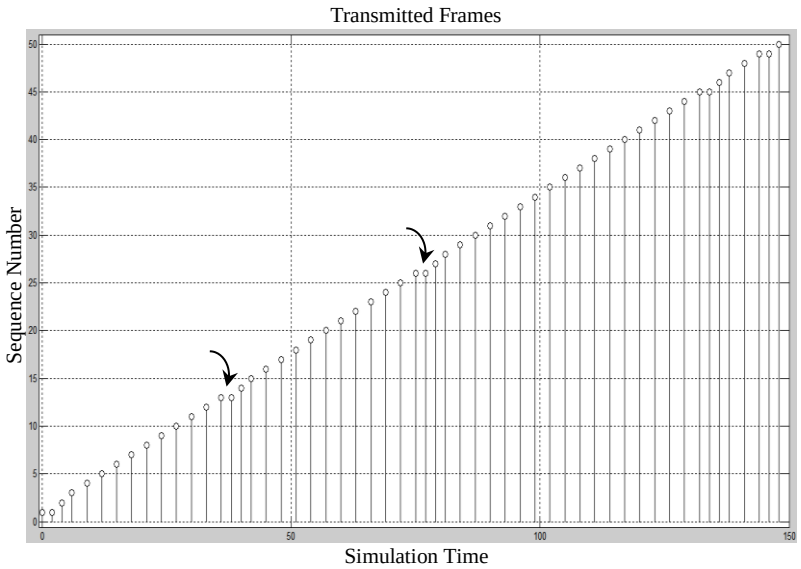


Figure 3.8: Constant packet generation with PER=0.1

d) Simulation and performance analysis

A. System Parameters

Packets are transmitted over a communication link based on the SW ARQ scheme. The transmitter and receiver operate with a 1Mbps channel capacity, which means that the transmission time needed to emit a 1000-bit packet is 1ms. Packets are generated at a constant data rate equal to 50 kbps, therefore the intergeneration time T_{gen} between two successive packets is chosen to be 0.02 seconds.

The propagation delay was chosen so that it demonstrates the performance of SW ARQ and verifies the constructed model. It represents the distance between the East and West Coasts of the United States - approximately 15 milliseconds - considering the distance and the propagation speed [80]. The model can be utilised in application with high propagation time delay such as Molecular and underwater communications [87, 88]. The PER in the forward channel was configured to vary according to a *Bernoulli* distribution model with error probability of 0.1. In the next section, the throughput performance in a typical point-to-point communication system is investigated; the parameters given above are adopted unless otherwise stated.

B. SW ARQ performance and results analysis

An SW ARQ system was implemented with a PER=0.6; Figure 3.9 depicts the achievable throughput for the system against the elapsed simulation time. Equation (3.3) states that the percentage of correctly received packets will be 40% when PER=0.6. Consequently, the achieved throughput when PER=0.6 will be equal to 40% of the maximum throughput of ~32 kbps i.e. 12.8 kbps, as shown in Figure 3.9. Implementing the same system parameters for different PER yielded the result shown in Figure 3.10. It can therefore be concluded that the throughput performance decreases gradually with any increase in the PER. This degradation comes from the

need to retransmit more corrupted packets and to prolong the transmitter idle time. In addition, the maximum throughput, 50 kbps (1000/ T_{gen}), has not been achieved since $T_{gen} < 2T_p+T_r$.

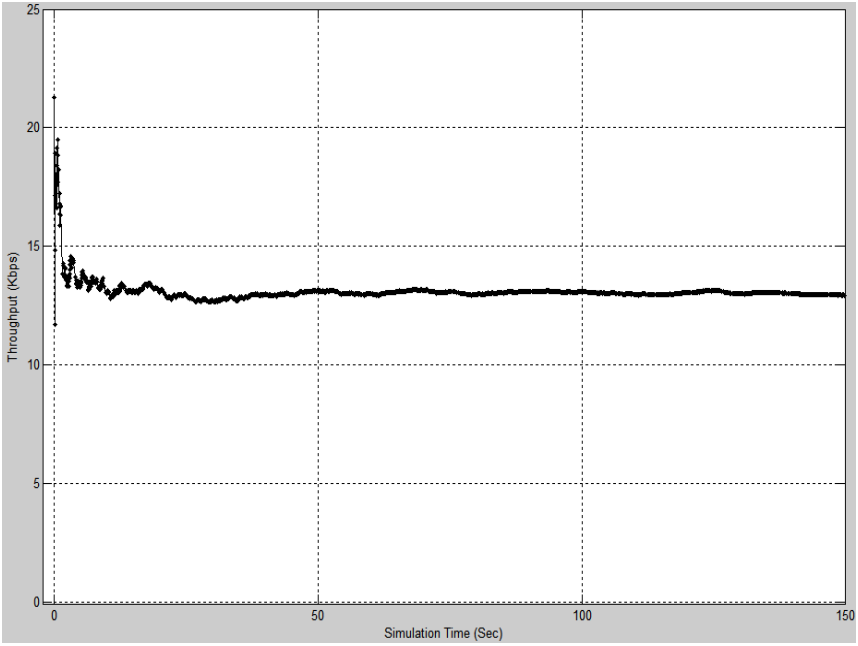


Figure 3.9: SW ARQ throughput when PER is 0.6

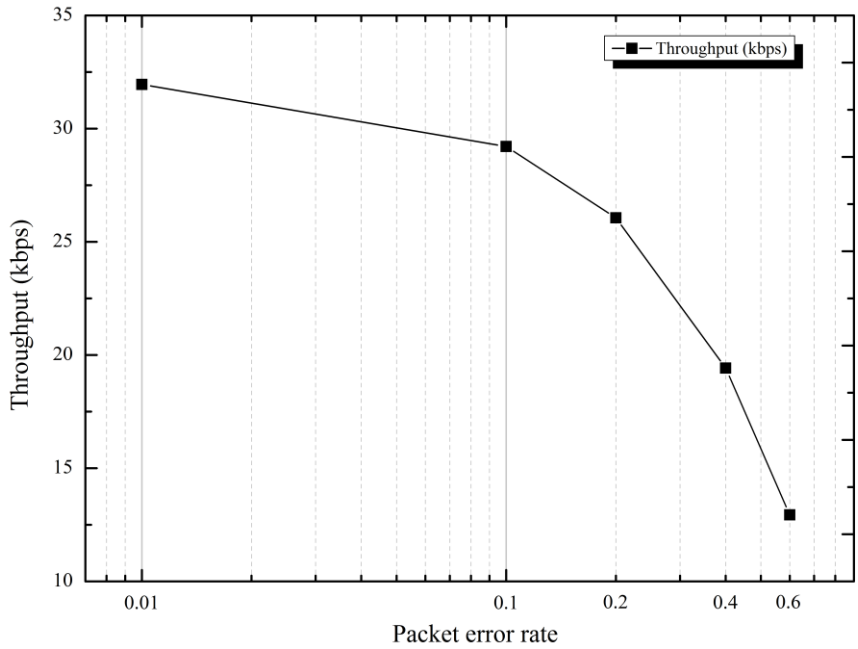


Figure 3.10: SW ARQ throughput as a function of PER

Figures 3.11 and 3.12 show the relationship between both the average queue length (which represents the average number of entities in the queue over time), and its

corresponding average packet waiting time, as a function of PER. It is clear that, with growing PER, packet retransmission increases, which stops new packets being advanced into the transmission link. Consequently, the average queue length increases with PER, as does packet waiting time.

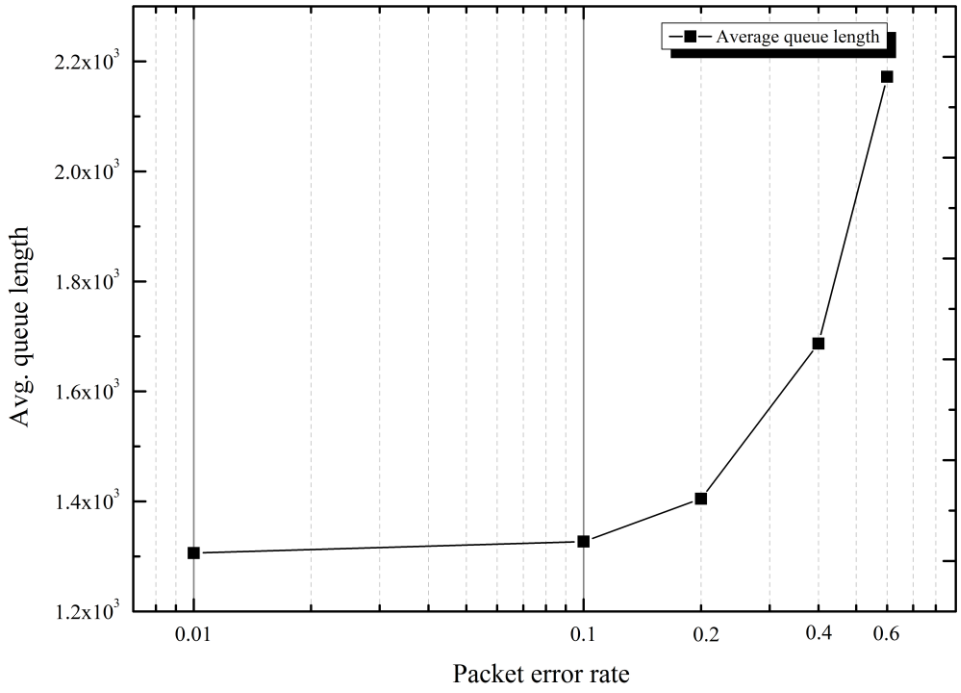


Figure 3.11: Average queue length as a function of PER

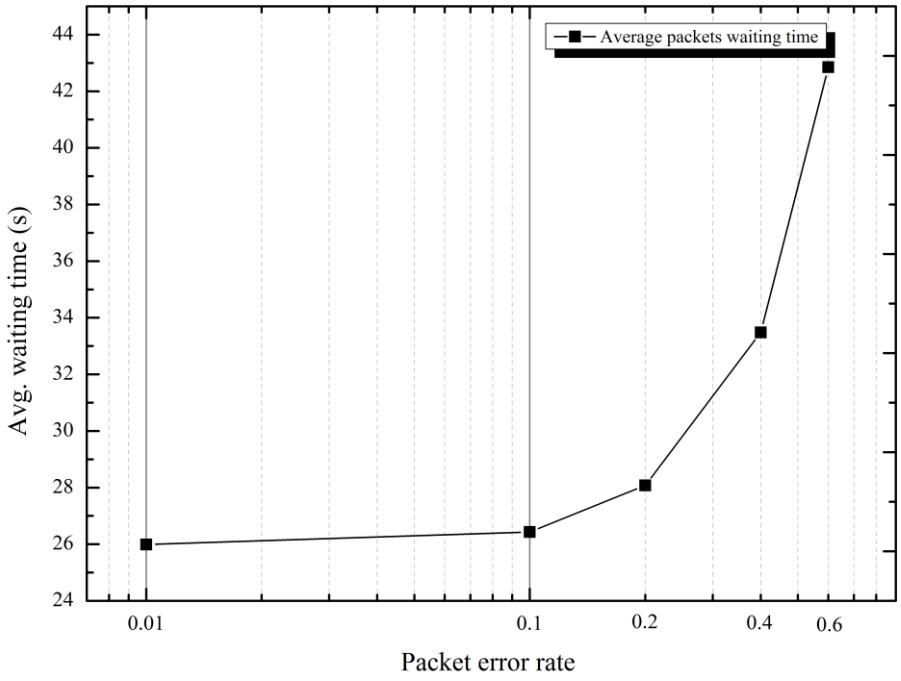


Figure 3.12: Average waiting time as a function of PER

To specify the required queue size within the transmitter at a specific data rate, the queue size measurement feature in SimEvents is utilized, which can give an average packet number stored in the queue at predefined data rates and propagation delays. When the intergeneration time is smaller than the time required between sending a packet and receiving its acknowledgment (i.e. $T_{gen} < 2T_p+T_r$), the number of outstanding packets in the queue will increase as more generated packets will arrive through the input queue before releasing new packets from the output. As shown in Figure 3.13, at a rate of ~32 kbps, the queue starts building up because at this point T_{gen} becomes smaller than $2T_p+T_r$.

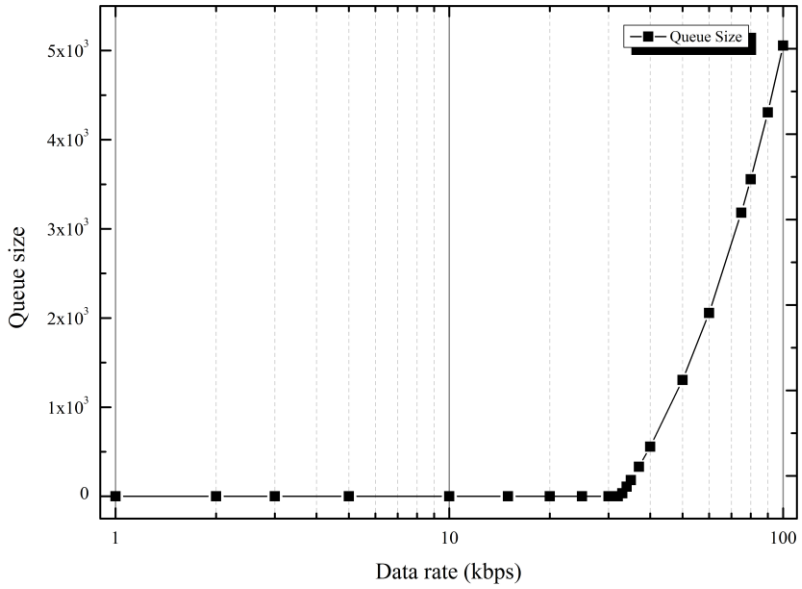


Figure 3.13: Queue size as a function of packets data rate

SimEvents has shown efficiency in simulating event-driven processes by determining resource requirements and identifying system performance. The benefit of using SimEvents for this purpose comes from the fact that it uses the MATLAB language, which is widely used among engineers, and in addition it can help in the realization of general-purpose models.

The advantage of SW ARQ is its simplicity and reliability. However, it becomes inefficient at high data rates, and in the presence of network propagation delays due to the time, during which the transmitter is idle. This necessitates the use of a continuous data transmission protocol, which is referred to as the 'sliding window' protocol, as discussed in the following section.

3.2.2 Sliding Window Flow Control

The efficiency of flow error control schemes can be greatly improved by allowing multiple packets to be in transit at the same time. The sender is allowed to send *W* packets to the receiver without waiting for any acknowledgment. The operation is

referred to as ‘sliding window flow control’, and has two main modes of operation; GBN and SR, as detailed in the following sections.

3.2.2.1 Go-Back-N (GBN)

SW ARQ is the simplest error control mechanism, but also the least efficient; this is because only one outstanding packet is filling the transmission pipeline at any one time during transmission. In GBN [82], up to *N* frames can be sent before receiving a request for a new packet from the receiver. Therefore, if *i* is the most recently received request, the transmitter is allowed to send packets within a window of *n* packets, from *i* to *i+n-1*; this window slides as higher successive requests are received from the receiver. Subsequent error-free packets following one erroneous packet are rejected by GBN, since it does not include the possibility of buffering out-of-order correctly received packets while waiting for the requested packet. It is evident that a single packet error can cause the unnecessary retransmission of a large number of subsequent correctly received packets, especially when the number of frames transited in a round-trip delay time is very large. This leads to some performance problems, mainly attributed to throughput efficiency. A buffering strategy is adopted in the alternative SR ARQ scheme, discussed in the following section.

3.2.2.2 Selective Repeat (SR)

When a single error occurs in a received packet, the GBN scheme must retransmit at least one round-trip delay worth of packets. The basic idea of SR ARQ is to accept out-of-order packets and to request retransmission for only those packets that are not correctly received. Since the requested packets are retransmitted within one round-trip delay period, the receiver must store out-of-order packets in a queue until a requested in-sequence packet is received correctly. Chapter 5 presents an

implementation of SR ARQ, where the receiver maintains a priority queue as required by the adopted sorting algorithm; this is explained in the following sections.

The SR ARQ protocol overcomes the performance problems of the other types of ARQ schemes previously considered [81] - such as SW and GBN - by permitting the sender to continuously send packets, and by allowing the receiver to accept correctly received packets even if earlier packets were damaged or lost. Furthermore, the SR ARQ scheme retransmits only those packets which were lost or corrupted; this requires the receiver to individually acknowledge each correctly received packet and re-sort out-of-order packets.

Figure 3.14 portrays a typically reliable data exchange session taking place between sending and receiving nodes operating with SR ARQ. Node A sends a batch of packets continuously to node B, and node A then receives back either ACK or NAK signals after the round-trip time. Consequently, node A carries on sending consecutive packets upon receiving an ACK signal, and also continues resending the corrupted packets corresponding to the NAK received signal. In Figure 3.14, packets 2, 3 and 6 are selectively retransmitted.

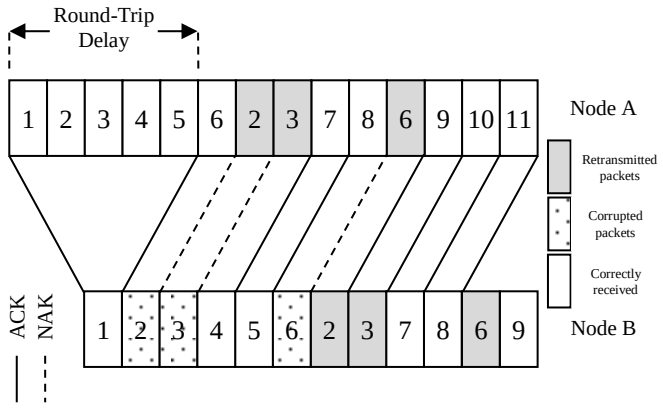


Figure 3.14: A typical SR ARQ transmission procedure [80]

In the SR ARQ scheme, a sliding window of size *W* is used to limit the number of unacknowledged packets sent over the transmission link. Since the receiver window size is greater than 1, correctly received packets following a damaged packet are always buffered, irrespective of their order. Thus, the received packets will be queued waiting for the missing packets to arrive. The receiver will send a batch of packets to the upper layer once the packets with lower sequence numbers are correctly received. Figure 3.15 shows the actions of the simulated transmitter upon receiving a new packet from the upper layer and an ACK signal from the lower layer, adapted from [80]. When data sets are ready to be transmitted from the upper layer, the sender checks its sequence number (*nextseqnum*); if the sequence number is within the sender's window, the data are packetized and sent; otherwise they are either buffered or returned to the upper layer. When an ACK is received, the sender marks it as having been delivered successfully. In addition, if the ACK sequence number is equal to the first sequence number in the queue (*send-base*), the window is then moved forward to the next unacknowledged packet, allowing the awaiting upper layer packet to be advanced.

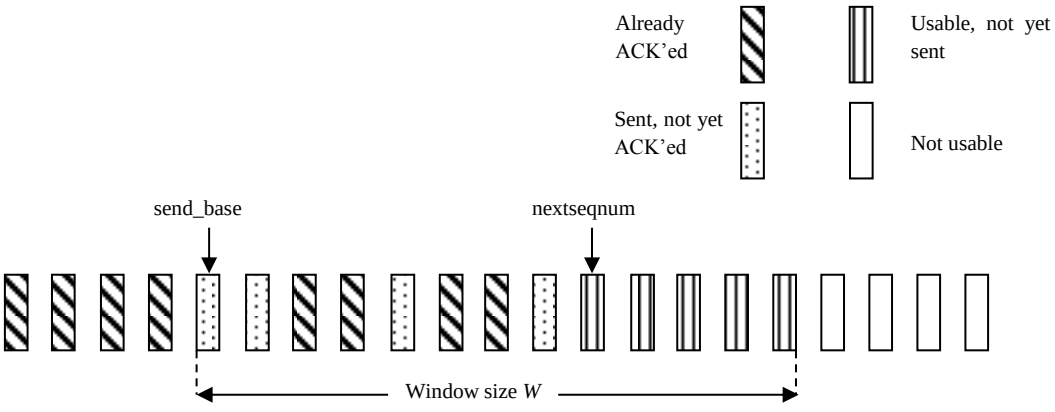


Figure 3.15: Sender view of sequence numbers in SR ARQ

The diagram in Figure 3.16 illustrates the receiver's view of the status of the packets. At the receiver side, correctly received packets are acknowledged and buffered in the

receiver queue. If the sequence number of the queued packet is equal to the first sequence number in the queue, then all the packets starting from the sequence number (*rcv_base*), up to those consecutive packets which have been already acknowledged, are submitted to the upper layer. The receiver window then advances a number of places equal to the number of packets delivered to the upper layer.

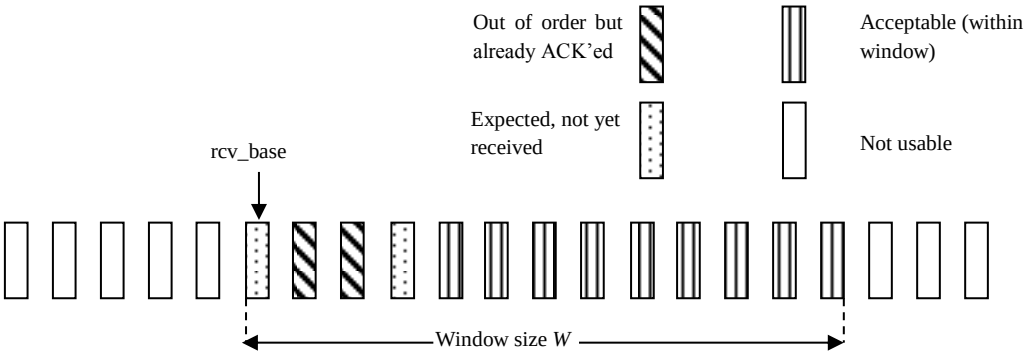
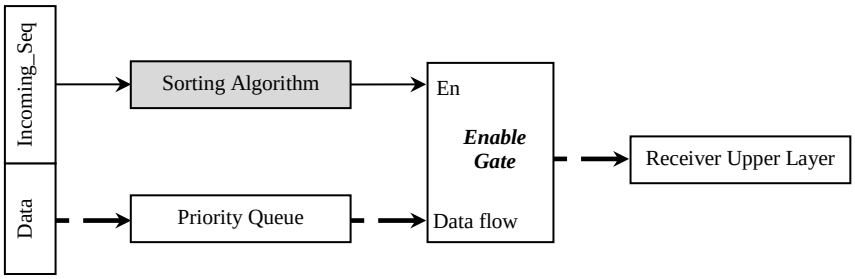


Figure 3.16: Receiver view of sequence numbers in SR ARQ

3.2.2.2.1 Sorting Algorithm

As described in the previous section, the receiver queue may contain out-of-order acknowledged packets, which necessitates the use of a sorting algorithm to deliver the packets in sequence to the receiver upper layer. The sorting algorithm presented in Figure 3.17 is adopted, where *rcv_base* is used to mark the expected packets. If the incoming packet sequence number (*Incoming_Seq*) is greater than *rcv_base*, *Incoming_Seq* is stored in the (*Incoming_Seq* modulo *Buffer_size*) location, then the corresponding packet is placed into the priority queue, which sends the lowest sequence number packet to the top; otherwise, the queue advances the packet to the upper layer along with those acknowledged packets with a higher sequence number, and updates *rcv_base* accordingly. The receivers use the pseudo code in algorithm 1 with reference to Figure 3.16.



1: **Set** [Buffer_size]=0, rcv_base=0
2: **if** the incoming_Seq equal to rcv_base **then**
3: **Send** packet to the upper layer
4: **update** rcv_base
5: **else**
6: **if** the incoming_Seq greater than rcv_base **then**
7: **store** packet in the priority queue
8: **update** incoming_Seq location

Algorithm 1. The SR ARQ Sorting Algorithm

Figure 3.17: Graphical and coded representation of the sorting algorithm

3.2.2.2.2 Priority queuing

In order to sort out-of-order correctly received packets, a *priority* queue is utilized to push packets with a low sequence number to the top of the queue, so as to deliver a batch of in-order successfully received packets to the upper layer once a low-sequenced packet has arrived after a successful retransmission. Since each packet is attached with a *SeqNum* as an *attribute*, the *Priority* queue can prioritise them according to the *ascending* criterion.

3.3 Conclusions

This chapter provides a brief overview of a number of error control schemes, and accurately models an error control scheme, SW ARQ, for a real-world communication link, using SimEvents. The model includes customized queuing, routing, delay processing, and other operations.

In the literature reviewed, there exists no known simulation approach for the deployment of a control unit to manage packet flow for SW ARQ; by developing a novel model that tackles this issue, a contribution is made to the implementation and performance analysis of an error control scheme, SW ARQ.

The following chapter examines the throughput benefits of NC when implemented in an intermediate link, along with SW ARQ.

Chapter 4

Network Coding for SW ARQ Based Communication

4.1 Introduction

Chapters 2 and 3 have provided a brief introduction to the concept of network coding (NC) and error control techniques, respectively, with the latter explaining the functionality of the three main ARQ techniques, namely stop-and-wait (SW), go-back-N (GBN) and selective repeat (SR). The SW ARQ protocol is perfectly adequate for short links where the propagation delay is smaller than the transmission delay. However, for long terrestrial links, the performance of SW ARQ falls off significantly as the data rate increases [81]. This chapter specifically studies the throughput performance of an SW ARQ-assisted network operating under these constraints, employing NC at an intermediate link, and exploiting the control unit presented in Chapter 3.

In traditional SW ARQ, the packet source sends one packet of size *n*, and must wait for an ACK/NAK signal, which indicates whether the packet was correctly received or not. However, in the scheme proposed in this chapter, the incoming packets, at the intermediate node, are aggregated and stored. Once *k* packets have been prepared, the node establishes a coding process which results in *k* new linearly coded packets - these are then sent to the receiver. At the receiver, a decoding process is performed, and only one ACK/NAK signal is sent back as an acknowledgment. There have been

several studies concerning the benefits that the use of feedback can offer for network coded traffic [6, 49, 53, 54, 89]. The authors in [53, 54, 89] investigated the throughput of an intermediate general network coding node employing SW ARQ as an error control scheme over all communication links. The study concentrates mainly on the throughput of the coding nodes, since they constrain the general throughput performance; the use of a coding matrix at the intermediate node and the analysis of the throughput as a function of the propagation time distinguish the work presented in this chapter.

The main aim of the work described in this chapter is to improve SW ARQ-based point-to-point communication by utilising NC techniques at an intermediate network link. The results show that NC offers a throughput advantage of between 2.5% and 50.5% over traditional SW ARQ, depending on the number of incoming links and packet error rates. The network model, its characteristics and presumed assumptions are presented in Section 4.2. The simulation model and throughput analysis with the results obtained are demonstrated in Sections 4.3 and Section 4.4, respectively. Section 4.5 concludes the chapter.

4.2 The Network Model

The system considered in this chapter is a typical intermediate link (W_1, W_2) located in a *butterfly* network as shown in Figure 4.1 [1]. The *Butterfly* network modelled can be described by a finite directed graph *G(V, E)* where *V* is the set of nodes and *E* is the set of edges. Packets are generated by a source $S \in V$, and transmitted through the network to both nodes $\{R_1, R_2\} \in V$, referred to as the sink nodes. It is assumed that all edges have the same capacity, *C*. The link (W_1, W_2) conveys packets through a forward channel, and receives acknowledgment through a feedback channel. The

feedback channel is assumed to be perfect and hence ACK/NAK signals arrive back safely at the sender. It is also assumed that packets arrive successfully at W_1 regardless of the error control used at its incoming links.

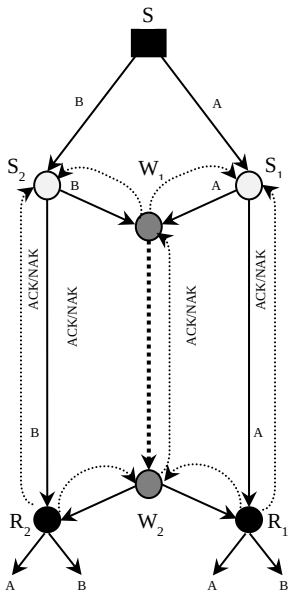


Figure 4.1: The multicast scenario model studied

As described in Chapter 3, in a typical SW ARQ system, the transmitter sends a single packet and waits for its acknowledgment, which takes a round-trip time R_{TT}, equal to the transmission time plus the propagation time spent on the forward and feedback channels. A modified NC scheme, NC-SW ARQ, is employed to help in reducing the number of ACKs/NAKs in the SW ARQ scheme at each transmission. Therefore, the achievable throughput can be significantly improved by reducing the propagation time needed for the ACK/NAK signals at each transmission, which consequently leads to a reduction in the transmitter idle time. Figure 4.2 (a) illustrates how the receiver sends, for example, three ACK signals to acknowledge the reception of three packets. However, in Figure 4.2 (b), the sender performs a coding operation and sends three correlated blocks b_1, b_2 and b_3, hence the receiver is

required only to send one ACK/NAK signal to acknowledge reception upon decoding.

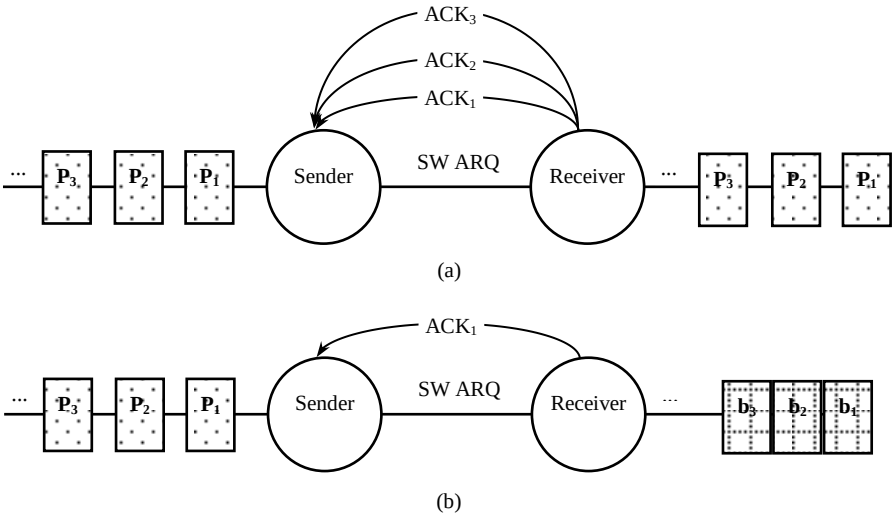


Figure 4.2: (a) Typical SW ARQ transmission, (b) NC based SW ARQ transmission

As described in Chapter 2, an outgoing symbol, b, of a NC node is a combination of the incoming message of k symbols $[p_1, p_2, \ldots, p_k]$ at this nodes; the node randomly chooses a *vector* of coding coefficients $[c_1, c_2, \ldots, c_k]$ to produce a linear combination, b, of the original data block of size n.

$$b = \sum_{i=1}^{k} c_i p_i \tag{4.1}$$

This combinatory method of NC is utilised in Chapters 5 and 6, where the benefits of NC are investigated in SR ARQ-based networks. However, in this chapter, packets at W_1 are linearly encoded with predetermined coefficients of an *encoding matrix* C_{ij}, then advanced into the channel. Using an encoding matrix, the intermediate node output will advance a generation of k linearly correlated packets, so that only one ACK/NAK signal will be required in order to acknowledge the reception of this generation upon decoding; this reduces the total transmission time from $k(2T_p + T_r)$ to $2T_p + kT_r$ time units, where T_p and T_r are the propagation and transmission times,

respectively. As mentioned in Chapter 2, coding coefficients can be randomly generated and embedded into each coded block header [26] in the so-called coding vector; this eliminates the need for centralised knowledge of the network topology, and makes the approach suitable for large and dynamic networks. Alternatively, the sender and receiver can agree on one deterministic invertible encoding matrix, **V**, which is the method adopted in this analysis. This method reduces the operational cost of the decoding, as receivers solve the same system of equations for each transmission [37]. A ($k \times k$) Vandermonde matrix [45] is used, where $\alpha_i \neq \alpha_j$ for $i \neq j$ to ensure det(**V**) $\neq 0$:

$$V = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_k \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \dots & \alpha_k^{k-1} \end{pmatrix} \tag{4.2}$$

The row elements of *V* ensure a non-linear relationship between the encoded blocks, which allows the receiver to successfully decode the blocks once *k* blocks have been received. As shown in Figure 4.3, packets at the transmitter are linearly encoded using a (*k, k*) matrix; this results in *k* linearly correlated packets, which can be advanced as one block, $b = [b_1, b_2, \dots, b_k]$. At the receiver side, a progressive decoding process [90], using Gauss-Jordan elimination, can be used to extract back the original packets for each generation. Immediately after the arrival of *k* independent linearly -coded blocks, which belong to the same generation, the receiver is able to recover the original packets in their entirety and sends an ACK to the sender.

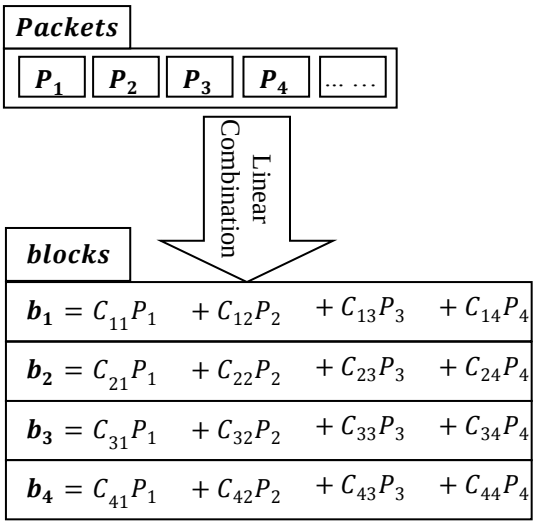


Figure 4.3: Packets are linearly combined to form one block, *k*=4

4.3 Simulation Model

The communication system employed is shown in Figure 4.4, consisting of a packet generator, transmitter, receiver, forward channel, and feedback channel. Coding and decoding stages are added at both the transmitter and receiver respectively. Packets are generated at a rate of λ packets per second, which is chosen according to the practical considerations discussed below. Blocks are not generated simultaneously; therefore the transmitter must wait for k blocks to arrive before advancing them to the coding stage.

Packets are transmitted over the transmission link, based on the SW ARQ scheme. The transmitter and receiver work over a 10 Mbps-capacity channel. The forward channel corrupts packets with a given probability according to a Bernoulli distribution model; this works on the assumption that the errors are not correlated, and forms a good starting point to investigate the utility of NC. Although Internet traffic shows significant correlation properties [91], there remain scenarios such as Satellite transmission [92], and interfaces for transmission between base stations and peer mobile devices [93], where packets are - to a large degree - uncorrelated. The

model assumes that the receiver can detect an error via a CRC check but cannot correct the error.

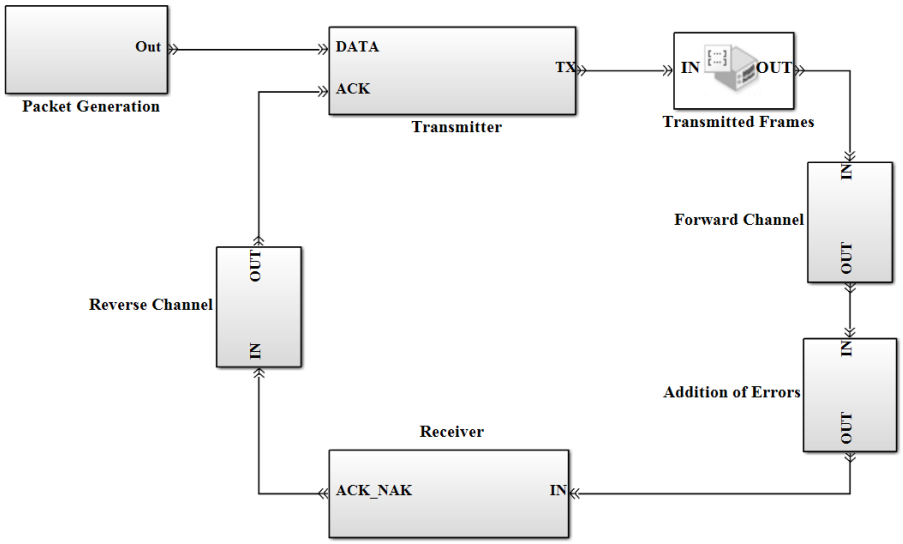


Figure 4.4: SW-ARQ based communication system

The packets produced by an entity generator arrive sequentially at the coding stage, which generates a function call to execute a NC program written in MATLAB, as depicted in Figure 4.5.

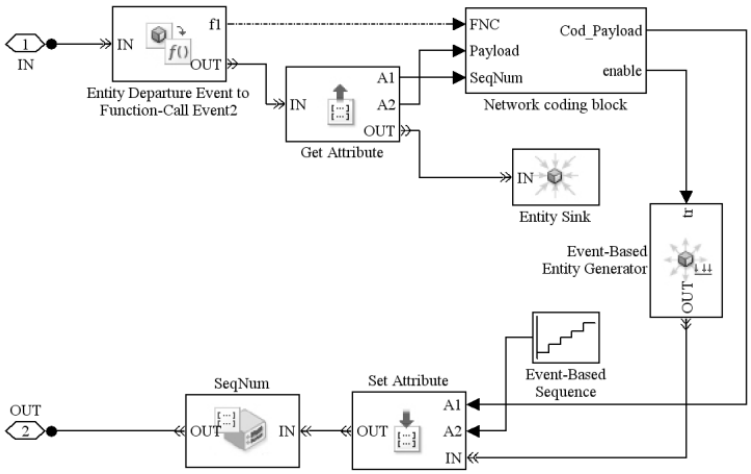


Figure 4.5: NC process model prior to transmission stage

The NC block is triggered by a *function-call* indicating the arrival of a new packet. The new packet is labelled with a sequence number, *SeqNum*, and a payload field, *Payload*. After *k* consecutive packets have arrived, coding is performed and the results are stored in the *Cod_Payload* field of a new packet, with a new unique

sequence number assigned to it. These coded packets are then advanced to the receiver, through the forward channel. The receiver performs a decoding process, and sends an acknowledgment to the sender through the feedback channel.

4.4 Throughput Analysis

The transmitter sends packets of size *n*, at a rate λ, to the receiver, using SW ARQ; hence, the intergeneration time between packets is given by $\zeta=1/\lambda$. The receiver is assumed to send back ACKs at a reasonable power, therefore the feedback channel is considered to be noise-free, so that the acknowledgment signals will always be successfully returned to the transmitter. As shown in Chapter 3, the transmission delay T_r is defined as the time consumed in emitting all of the bits comprising the block into the transmission channel. Given the Bit Error Rate (BER), the probability that a transmitted block of size *n* contains an error ***PER***, is given by:

$$\mathrm{PER} = 1 - (1 - \mathrm{BER})^{n} \tag{4.3}$$

An additional time delay, T_{cod}, is introduced when NC is employed; this delay mainly represents the time delay $k \times \zeta$ incurred waiting for the aggregation of *k* packets, since they are not always ready to transmit immediately. The coding operation itself, in today's advanced routers, tends to take a relatively negligible length of time.

The link utilization of SW ARQ is defined by the time when the transmitter is not idle, which is given by $U = T_r / R_{TT}$, where R_{TT} is the overall round-trip time needed to convey one packet [77]. The throughput, η, of a SW ARQ scheme is hence given by η = U*C, where *C* is the channel capacity [46]. In typical SW ARQ, the propagation delay T_p plays a significant role in R_{TT} ; that is, when sending *k* packets of data, the total transmission time takes $k(2T_p + T_r)$ while in NC-SW ARQ that is

reduced to $2T_p + kT_r$ which saves 2 (k-1) T_p. In the case of SW ARQ, the maximum throughput achievable is when the system is operated over error-free channels, and the intergeneration time is slightly greater than the round-trip delay R_{TT}. The time R_{TT}, in SW ARQ, is given by:

$$R_{TT} = 2T_p + T_r \tag{4.4}$$

Whereas in NC-SW ARQ, R_{TT} and T_{cod} become part of the total transmission time R_{tot}

$$R_{tot} = R_{TT} + T_{cod} \tag{4.5}$$

Equation (4.4) indicates that T_p plays a key role in the efficiency of the SW ARQ scheme, which can be improved by reducing the time required to convey data. Equation (4.5) states that coding delays in NC-SW ARQ could play a significant role in addition to T_p. Nevertheless, the coding delay can be reduced to a minimum once the intergeneration time is decreased, if possible, as shown in Section 4.4.3.

The propagation delay, T_p, is adopted from [80], according to the distance between the East and West Coast of the United States, which is approximately 15 milliseconds, based on the distance and the speed-of-light propagation. However, the benefit of NC-SW for various values of T_p is explored in Section 4.4.2. The system operates at 10 Mbps, so that the time needed to actually transmit a 1000 bit packet, T_r, is 100 μs. The intergeneration time, ζ, is the time interval between successive entities that the block generates and is given a value of 0.02 seconds for both the SW and the NC-SW ARQ scenarios. It is initially assumed that the system sends k = 5 blocks and receives one acknowledgment; later, k is set to vary between two and five to examine the throughput benefits this may bring to the traditional SW ARQ. Numerically, SW ARQ throughput will take the value *η=33.22 kbps* as examined in

Chapter 3. However, for NC-SW ARQ with k=5, $\eta = \frac{1\times10\times10^{6}}{2\times\frac{0.015}{5\times10^{-4}}+1}$=*163.9 kbps*. This value of throughput shows a significant improvement over the traditional SW ARQ, however, it will not exceed the data rate at which the packets are generated. This improvement corresponds with the case where packets are transferred over an error-free channel. However, it will be shown in Section 4.4.2 that network performance degrades drastically with a high PER and a relatively large generation, when NC-SW is adopted.

Figure 4.6 depicts the maximum achievable throughput for the SW and NC-SW ARQ systems, plotted against the elapsed simulation time, for the same packet intergeneration time. It may be observed that NC-SW ARQ achieves a throughput of ~50 kbps, while only ~33 kbps delivered by the traditional SW ARQ. This is because NC-SW saves a time of 2 (k-1) T_p in every k transmissions, whereas $2T_p+T_r$ is required for each packet transmission in SW ARQ.

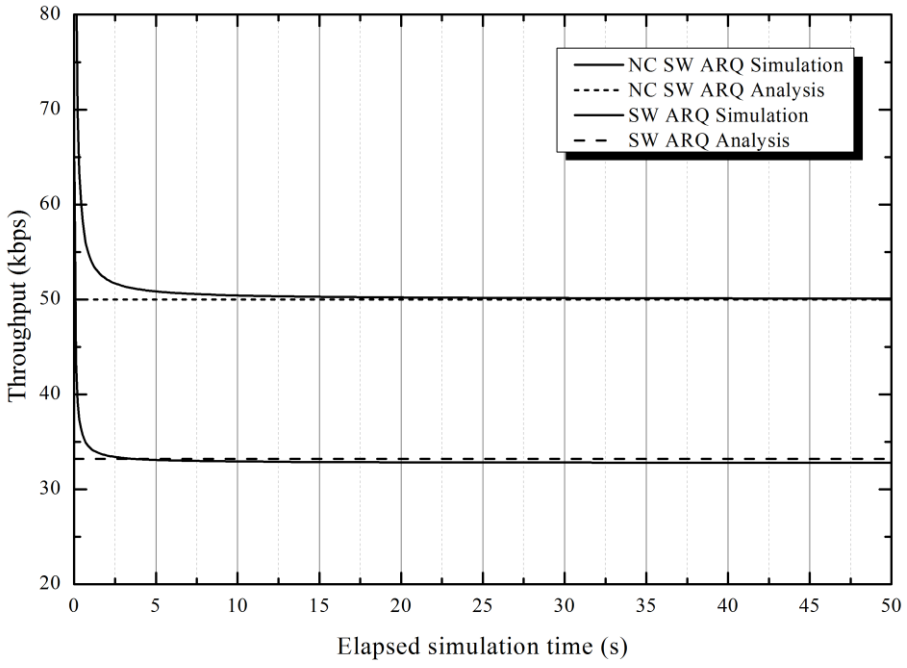


Figure 4.6: Traditional SW and NC-SW throughput over an error-free channel, k = 5

The same scenario is applied with PER=0.1. Figure 4.7 shows that, with increasing PER, packet retransmission occurs, adding a delay equal to the time required for sending the same number of correct packets.

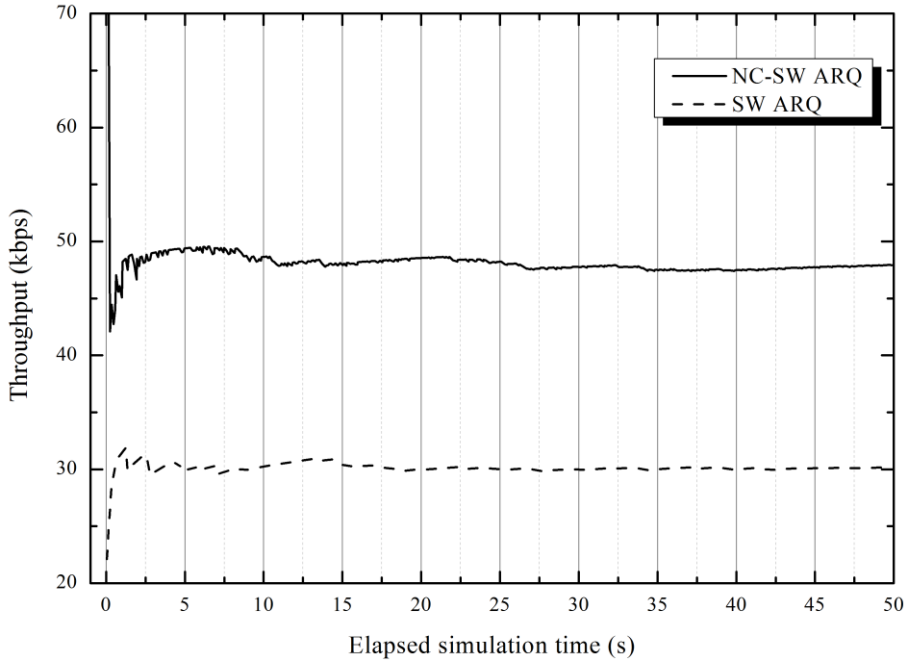


Figure 4.7: Traditional SW and NC-SW throughput when PER is 0.1, k = 5

4.4.1 Throughput comparison under varying the number of incoming links and packet error probability

In this section, the number of incoming links, *k*, is set to vary between two and five in the case of NC-SW, whereas it takes the value one in SW ARQ. The average throughput for SW and NC-SW with PER is demonstrated in Figure 4.8, where a clear benefit from using NC can be seen for most of the cases considered.

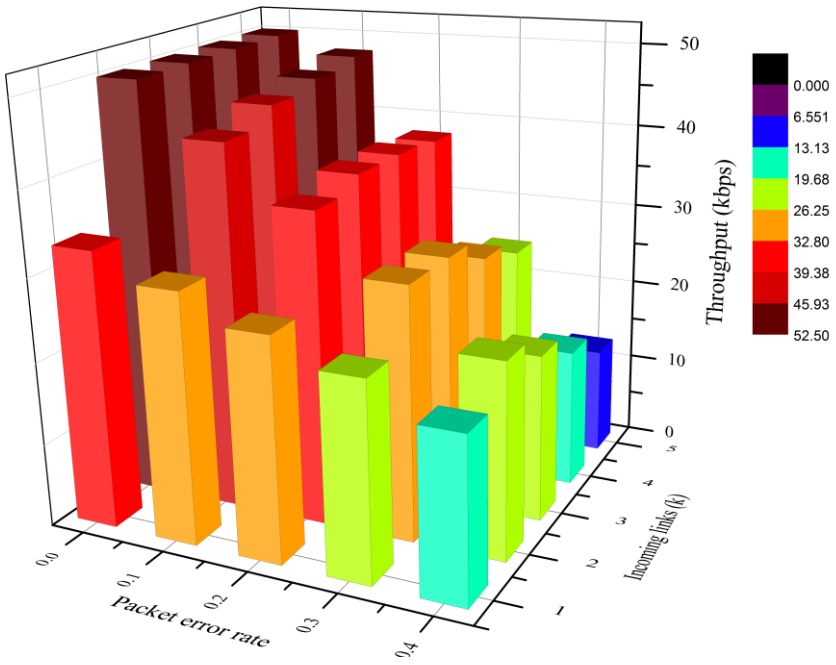


Figure 4.8: Throughput comparison of SW (k=1) and NC-SW ARQ with various PER and incoming links.

For error-free transmission, and a PER equal to 0.1 and 0.2, NC-SW ARQ offers a 50.5% and 42% throughput increase over the traditional SW ARQ, respectively. For PER = 0.3, NC-SW offers a 27% increase, when k = 2 and 3, and 4.8% and 4% for k = 4 and 5, respectively. When k = 2 and 3 and PER = 0.4, NC-SW offers an increase over SW ARQ of 19.4% and 2.5%, respectively. However, the performance of NC-SW starts to degrade for values of k higher than 4, in the presence of PER=0.4.

In conclusion, NC-SW ARQ offers a significant advantage over traditional SW ARQ when PER is less than 0.4, regardless of the value of k. SW ARQ is generally the preferred ARQ protocol in underwater acoustic channels, due to their half-duplex properties. It may thus be stated here that the NC-SW scheme offers good prospects for applications in underwater communication systems [57, 94].

4.4.2 Throughput comparison under varying propagation time

The previous observations can be further enhanced by investigating the throughput benefits of NC for various propagation delays. In this scenario, propagation delays

are set to take values between 10μs and 15ms, for 10 Mbps channel capacity and PER=0.1. From Figure 4.9, it can be seen that NC is favourable in scenarios with relatively high T_p; NC-SW ARQ achieved a throughput advantage of between 3% and 60% when T_p varied between 0.009 and 0.015 seconds.

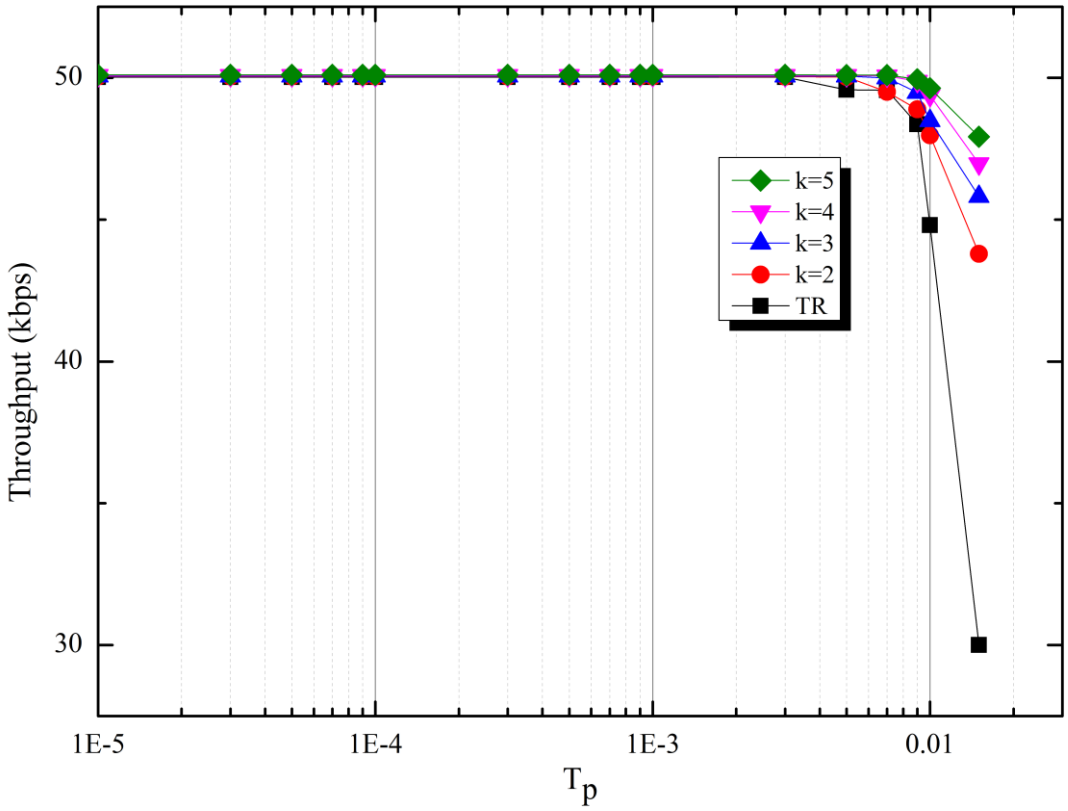


Figure 4.9: SW throughput comparison of traditional routing and NC with various propagation times

4.4.3 Performance optimization

To achieve the maximum throughput possible, the packets generated should always be ready to be sent. In the case of SW ARQ, the packet intergeneration time, ζ, has to be equal to, or slightly smaller than, the round-trip delay, i.e.

$$\zeta \leq 2T_p + T_r \tag{4.6}$$

Therefore, the optimum throughput in this case can be achieved when ζ obeys (4.6). However, any decrease in ζ will incur a queuing problem, and will not affect the

throughput performance of SW ARQ, due to the previously described propagation time domination constraint.

In the case of NC-SW ARQ, ζ can be calculated by making the time needed for k packets to arrive, T_{cod} = kζ, slightly smaller than the round-trip delay. This allows a faster data rate, which can be easily achieved using modern technology, resulting in:

$$k\zeta \leq 2T_p + kT_r \tag{4.7}$$

$$\zeta \leq (2T_p + kT_r)/k \tag{4.8}$$

Implementing equation (4.8) when k=5, in an error free scenario, yielded the results shown in Figure 4.10 as calculated in Section 4.4.

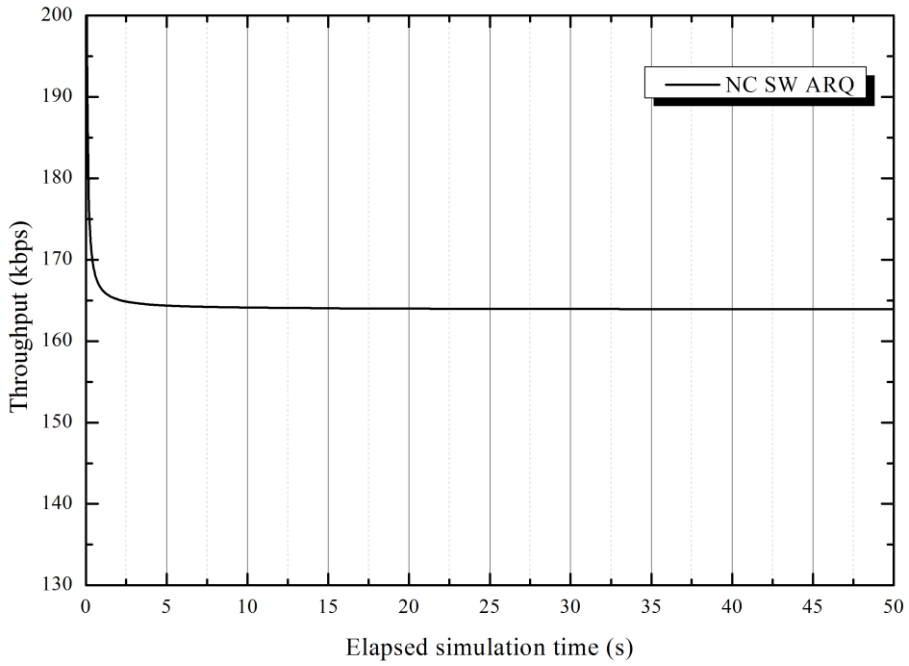


Figure 4.10: Optimized NC-SW ARQ

In contrast to SW ARQ, it may be seen that the value of ζ can greatly improve the throughput of an NC-SW ARQ system. This improvement comes from allowing new packets to always be ready for coding, rather than waiting for the packets to be generated.

4.5 Conclusions

This chapter has investigated the throughput performance of a NC node at an intermediate link, where an SW ARQ error control scheme is employed. It shows that, in the presence of higher data rates and long round-trip delays, conventional SW ARQ becomes inefficient, due to the increased retransmissions and transmitter idle time. The proposed NC scheme offers an increase in throughput over traditional SW ARQ of between 2.5% and 50.5% in each link, with negligible decoding delay. This benefit applies for packet error rates up to 0.4 and the number of coded packets $k > 4$, when SW ARQ throughput starts to degrade due to the retransmission of large generations of coded packets. In addition, optimising the NC intergeneration time offers much higher throughput, because packets are always waiting to be decoded in this optimised situation.

NC allows a remarkable gain in network throughput; the throughput can be further improved by adopting a better error control scheme where propagation delay has less effect. This requires a Selective Repeat (SR) ARQ scheme, which is described in the next chapter.

Chapter 5

Network Coding for SR ARQ Communication

5.1 Introduction

From the performance results shown in Chapter 4, the efficiency of the SW ARQ scheme has been improved by employing NC in point-to-point communication at intermediate nodes. The principal advantage of SW ARQ is its simplicity; however, it is inefficient, since only one packet at a time can be in transit. This chapter therefore presents an application of NC in a more sophisticated ARQ scheme, Selective Repeat (SR), for both unicast and multicast network models. Chapter 3 has shown that the SR ARQ scheme is considered to be the most efficient of the ARQ schemes studied; the sender node is allowed to send its packets continuously, and is required to retransmit only the corrupted packets, unlike in SW and GBN schemes. This chapter is split into two parts; the first part states the overall throughput performance of a typical NC-based SR ARQ scheme, termed as NC-SR ARQ, at an intermediate link experiencing different *Packet Error Rates* (PERs), using both simulation and numerical approaches; the second part investigates the advantages of a NC-based SR ARQ scheme in a multicast *butterfly* network under different communication bandwidths and error pattern scenarios.

A detailed simulation model for the network topology and elements necessary to construct a NC-based network infrastructure with the SR ARQ protocol are provided. Although there has been generally some recent previous work on NC for general ARQ in wired and wireless networks [6, 49-51, 54, 95, 96], what is

presented here is the first illustration of the effect of varying propagation times compared to the transmission times in SR ARQ.

The throughput performance is again studied as a function of the factor α, which represents the ratio of the propagation time, T_p, to the packet transmission time, T_r. The analysis shows, in both unicast and multicast scenarios, that the throughput performance is found to be sensitive to the number of incoming links per node, and the available bandwidth at which the channel is operating.

5.2 The Unicast Scenario

5.2.1 Performance analysis

Let $b_1, b_2, \dots b_k \in \mathbb{F}_q$ be the packets arriving at the intermediate node W_1, which is considered as the source node in the simulated model, as shown in Figure 5.1. The transmission on the link is assumed to be synchronized in such a way that packets present at W_1 are always ready to send.

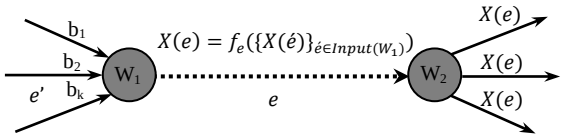


Figure 5.1: A NC intermediate link

Let $X(e)$ denote the symbol being sent on link $e \in E$, where E is the set of the network edges, and $X(e)$ is expressed as a combination of the packets that enter at node W_1 as follows:

$$X(e) = f_e(\{X(\acute{e})\}_{\acute{e} \in Input(W_1)})$$

The intermediate node W_2 relays the coded blocks $X(e)$ to its outgoing ports toward the sink nodes $t_i \subset T$, which should be able to retrieve the original packets by solving the set of equations $\{f_e(b_1, b_2, \dots b_k) = X(e)\}_{e \in input\ (t_i)}$.

When the data segment blocks are about to transmit, node W_1 chooses a set of coding coefficients to produce linear combinations, *X*, of the original data blocks:

$$X = \sum_{i=1}^{k} c_i b_i \tag{5.1}$$

where *k* is the number of incoming links at W_1.

The total time required between emitting one packet and receiving its acknowledgment is equal to $2T_p+T_r$. The link utilization, *U*, for SR protocol is given by [77];

$$U = \frac{WT_r}{2T_p + T_r} = \frac{W}{2\alpha + 1} \tag{5.2}$$

where *W* is the transmitter window size, as defined in Chapter 3.

Two cases need to be considered:

Case 1: $W \geq 2\alpha + 1$. The ACK/NAK signal of any transmitted packet will be returned to the sender before the transmitter queue is exhausted. In this case, the transmitter sends packets continuously, without pause; hence the throughput will be at its maximum.

Case 2: $W < 2\alpha + 1$. The transmitter packet queue starts to become accumulatively exhausted. Therefore, additional packets cannot be sent during the transmission, until a space becomes available due to the receipt of an ACK/NAK signal, as shown in Figure 5.2.

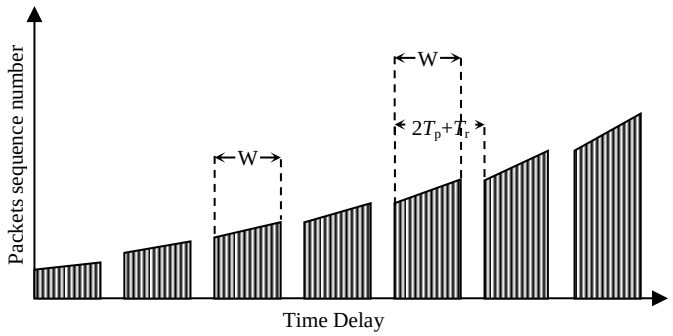


Figure 5.2: SR ARQ Packets over the channel when $W < 2T_p + T_r$

Therefore, the link utilization can be expressed as follows:

$$U = \begin{cases} 1 & W \geq 2\alpha + 1 \\ \dfrac{W}{2\alpha + 1} & W < 2\alpha + 1 \end{cases} \tag{5.3}$$

The throughput, *Th*, can be obtained by multiplying the link utilization by the channel capacity [46]. If the NC-SR ARQ scheme is adopted, the probability that a coded block of size *n* contains an error is given by:

$$PER = 1 - (1 - BER)^n \tag{5.4}$$

where BER is the bit error rate.

In a traditional SR ARQ scenario, any un-coded packets of size $k \times n$ can be potentially represented by *n* coded packets if NC is employed, depending on the number of incoming links. Therefore, the value of the packet error rate varies according to the following equation:

$$PER_{tr} = 1 - (1 - BER)^{kn} \tag{5.5}$$

By rearranging (5.5), PER_{tr} can be rewritten as:

$$PER_{tr} = 1 - (1 - PER)^k \tag{5.6}$$

where *k* represents the number of packets present at the input port of the intermediate node and destined for the same output port. It should be noted that NC reduces the

number of retransmissions, since one coded packet is less prone to error than two or more un-coded consecutive packets.

If errors occur, equation (5.3) is divided by the expected number of transmissions, *N*r. If P is the probability that a packet is received in error, the probability that a single frame is received successfully after $\mathcal{E}$ attempts is $\mathrm{P}_{\mathcal{E}} = \mathrm{P}^{\mathcal{E}-1}(1-\mathrm{P}), \mathcal{E} = 1,2,3,\ldots$ That is, there is a probability of ($\mathcal{E}$ -1) unsuccessful attempts followed by one successful attempt (1-P). Hence, recognizing the arithmetico-geometric sequence $\varepsilon \mathrm{P}^{\varepsilon-1}$, the average number of retransmissions is given by:

$$N_{\mathrm{r}} = \sum_{\mathcal{E}=1}^{\infty} \mathcal{E}\mathrm{P}_{\mathcal{E}} = \sum_{\mathcal{E}=1}^{\infty} \mathcal{E}\mathrm{P}^{\mathcal{E}-1}(1-\mathrm{P}) = \frac{1}{1-\mathrm{P}} = \frac{1}{1-\mathrm{PER}} \tag{5.7}$$

Hence, U becomes

$$U = \begin{cases} 1-PER & W \geq 2\alpha+1 \\ \dfrac{W(1-PER)}{2\alpha+1} & W < 2\alpha+1 \end{cases} \tag{5.8}$$

By substituting equation (5.6) into (5.8), the link utilization in traditional SR ARQ, U_{tr}, is given by the following equation:

$$U_{\mathrm{tr}} = \begin{cases} (1-\mathrm{PER})^{\mathrm{k}} & \mathrm{W} \geq 2\alpha+1 \\ \dfrac{\mathrm{W}(1-\mathrm{PER})^{\mathrm{k}}}{2\alpha+1} & \mathrm{W} < 2\alpha+1 \end{cases} \tag{5.9}$$

A comparison between equations (5.8) and (5.9) indicates that U_{tr} decreases exponentially with increasing values of *k*, which means that, as the number of incoming node links increases, the potential performance improvement which NC can offer also increases.

In a traditional SR ARQ scheme, the receiver acknowledges correctly received packets and sends ACK signals for each packet. Nevertheless, in a NC-SR ARQ

system, the receiver acknowledges coded blocks of size *n*, which represent a collection of *k* packets, assembled as one coded block. Therefore, one ACK/NAK signal is sufficient to acknowledge several packets at once. Furthermore, the error probability will be drastically reduced, as it is directly proportional to the transferred data size. Consequently, this can significantly improve the throughput efficiency of the system, especially in a noisy environment.

In the traditional SR ARQ scenario, packets are advanced one after the other at the output port, resulting in a transmission time delay, kT_r. Therefore, in addition to the packet error rate, the key factors that specify the link performance are the bandwidth at which the node is operating, and the packet size to be handled at the intermediate node. The bandwidth and the packet size can be investigated as one factor, since T_r = *packet size* / *channel capacity*. Therefore, the simulation is concentrated mainly on investigating the variations in throughput, relative to packet error rate and channel bandwidth. The termslink speed and chanel capacity refer to the rate at which the channel can trasfer data, given in bits-per-second (bps).

5.2.2 The Unicast communication model

Figure 5.3 shows a SimEvents-based end-to-end communication model, consisting of an information source, transmitter, forward and reverse communication channels, and a receiver. The model used in this analysis incorporates the major features of the SR ARQ scheme detailed in Chapter 3 and use similar block diagrams of Figure 4.4. The information source, represented by the *Packet Generation* subsystem, generates packets with a constant intergeneration time, and each packet has a unique sequence number which identifies it.

The transmitter has two functionalities; one is to advance the incoming packets into the communication channel and store a copy of them in the forward queue of size *W*; the second is to manipulate the received ACK/NAK signals so that the forward queue discards the packets or returns them to the transmitter to be sent again, accordingly. The flow control is handled through the forward queue, which becomes full when the number of the unacknowledged packets becomes equal to the predefined sliding window size, *W*.

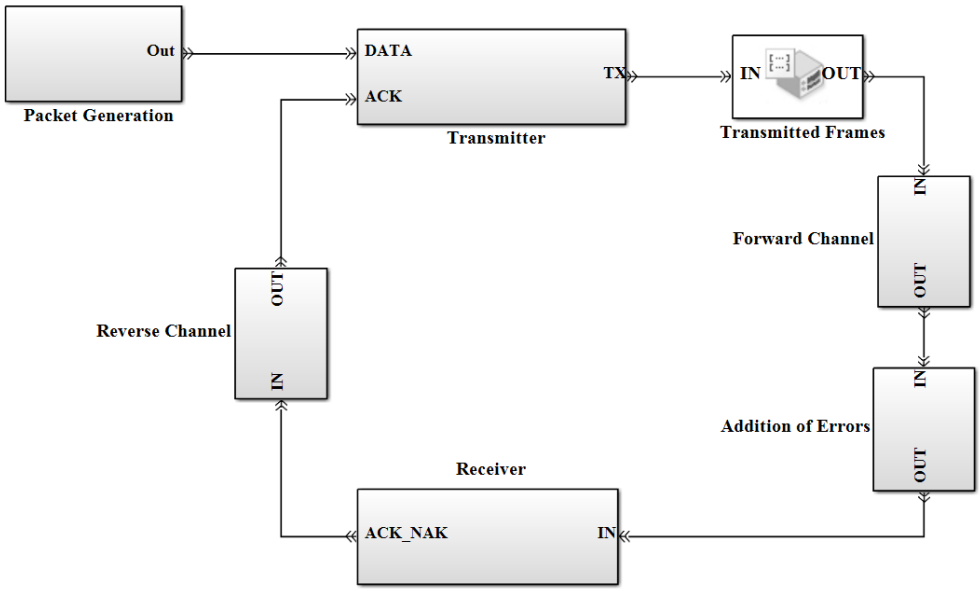


Figure 5.3: SimEvents-based communication network

The forward and reverse channels are represented by a *server*, the parameters of which determine the propagation delay. The forward channel introduces errors according to a *Bernoulli* distribution. The packets generated are sent continuously through the forward channel, and a copy of each one is stored in the forward queue before it becomes full.

Figure 5.4 illustrates the general functions of the *Packet Generation* subsystem. The packets are generated using a *time-based generator* block; the data, packet size and sequence number are then attributed to each packet using a *set attribute* block.

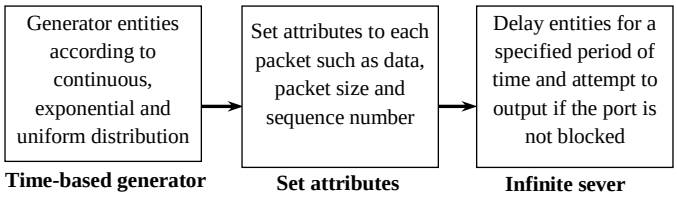


Figure 5.4: Packet generation general blocks

Using the packet size attribute and the available bandwidth, the infinite server then imposes a transmission time delay on each passing packet. This delay can be set to vary according to the number of packets present at the input port of the node, in the case of traditional SR ARQ.

The receiver accepts the correctly received packets, and delivers them to the upper layer. Otherwise, if an error occurs during the transmission, a NAK signal is sent to the transmitter to fetch the corresponding stored copy of the corrupted packets from the forward queue. At the receiver, out-of-order, safely-received packets are stored, and are subjected to the sorting algorithm detailed in Chapter 3; this delivers a batch of correct and properly-ordered packets to the upper layers of the protocol stack.

5.2.3 Simulation results for the unicast model

This section investigates the throughput performance as a function of the factor α, which varies between the values 0.1 and 1000, following the recommendation in [77]. In the studied scenarios, both sender and receiver nodes are set to have a sliding window that fits up to W=120 packets or entities.

In case of NC-SR ARQ, k of n-dimensional packets, are firstly coded into only one packet of size n, and then advanced into the channel. This makes the coded block less prone to error compared to k multiple packets of the same size. The value of k is chosen to vary between 2 and 5 incoming links. The test is first run under an error-prone transmission, where PER=0.01, with a large channel capacity of 10Mbps.

With a packet size equal to 1000 bits, each transmitted frame will take 0.0001 seconds to be emitted into the communication channel. Figure 5.5 presents the normalized throughput of node W_2. Under the aforementioned parameters, NC-SR ARQ can achieve a gain in throughput over SR ARQ of between 22% and 44%, when *k* varies between 2 and 5.

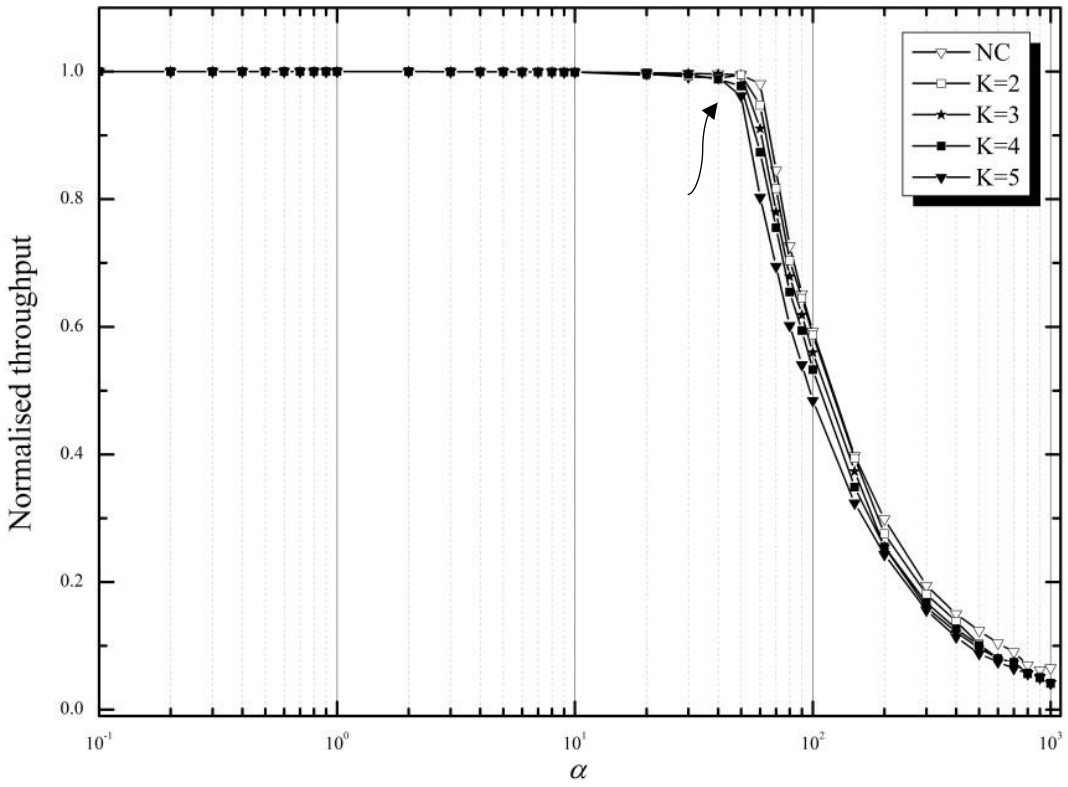


Figure 5.5: SR and NC-SR throughput as a function of α when PER is 0.01

In the second part of the experiment, simulation is carried out when the channel experiences a PER equal to 0.1. Figure 5.6 shows the normalized network throughput performance for both SR and NC-SR schemes having $k = 2,\ldots, 5$. It highlights the fact that, with a higher PER, the advantage of NC becomes more evident, because a coded block of size *n* is less prone to error than aggregated packets of size *kn* addressing the same destination. Moreover, the comparison between the results in Figures 5.5 and 5.6 demonstrates a throughput degradation in both scenarios, when PER=0.01 and 0.1, respectively. It can be seen in Figure 5.5

that the degradation when PER=0.1 starts at an earlier value of α, compared with Figure 5.6 where PER=0.01.

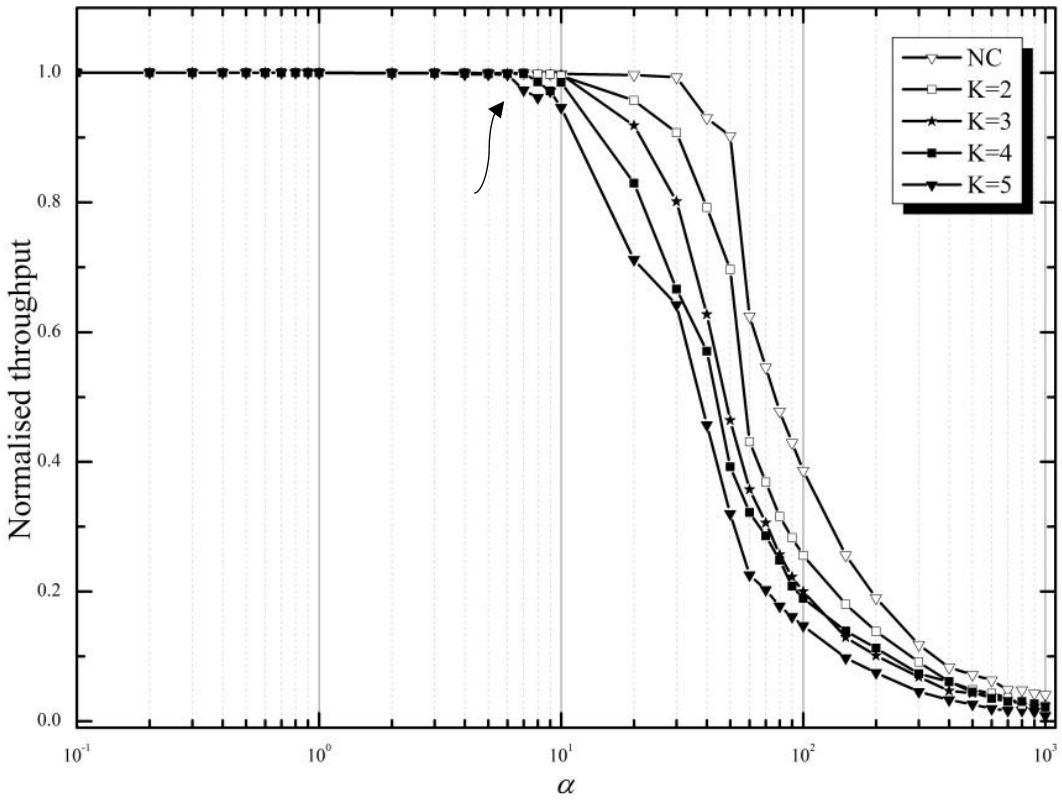


Figure 5.6: SR and NC-SR throughput as a function of α when PER is 0.1

To illustrate how the throughput performance starts to drop at a certain value of α, the statistical aggregation provided with the queue block is exploited to visualize the queue size behaviour over time. Figure 5.7 shows that the transmitter queue begins to grow at the start of the simulation, to be filled at a certain time after this. The queue begins to empty its packets once an ACK/NAK signal is received. However, it becomes full again as a new packet from the upper layer is allocated, instead of the one that the queue has just emptied.

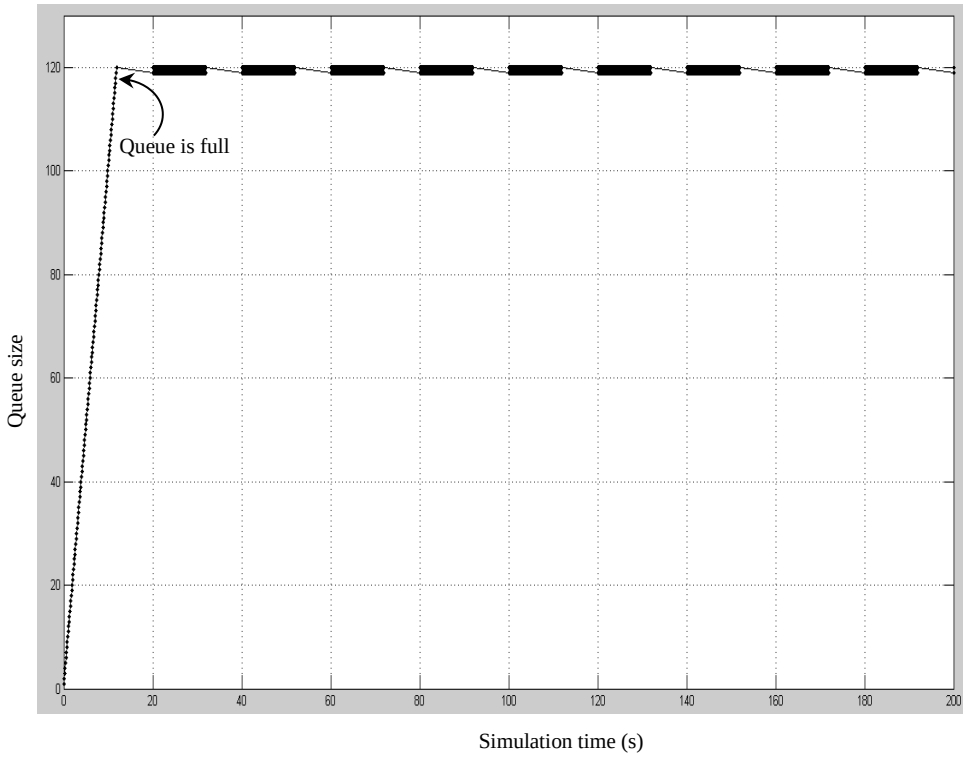


Figure 5.7: NC-SR ARQ forward queue size when PER = 0.01, W=120

In traditional *store-and-forward* routing, kT_r transmission time is required to fully advance k incoming packets addressing one output port into the channel, assuming that the packets are always ready to send. In NC-SR ARQ, however, k of n-dimensional packets are coded into only one block of size n, which leads to a saving of $(k\text{-}1)T_r$ in the transmission time, compared to that of traditional routing. In a high bandwidth channel, the transmission time becomes small and has less effect on the throughput performance. Results show that NC-SR ARQ offers a gain of only 0.003% over SR ARQ in an error-free 100 kbps channel. However, the impact of NC can be significantly noticeable in low-bandwidth communication channels, where the transmission time becomes large, especially when long packets are transmitted. Figure 5.8 shows the performance of a simulated scenario where NC provides a throughput gain of between 100% and 400%, with values of k between 2 and 5 and a packet size of 1000 bits, in a 10 kbps error-free channel. When operated at low bandwidth, the output port of the intermediate node requires a long time to emit each

packet. Since the use of NC saves a time of $(k\text{-}1)T_r$, the end-to-end delay becomes smaller, leading to a significant improvement in throughput performance.

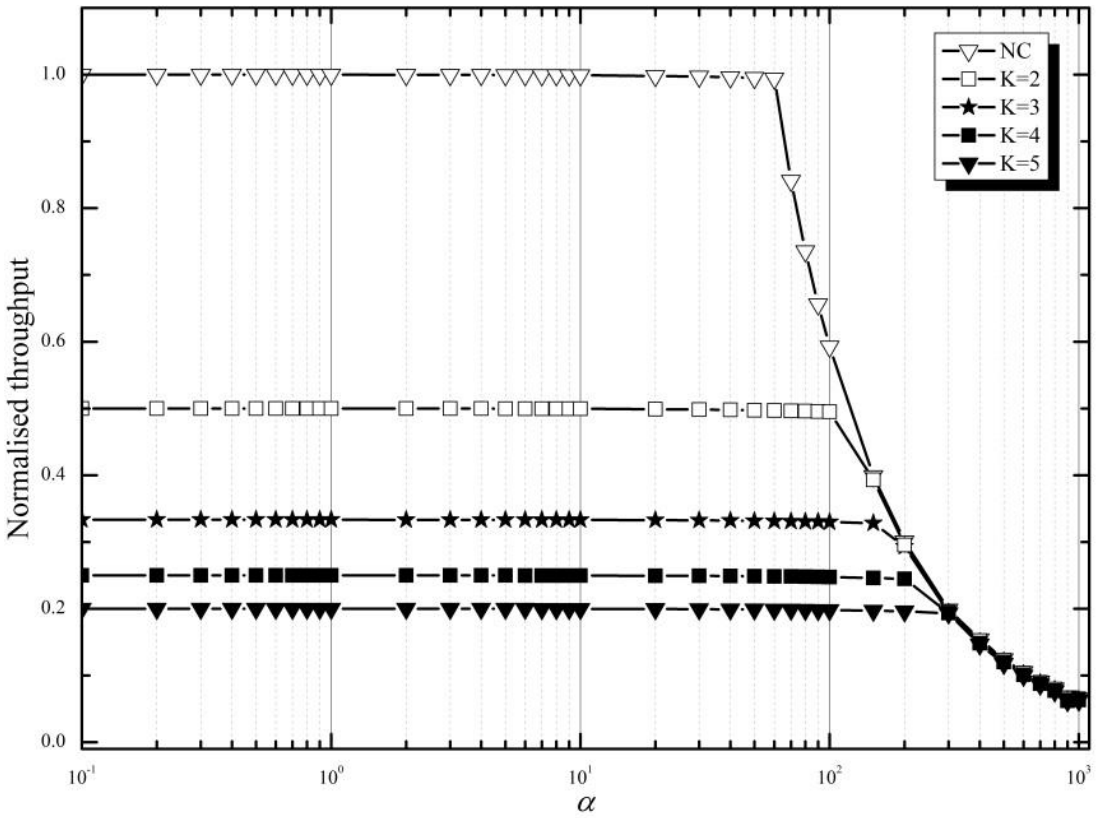


Figure 5.8: SR and NC-SR throughput when bandwidth is 10 kbps

From the performance results, it can be seen that, by adopting NC in an intermediate node with several incoming links addressing the same output port, the throughput performance becomes noticeably improved, especially in error-prone and low channel capacity.

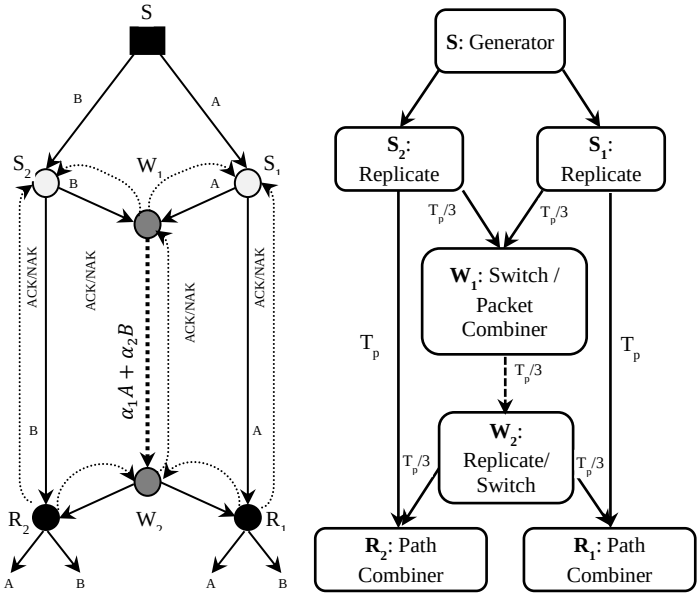
5.3 Two-Sink Multicast Scenario

The prior work is expanded here by investigating the throughput performance of the *butterfly* multicast network discussed in Chapter 2, applying SR ARQ in conjunction with a NC scheme. The basic NC technique is adopted, where a number of packets are encoded via an encoding vector and sent as one coded packet. The total delivery delay can thereby be reduced, and save the required number of ACK/NAK signals

compared to traditional SR ARQ. In [54], the throughput of the butterfly network is investigated in the context of the three ARQ schemes under different *Packet Error Rates* (PER). The results lead to the conclusion that SR ARQ is capable of attaining the highest throughput among the three ARQ schemes. The research emphasis of the following study is therefore placed upon investigating the throughput of error-prone *butterfly* networks under various propagation times, and compared with the transmission times.

5.3.1 The communication model

The model designed is composed of a set of sending, receiving and intermediate nodes. All nodes are connected through a communication link which employs the SR ARQ protocol as its error control scheme. Moreover, each link has an adjustable propagation time delay, and a configurable error pattern.



(a) The multicast SR ARQ model (b) Simevents-based model

Figure 5.9: The studied multicast model

As shown in Figure 5.9 (b), sender S advances a replica of its packets to the input port of both senders S_1 and S_2 using the *Replicate* box. The nodes W_1, R_1 and R_2 are equipped with two receivers; the output ports of S_1 and S_2 are directly connected to

receiver R_1 and R_2 respectively, and the other is connected to one of W_1 receivers. The receivers check and acknowledge the correctly received packets by sending ACK/NAK signals through the feedback channels as shown in Figure 5.9 (a). Receivers R_1 and R_2 are provided with a *Path Combiner* box which combines several incoming streams into one stream; this allows both receivers to calculate the throughput value at the outgoing links. The major infrastructure difference between the traditional SR ARQ and NC-SR ARQ scenarios is the type of the intermediate node, W_1. While a typical *switch* is used in the traditional SR ARQ, a *Packet Combiner* box is chosen to represent W_1 in the NC-SR ARQ system. The switch routes the incoming packets according to certain switching criteria. The *Round Robin* switching criterion is the choice for the studied model; after each departure, the switch selects the entity input port adjacent to the previously selected port [9].

The incoming packets from node S_1 are labelled with an identification number (ID) which distinguishes them from those packets coming from node S_2. Using the ID number, node W_2 is capable of routing the packets coming from (W_1, W_2) to their designated destinations, i.e. R_1 and R_2. On the other hand, the *Packet Combiner* acts like a coding stage, where it waits for all packets to be present at its incoming links to produce one combined packet, termed as *coded packet*. This packet is advanced to W_2 through the intermediate link, which in turn makes a copy of the coded packets using the *Replicate* box, and advances them to both R_1 and R_2. If W_2, in the SR ARQ scenario, is considered as merely a relay station, receivers R_1 and R_2 receive extra copies of packets A and B, respectively. This will incur consequent throughput wastage at the receivers; however, results show that this wastage becomes negligible when the system is operating at a high data rate, where the transmission time delay has a very small value.

Nodes at the transmitting side are provided with a FIFO queue, which controls the number of unacknowledged packets on the transmission line. Similarly, receivers maintain a queue of the same size to store out-of-order, correctly received packets, until receiving any requested low-sequenced packet leading to a sequential submission of a batch of packets to the upper layer.

5.3.2 Performance analysis

The modelled *butterfly* network, shown in Figure 5.9(a), can be described by a finite directed graph G(V, E) where *V* is the set of nodes and *E* is the set of edges. The analysis of the unicast model is again followed here since the unicast intermediate link in a *butterfly* network dominates the throughout performance due to its location in the network.

The source messages are generated by source $S \in V$ and transmitted through the network to each node $\{R_1, R_2\} \in V$, referred to as the sink nodes. The function of S_1, S_2, R_1, R_2 and W_2 is identical to the conventional "store-and-forward" nodes. Thus, packets travel without coding on (S_1, R_1) and (S_2, R_2) links. The intermediate node, W_1, has the option to encode blocks from the incoming links (S_1, W_1) and (S_2, W_1), and sinks R_1 and R_2 are capable of decoding according to the same method applied in W_1. The source *S* sends one *generation* at a time, composed of two *n-dimensional* packets A and B. All edges are assumed to have the same capacity *C*, and noise can be applied on each edge separately. With the implementation of NC using (α_1 and α_2) as encoding coefficients, the link (W_1, W_2) can transmit coded blocks $\alpha_1 A + \alpha_2 B$ of the same size *n* to both R_1 and R_2, saving half of the transmission time needed at W_1 for each transmission. Thus, the sink R_1, for instance, can extract the original packets by solving the following equation:

$$R_1: \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \alpha_1 & \alpha_2 \end{pmatrix}^{-1} \begin{Bmatrix} A \\ \alpha_1 A + \alpha_2 B \end{Bmatrix} \quad (5.10)$$

Hence, equations (5.8) and (5.9) are applicable in the multicast scenario, since it has an intermediate link employing NC. The coding nodes must satisfy the inversion condition by ensuring a nonzero positive value of the decoding determinant.

5.3.3 Simulation results for the multicast model

The study focuses on the throughput performance in different bandwidth and error pattern scenarios, and seeks to examine the usefulness of NC in these scenarios. Unless otherwise stated, entities are generated at the source node S as data packets at a rate of 10 per second with length of 1000 bits, and the transmitter forward queue size is set to fit 100 packets.

A. Different communication bandwidths scenario

In this scenario, the simulated model is analysed based on error-free channels with 100kbps bandwidth. Therefore, with a packet length of 1000 bits, the transmission time required for a node to emit one packet onto the transmission line is equal to 0.01 seconds.

Figure 5.10 shows that NC achieves throughput gains of between 1.35% and 18% over the traditional SR ARQ, when employed in one communication link. This is because the packets do not have to wait at the intermediate nodes to be forwarded through the shared link. Figure 5.11, however, shows that the multicast network suffers from underutilization of the available bandwidth when operated at channel rates over 10 kbps, this necessitates the adoption of NC in low-bandwidth communication links to gain the greatest benefit from its characteristics. Numerically, NC almost doubles the throughput (96% gain) over the useful range of α, up to a value of approximately 50 before the performance degrades significantly.

At this point, the gain starts to converge at a certain point where, for both scenarios, queue W_1 becomes full and hence starts to regulate the data rate instead of the transmitter S. The aforementioned advantage of NC in a low bandwidth environment has already triggered some research efforts in underwater sensor networks where the low bandwidth, long propagation delays and high error probability are the main characteristics of the channel [57, 97]. Similar tests conducted on 10 and 1 Mbps channels showed that NC provides little improvement in terms of the throughput gain.

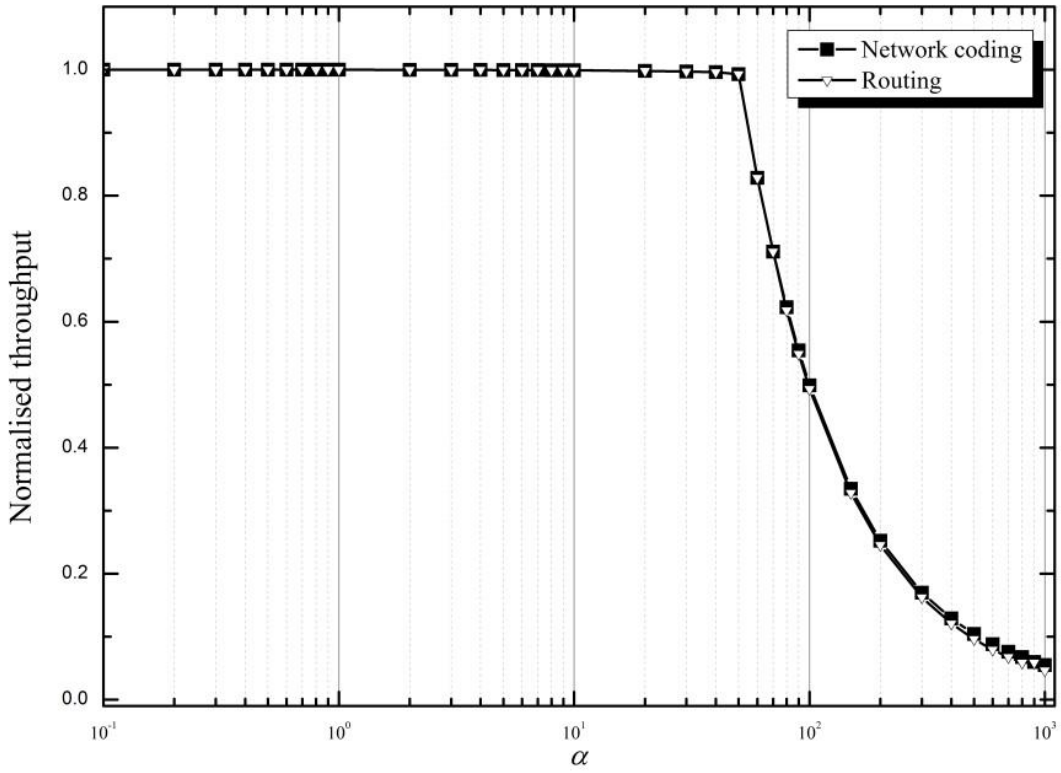


Figure 5.10: A normalized throughput of 100 kbps network bandwidth

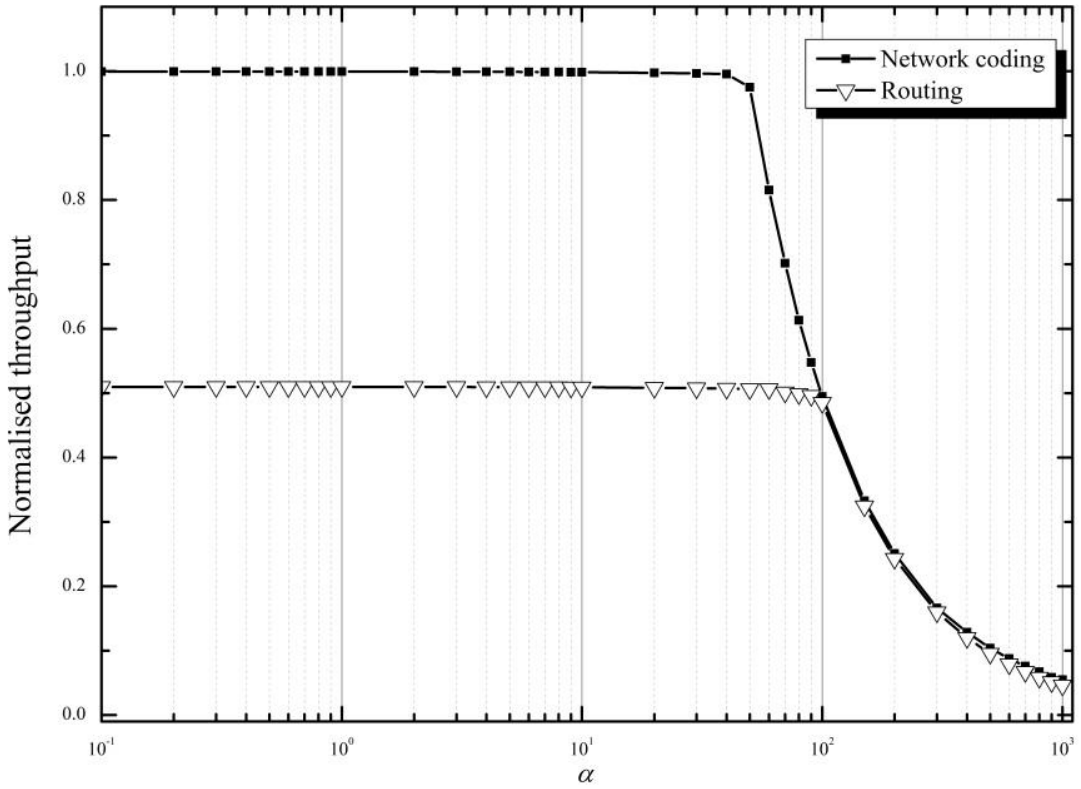


Figure 5.11: A normalized throughput of 10 kbps network

B. Different error patterns scenario

The case where the network channels are not necessarily error-free is considered, and packet errors with a probability of 0.1 and a bandwidth equal to 10 kbps are applied. The comparison is carried out first by injecting errors at the coding link (W_1, W_2) and then at the non-coding link (S_1, R_1).

- Errors occur at the *coding* link (W_1, W_2):

In the routing case: original packets are *equally* subject to errors, leading to no difference in throughput performance between R_1 and R_2.

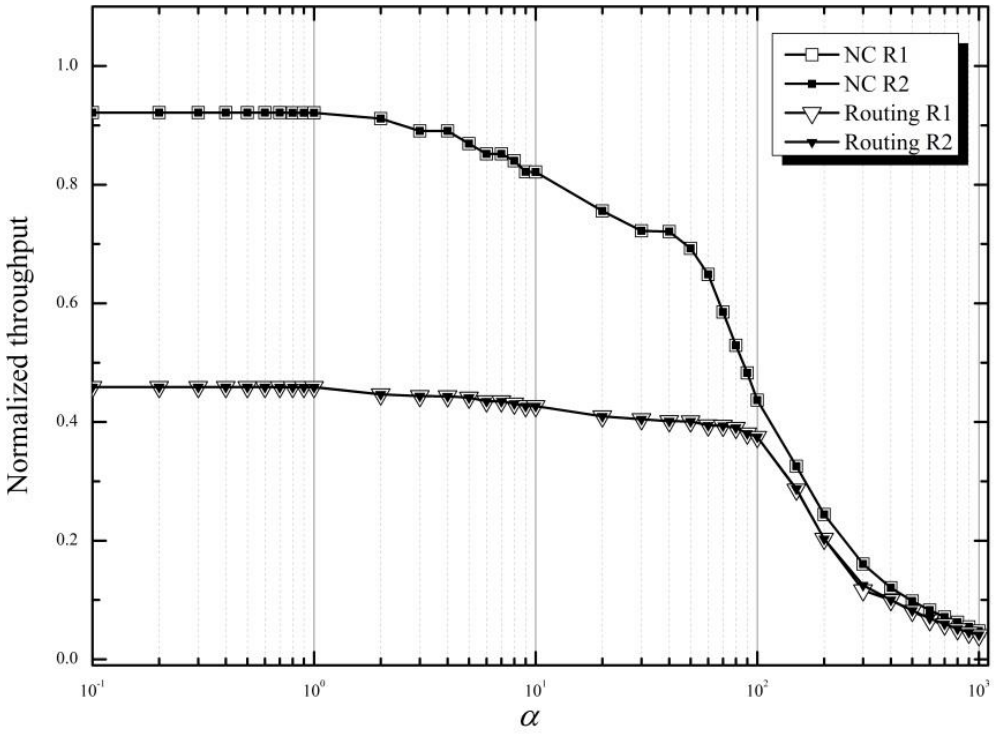


Figure 5.12: Error at a coding link

In the network coding case: because only the coded blocks are transmitted over (W_1, W_2), both R_1 and R_2 will have the same throughput value. Shown in Figure 5.12, NC offers double the throughput (101% gain) over the useful α range up to 50, which shows the merit of mixing at the intermediate node.

- Errors occur at the *non-coding* link (S_1, R_1):

In both Routing and Network coding: R_1 requests a retransmission for the corrupted packets, which leads to a degradation in R_1 throughput performance. Additionally, for both scenarios, the data rate of the (W_2, R_1) and (W_2, R_2) links will be equal. However, its value will be higher in NC, due to the merit of mixing as shown previously in Figure 5.11. When the factor α exceeds 10, NC enables the throughput to remain higher than the routing case, with gains of approximately 20% until the general decline at $\alpha = 90$ as shown in Figure 5.13.

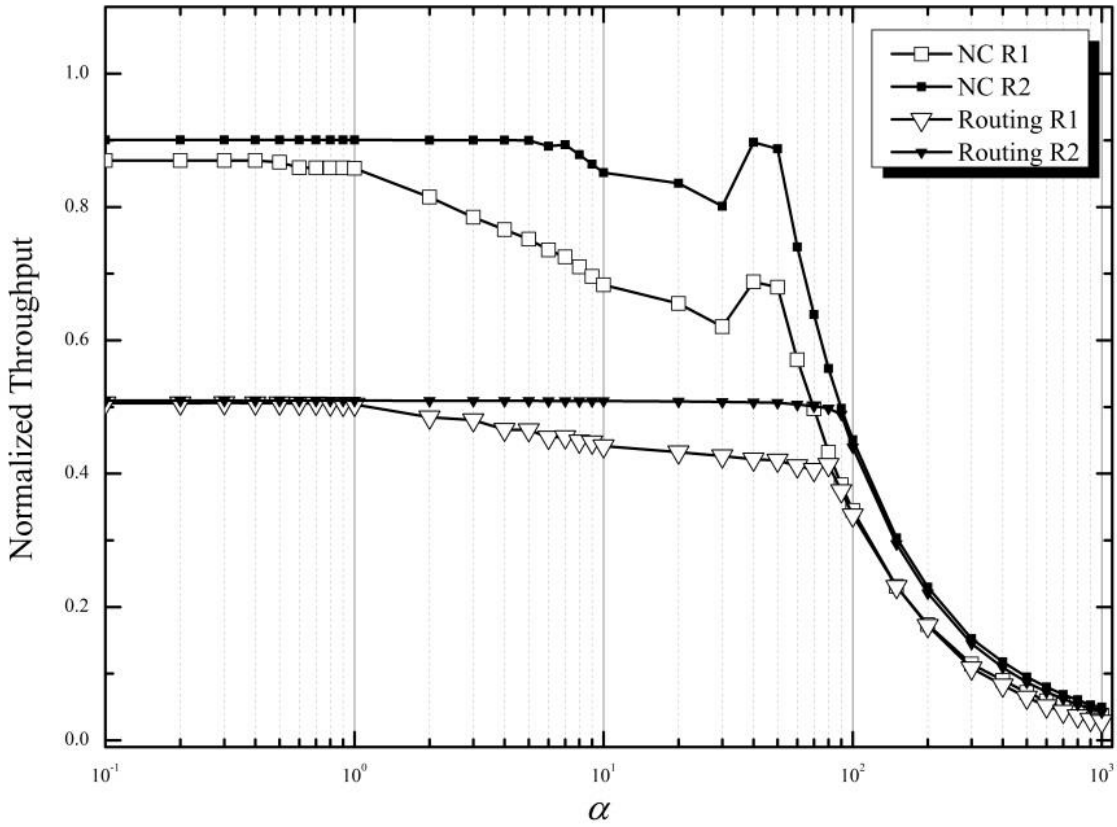


Figure 5.13: Error at a non-coding link

5.4 Conclusions

NC offers the prospect of enhancing throughput from its algebraic packets combination, which reduces the number of packets needed to transmit information. This chapter has evaluated the throughput performance of SR ARQ in conjunction with NC, in both unicast and multicast scenarios. Under both error-prone and low-bandwidth scenarios, NC has shown a noticeable improvement to conventional SR ARQ throughput performance. The Unicast scenario was studied through an intermediate link with various numbers of incoming links. The results show that NC becomes more helpful in an error-prone environment, with α having a high value, and also in a low-capacity channel with α having a low value. In the multicast scenario, NC delivered a significant SR ARQ throughput enhancement (approximately 100%) at the lower bandwidth. It is evident that NC can be utilized

to reduce the bottleneck which occurs at a shared server when a number of packets addressing the same destination compete to be processed. When packet errors occur at a non-coding node, NC also extends the useful working range from $\alpha \geq 10$ to $\alpha \leq 90$.

This impact becomes more evident once NC is employed over a properly designed network, where several links are required to apply NC. In the following chapter, NC in a three-sink multicast network is implemented, based on SR ARQ as an error scheme. The analysis is enhanced in more sophisticated communications links where Additive White Gaussian Noise (AWGN) and a Cyclic Redundancy Check (CRC) algorithm are adopted as a linear addition of noise, and an error detection mechanism, respectively.

Chapter 6

Improved Selective Repeat Protocol in a Network Coding Scheme

6.1 Introduction

Chapter 5 has explored the benefits of Network Coding (NC) in both unicast and *butterfly* multicast networks, employing a SR ARQ protocol at each link. The aim of this chapter is to model a NC-based *extended butterfly* network employing the SR ARQ protocol; in this network, three sources send their packets to three sinks through two intermediate nodes and shared links. A detailed simulation model for the network topology and elements necessary to construct a NC-based network infrastructure with the SR ARQ protocol are proposed.

In this analysis, deterministic NC is adopted [22, 36]; this requires full network topology knowledge stored *a priori*. The encoding function remains fixed and known to every node, eliminating the random NC overhead and lowering the operational decoding cost. As described in Chapter 5, with the use of NC, several data packets travel as one coded block, offering the opportunity for a single acknowledgment to provide delivery status feedback to the transmitter concerning several packets.

This chapter deploys both traditional and NC SR ARQ-based multicast scenarios based on [1, 77] - the latter is termed NC-SR ARQ. A range of network conditions is considered in order to determine where NC network efficiency gains can be

achieved. Several SimEvents aggregation blocks are utilized to examine each sequential transmission between the communication links under various network conditions. In addition, a comprehensive parametric-based study of the multicast network is conducted to elucidate important design factors in the performance of NC-SR ARQ.

The model implemented provides throughput performance results for both routing-based and NC-based scenarios, subject to various data rates, signal to noise ratios and transmission delay values. The results show that NC offers a throughput advantage of up to 48% in error-free conditions, and an average of 50% in error-prone scenarios. The results also demonstrate, for different link characteristics, the conditions in which coding opportunities can be maximized. In addition, the comparative results show that NC is capable of producing a throughput increase of between 13% and 50% in networks with varying transmission delays or packet length. This chapter also proposes a synchronized NC scheme which exhibits some degradation in NC throughput performance. However, in the presence of low SNR and high packet transmission delay, a throughput increase of between 78% and 86% can still be achieved.

The chapter is organized as follows; Section 6.2 explains the NC scheme adopted and the multicast topology. The detailed network model and implementation for NC architecture in SimEvents is presented in Sections 6.3 and 6.4. Section 6.5 shows a detailed comparison of the simulated traditional SR and NC-SR ARQ performance results. Finally, Section 6.6 concludes the chapter.

6.2 System Model

As shown in Figure 6.1, the 14-node network is considered as the basis for the investigation, as this represents an extension to the well-known 'butterfly network' architecture. Each node in the proposed network, excluding the source node, has a finite queue capacity of W entities (packets). The communication network is represented by a finite acyclic directed graph G= {V, E}, where V is the set of nodes and E is the set of edges. In Figure 6.1, the source file is divided into generations, each of which contains k=3 packets [26].

The symbol X(e) sent on link e = {(I_1, I_3), (I_2, I_4)} ∈ E, is expressed as a combination (function f_e) of packets entering the intermediate nodes I_1 and I_2:

$$X_e = f_e\{X_\varepsilon\}_{\varepsilon \in \mathrm{Input}(I_1, I_2)}$$

The coded blocks X_e are relayed from the outgoing ports of the intermediate nodes towards the sink nodes $R_i \subset V$, which are able to retrieve the original packets by solving the set of equations $\{f_e(b_1, b_2, \ldots b_k) = X(e)\}_{e \in input\,(R_i)}$. Given a local number δ of the incoming links which are eligible for coding at the intermediate nodes, I_1 and I_2 deterministically choose a vector set of coding coefficients $\boldsymbol{C} = [c_1, \ldots, c_\delta]$ to produce a linear combination X of the original data blocks [26].

$$X(e) = \sum_{i=1}^{\delta} b_i c_i \tag{6.1}$$

A Cyclic Redundancy Check (CRC), defined for any sequence of bits from a binary polynomial, is employed to preserve transmission data integrity [4]. Since the packet size in the model studied is less than 2^{15}-1 bits, the CRC-16 scheme (generated in MATLAB using $g(x)=x^{16}+x^{14}+x^2+1$) can detect all odd numbers of errors and also double errors [5].

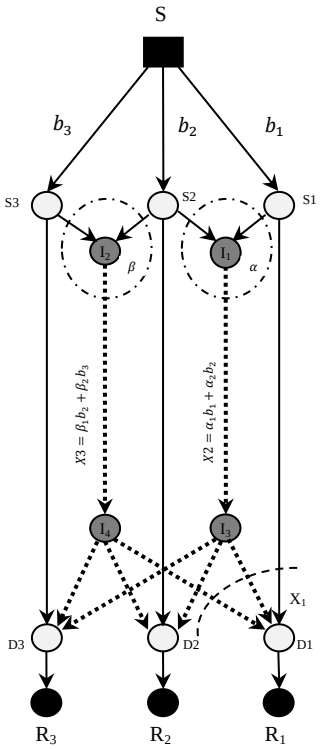


Figure 6.1: A multicast network distributing three blocks with NC

At the transmitter, the CRC is calculated for each packet generated and placed at the tail of the packet for subsequent use at the receiver. The receiver queue may contain out-of-order acknowledged packets, which necessitates the use of a sorting algorithm to deliver the packets in sequence to the receiver upper layer. The sorting algorithm, presented in Chapter 3 and utilised in Chapter 5, is also adopted here.

In Figure 6.1, the dotted lines carry the coded blocks *X*. All sinks R_1, R_2 and R_3 are capable of decoding using the same method applied in I_1 and I_2, and use the same coefficients $\{\alpha, \beta\} \in C$, which are known throughout the network.

The source *S* sends one generation of three packets [b_1, b_2, b_3] at a time. Both intermediate nodes I_1 and I_2 receive b_2 and pass it to all sinks (R_1, R_2, and R_3) through the links (I_1, I_3) and (I_2, I_4). Generally, both nodes I_1 and I_2 can serve R_1 at this point, although they may end up transmitting the same packet, b_2, to node R_1, since there is no coordinated transmission between them, which results in a wastage

of the bandwidth of nodes I_1 and I_2. It could be assumed that I_3 and I_4 are merely relay stations, but they are imbued here with routing capabilities eliminating packet duplication at the receivers.

If NC is implemented, nodes I_1 and I_2 can transmit coded blocks $X_2 = \alpha_1 b_1 + \alpha_2 b_2$ and $X_3 = \beta_1 b_2 + \beta_2 b_3$ to all receivers as shown in Figure 6.1. Therefore, receivers R_1, R_2 and R_3 receive sufficient information to perform decoding and to re-construct the original packets. Thus, for instance, sink R_1 can extract all packets [b_1, b_2, b_3] by solving the following equations:

$$D_1: \begin{Bmatrix} X_1 \\ X_2 \\ X_3 \end{Bmatrix} = [b_1\ b_2\ b_3] \begin{pmatrix} 1 & \alpha_1 & 0 \\ 0 & \alpha_2 & \beta_1 \\ 0 & 0 & \beta_2 \end{pmatrix}$$

$$D_1: \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{pmatrix} 1 & \alpha_1 & 0 \\ 0 & \alpha_2 & \beta_1 \\ 0 & 0 & \beta_2 \end{pmatrix}^{-1} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} \tag{6.2}$$

The receiver receives a collection of coded blocks so that the corresponding encoding vector is of rank *k*. The original blocks can be reconstructed by solving the *k* linear equations that represent coded blocks belonging to the same generation. Gauss-Jordan elimination can be used as a progressive decoding process [90], where decoding occurs as coded blocks are being received; in Chapter 4, a ($k \times k$) Vandermonde matrix [45] was used to produce *k* coded packets and ensure a non-linear relationship between them. However, in this analysis, the coding coefficients are chosen from the finite field GF (2) to produce one coded packet. Therefore, both intermediate nodes perform an add operation over GF (2), which can be represented by the logical operation XOR. Hence, R_1 can extract b_2 by performing XOR (X_1, X_3), with the result that X' and b_3 can be extracted using XOR (X', X_3). The same analysis

performed in Chapter 5 is applicable here as well. The link utilization in each link in NC-SR ARQ, and is given by:

$$U = \begin{cases} 1 - PER & W \geq 2\alpha + 1 \\ \dfrac{W(1 - PER)}{2\alpha + 1} & W < 2\alpha + 1 \end{cases} \tag{6.3}$$

U_{tr} in traditional SR ARQ can be given by:

$$U_{tr} = \begin{cases} (1 - PER)^k & W \geq 2\alpha + 1 \\ \dfrac{W(1 - PER)^k}{2c} & W < 2\alpha + 1 \end{cases} \tag{6.4}$$

A comparison between equations (6.3) and (6.4) indicates that U_{tr} decreases exponentially with increasing values of *k*, which means that, as the number of incoming node links increases, the potential performance improvement which NC can offer also increases.

6.3 NC-SR ARQ Architecture in SimEvents

6.3.1 The multicast network modelled

The communication system studied requires a packet source to send its messages to three common destinations over the extended butterfly network shown in Figures 6.1 and 6.2. The SimEvents model designed is composed of a set of sending, receiving and intermediate nodes. All nodes are connected through communication links that employ the SR ARQ protocol as their error control scheme, and each link has a configurable propagation time delay and an adjustable error pattern. As an enhancement to the previous work in Chapter 5, the latter is drawn from an Additive White Gaussian Noise (AWGN) channel, specified in the relevant Simulink block by the signal to noise ratio (SNR). Independent sources S_1, S_2 and S_3 receive messages from a common source S at a specified data rate, and must convey these packets to sinks R_1, R_2 and R_3. For classic routing, nodes I_3 and I_4 duplicate the received

packets from nodes I_1 and I_2, and forward the copied packets to nodes D_1, D_2 and D_3. Employing NC, nodes I_1 and I_2 sum their incoming streams of packets over $GF(2)$, and relay the coded messages to nodes I_3 and I_4 for duplication.

As in Chapter 5, receivers are provided with a *Path Combiner* box in the simulation, which combines several incoming streams into one stream. This allows both receivers to calculate the throughput value at the outgoing links. At the intermediate nodes, I_1 and I_2, in SR ARQ, a *switch* routes the incoming packets according to the *Round Robin* criterion, and an identification number (ID) is added to incoming packets to assist with onward routing without duplication. A *Packet Combiner* is utilized in NC-SR ARQ, which acts as a coding stage, waiting for all packets to be present at its incoming links to produce a single coded NC packet. In NC-SR ARQ, these coded packets are advanced through the intermediate links to I_3 and I_4, which in turn make a copy of the coded packets using the *Replicate* box. The nodes I_3 and I_4 are also enhanced with routing capability through the addition of a *switch* after the *replicate* box to route packet using the ID provided, which avoids throughput wastage by preventing packet duplication.

Nodes at the transmitting side maintain a FIFO queue with a size W=120, which controls the number of un-acknowledged packets on the transmission line. Similarly, receivers maintain a queue of the same size to store out-of-order but correctly received packets, until receiving any requested low-sequenced packet; this leads to a sequential submission of a batch of packets to the upper layer.

In all scenarios, the receivers are assumed to be capable of decoding in the absence of synchronization, i.e. coded packets are not required to wait for the arrival of their corresponding packet in the other flow, but rather they will arrive at the receivers in a random order. This is then enhanced by proposing a scheme by which coded

packets are guaranteed to arrive simultaneously and in pairs; the effect of this method on the NC performance is studied. To avoid peculiarity, and since some packets between source and destination take three hops, the total propagation time of these packets is set to vary between the value for a single hop by, setting ε=1/3 (for the direct source-destination link) to ε=1 as shown in Figure 6.2.

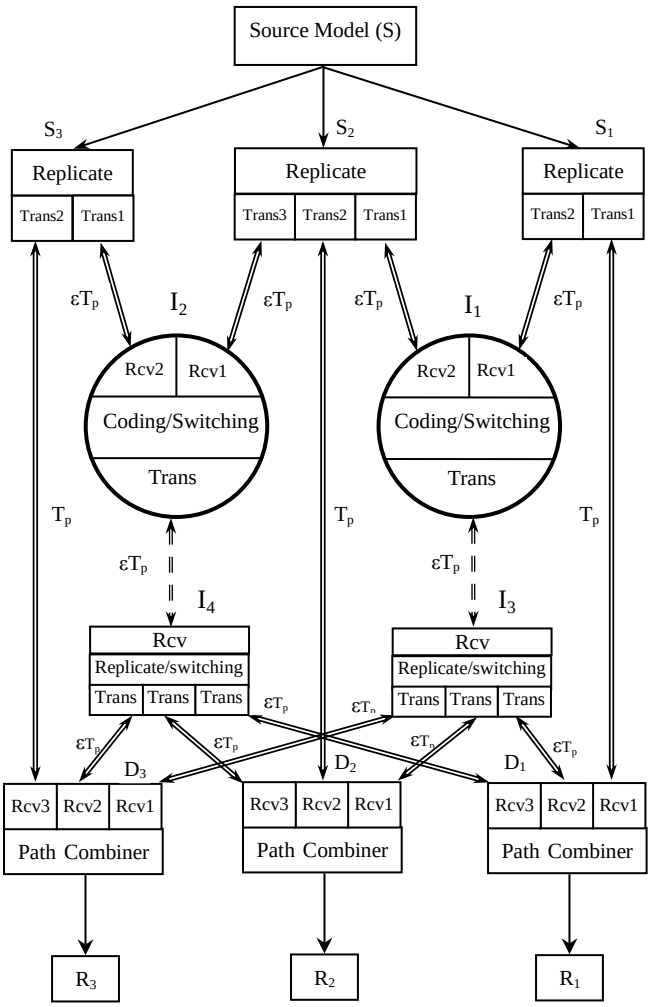


Figure 6.2: An extended butterfly network in SimEvents

6.3.2 Block libraries used for NC-SR ARQ simulation in SimEvents

The blocks necessary in SimEvents for NC-SR ARQ modelling include *inter alia*:

12. ***Time-based Entity Generator***: This block is used to represent the source node, by generating entities in the form of packets. The inter-arrival time is chosen to vary according to a constant, uniform and exponential distribution.
13. ***Set Attribute***: Data, CRC bits and sequence numbers are assigned to entities as attributes using this block; these attributes can be fetched back using the *get attribute* block.
14. ***Entity Departure Counter***: This block is used to compute the number of entities that have departed from a block, and writes this number to a named packet *attribute* field.
15. ***CRC-N Generator***: CRC bits are generated and then appended for each input data frame using this block.
16. ***CRC-N Syndrome Detector***: Receivers are capable of detecting errors using this block, after selecting the "CRC-16" method. Upon receiving the checksum result, the receiver can send either ACK or NAK signals.
17. ***FIFO Queue***: The transmitter must maintain a queue of a specific capacity in order to temporarily store the sent-but-unacknowledged packets. This block can store an infinite number of packets in a first-in, first-out (FIFO) fashion. The queue retains the entity if the OUT port is blocked, and attempts to output an entity if the path is available. This block provides several metrics, such as an average number of entities in the queue over time, a sample mean of the waiting times in the queue for all entities that have departed, and the number of entities that have experienced a timeout.
18. ***Priority Queue***: In order to re-sort out-of-order packets at the receiver, the priory queue offers the choice of sorting entities according to the value of an attribute, either in ascending or descending order.

19. ***Enabled Gate***: The simulated system uses this block to control the packet flow through its input ***en***; the gate opens when ***en*** is positive, and closes when it is zero or negative. This block is used to allow sorted packets, at the receiver, to be advanced to the upper layer once an awaited packet arrives.
20. ***Replicate***: For the sake of simulation simplicity, packets must follow two or more paths. Therefore, this block is used to output a number of specified copies of the arriving entity.
21. ***Path Combiner***: This block is used when the merging of entity paths is required, for example, merging the paths of the new and re-transmitted packets at the transmitter, or allocating packets from different links into a receiver path.
22. ***Switch***: Based on a specified switching criterion, this block is able to guide packets according to certain attributes, which can be specified by the user.
23. ***Server***: This block is used to serve one or more entities or to delay them for a period of time - called the service time. The service time for each entity is computed when it arrives; during the service period, the block is said to be serving the entity that it stores.
24. ***Function-Call Subsystem***: Function-call signals can conditionally invoke this block to execute another user-designed function within the block.

6.4 NC Implementation in SimEvents

6.4.1 Source model

The traffic source is an entity generator model that can emit entities with constant, uniform and exponential inter-arrival distributions. The entities generated are then assigned 100 bits of payload data to construct a frame. Although the data packet length is chosen according to simulation constraints, the transmission delay will be

varied to overcome this limitation. A CRC-N generator is utilised inside a function-call subsystem, the execution of which is determined by a trigger from the *Entity Departure Function-Call* Generator. Each input data frame is appended with 16 CRC bits, and then modulated using binary phase-shift keying (BPSK). The source delays the generation of packets if the succeeding transmitter queue is temporarily filled. There are two classes of generated packet, namely those with a one link to their destination and those sharing links with other packets of a common destination. The complete source model is shown in Figure 6.3.

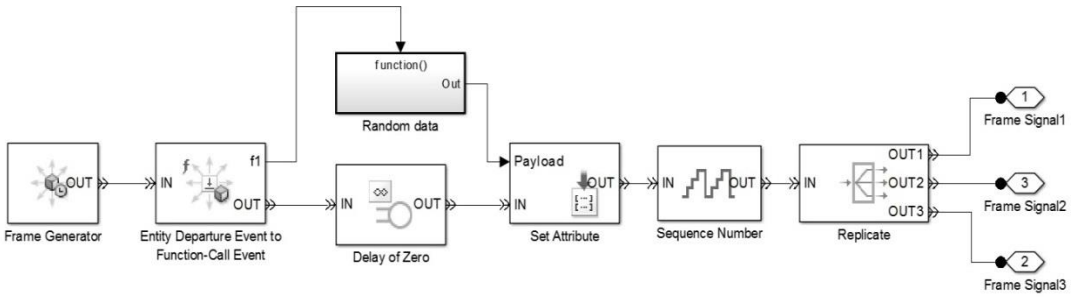


Figure 6.3: A representation of a source node in SimEvents

6.4.2 Traditional intermediate node

As shown in Figure 6.4, the traditional node is equipped with two receivers and one transmitter. Packets coming from different links are labelled with different ID numbers, with the purpose of later helping nodes I_3 and I_4 to route them to their intended destinations. Each packet is then assigned a sequence number to allow receivers in I_3 and I_4 to sort out of order packets and correctly request retransmission for corrupted packets.

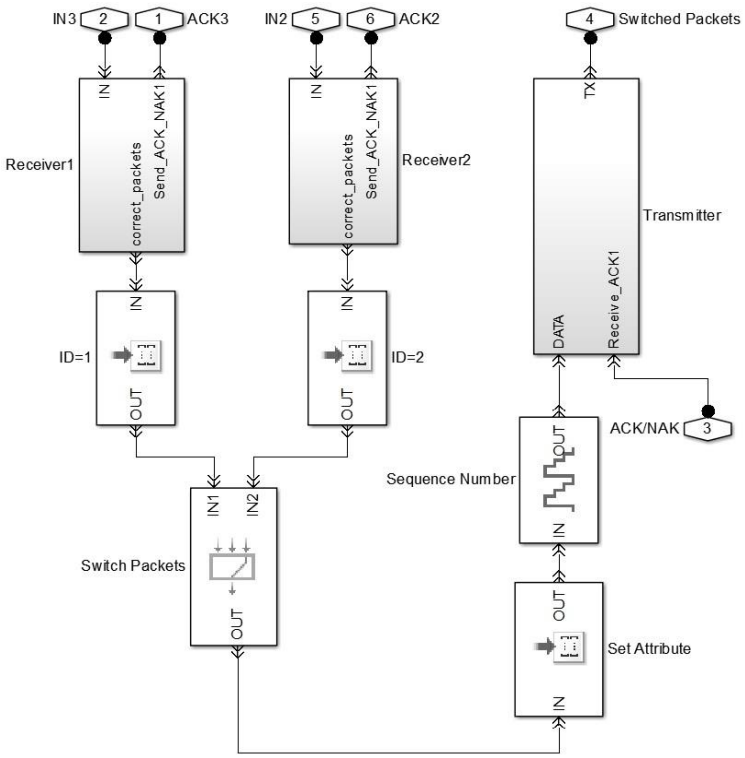


Figure 6.4: A traditional intermediate node modeled in SimEvents

6.4.3 NC intermediate node

A NC node must possess a structure which matches packets from different links to be suitable for NC, and also has to modify the sequence number for each encoded packet. In Figure 6.5, the NC node is composed of an entity combiner and a MATLAB script, which algebraically adds data packets modulo 2. Subsystems 1 and 2 are constructed to eliminate the *seqNum* attributes of the incoming packets, since the packet combiner block does not accept common *attributes* when the parameter "*Retain attributes in departing entity*" is selected.

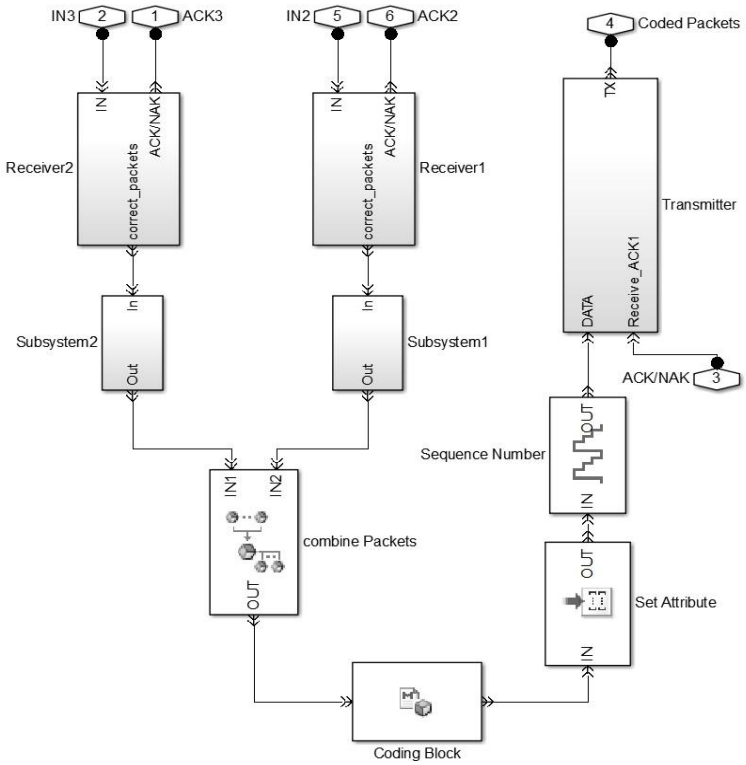


Figure 6.5: The NC intermediate node modeled in SimEvents

6.5 NC-SR ARQ Performance Analysis in SimEvents

In this section, six different multicast scenarios are investigated to reveal the network conditions where NC becomes most efficient. For clarity, the performance of the NC system is compared to that of the operation of a closely-equivalent classic-routing system, with the same physical resource allocation in both cases. The overhead for transmitting ACKs is neglected in this model, since only one bit is used to represent ACK/NAK signals. System performance is analysed by investigating the maximal achievable throughput when the network operates under stringent service constraints.

6.5.1 Assumptions and parameter setup

The results obtained from the traditional SR and NC-SR ARQ calculations are denoted by “throughput” and “throughput (NC)”, respectively. In all scenarios, unless otherwise stated, the data source generates entities at a rate of 20 packets per second, with a length of 100 bits, and the transmitter forward queue size is set to fit

120 packets. Adopted from [80], the propagation delay is set at T_p= 0.015 seconds, from a source to each destination, as previously shown in Figure 6.1.

6.5.2 Performance results

In this section, the network performance is assessed for both SR and NC-SR ARQ scenarios under various network conditions, and is analysed with identical link capacities and adjustable noise levels. Firstly, the SR ARQ efficiency is studied as the parameter α adopted from [77] varies in error-free and error-prone scenarios, with SNR=4; both scenarios are then examined under various SNR values. This is followed by investigating both scenarios under various channel capacities. For the first three scenarios, two extreme values of ε, 1/3 and 1, are chosen. For ε=1/3, the throughput benefit of NC in error-prone scenarios is examined, where SNR takes different values. This is followed by investigating the optimum data rate at which NC throughput gain is maximized. The final scenario investigates synchronous NC performance over traditional routing, allowing the receivers to hold early-arrived coded packets until all required packets are present. The achieved capacity gain comes from alleviating local congestion at the intermediate nodes; the maximum benefit of NC is observed to be achieved when the network is heavily utilized, and data is available to be sent at every possible opportunity.

a) **Error-free scenario**

In this scenario, the simulated model is analysed based on error-free channels, with the transmission time equal to 0.01 seconds. The values of α are set to vary from 0.1 to 1000, according to standard recommendations in [77]. Figure 6.6 shows that, as α becomes larger, NC achieves a throughput improvement over traditional SR ARQ, that rises from 3% at α = 500 to some 48% at α = 1000 when ε=1/3, and from 20% to 47% when ε=1. This enhancement originates from the fact that, with increasing

values of α, each transmitting node has to wait longer for any successive receiver acknowledgements. This causes a rapid rise in the transmitter queue length, which is ameliorated by NC, since it increases packet release speed by advancing two packets through any intermediate node.

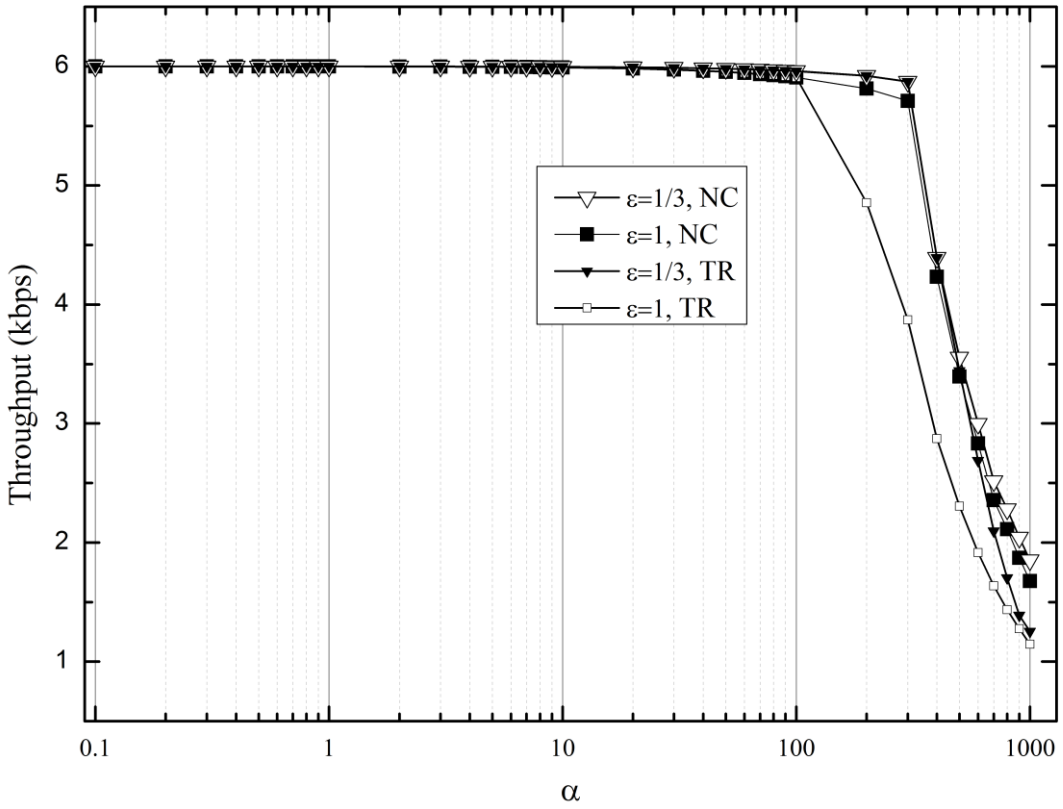


Figure 6.6: A throughput comparison of both NC and Traditional multicast network.

b) **Noisy scenario**

The case where the communication channel is not necessarily error-free is now considered. The same parameters as in the first scenario are used, but with an SNR value of 4 dB. This value, for example, is the recommended or the minimum Signal-to-Noise ratio for wireless voice and data cells operating at a low data rate of 1 Mbps [98]. Figure 6.7 shows that NC provides a typical throughput benefit of 50% in this case, over the whole range of α from 0.1 and 100; this is because the transmission

errors in succeeding links cause a queuing build-up at transmitting nodes that is alleviated by the employment of NC.

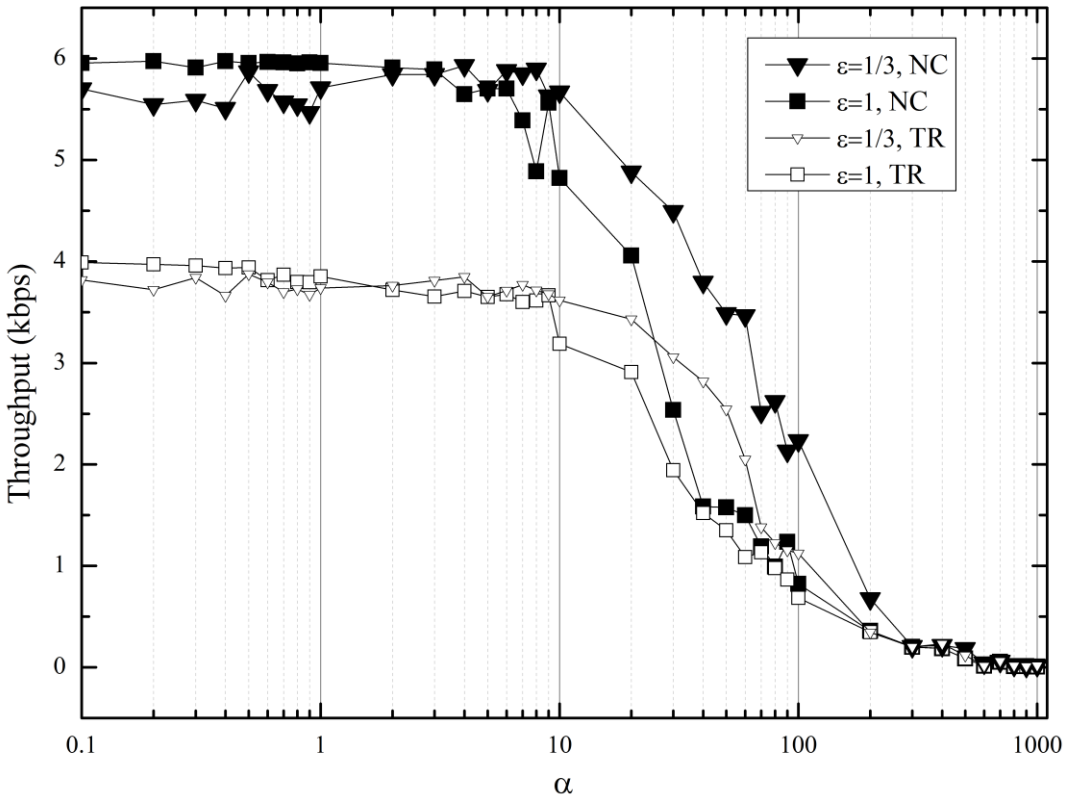


Figure 6.7: A throughput comparison of both NC and traditional multicast network in the presence of errors (SNR = 4 dB)

c) **Different channel bandwidth scenario**

Another important factor for evaluating network performance is the ratio β, of packet length to channel capacity. This ratio effectively represents the delays being incurred while packets are being emitted at each transmitting node. The subsequent packets coming from different streams may halt when β has a relatively large value. Figure 6.8 shows that, when β exceeds approximately 0.02, NC offers throughput improvements from 13% to 50%.

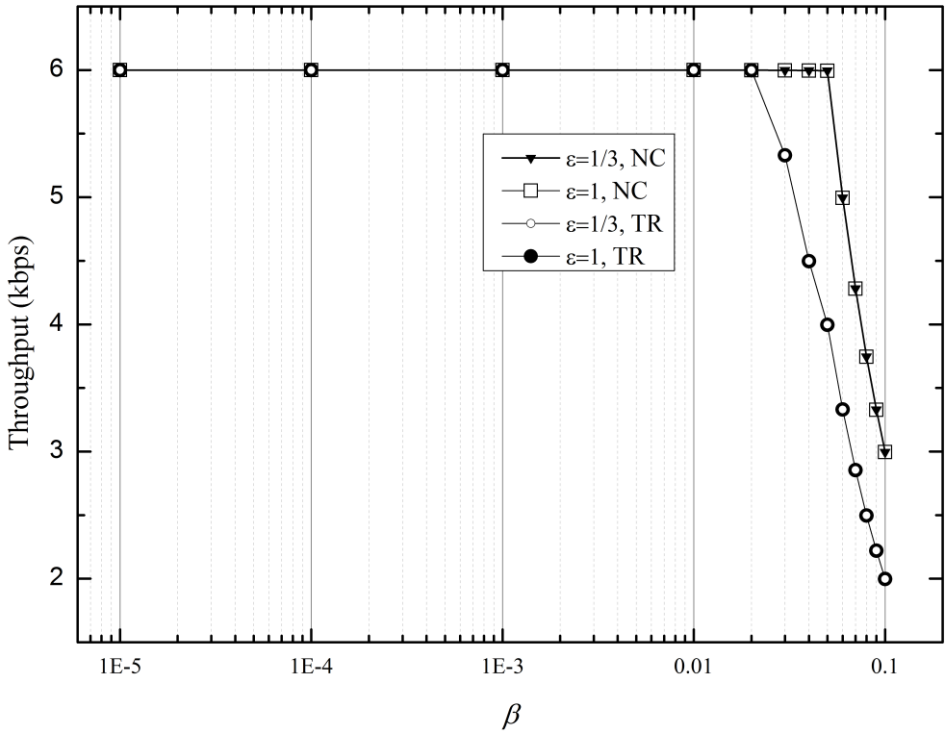


Figure 6.8: A comparison of throughput performance as a function of β

These performance improvements arise from the fact that NC always requires the presence of all packets at the intermediate nodes. Since β is large enough to halt packets at the intermediate node, incoming links using NC will have the ability to evacuate their data up to twice as fast as when traditional routing is employed, given that the intermediate nodes possess only two incoming links.

d) **Different SNR scenarios**

In this scenario, the impact of the packet transmission time T_r on throughput performance is investigated, when NC experiences different SNR values. Figure 6.9 clarifies that NC offers a throughput advantage over traditional routing in the presence of a poor SNR and larger transmission delay.

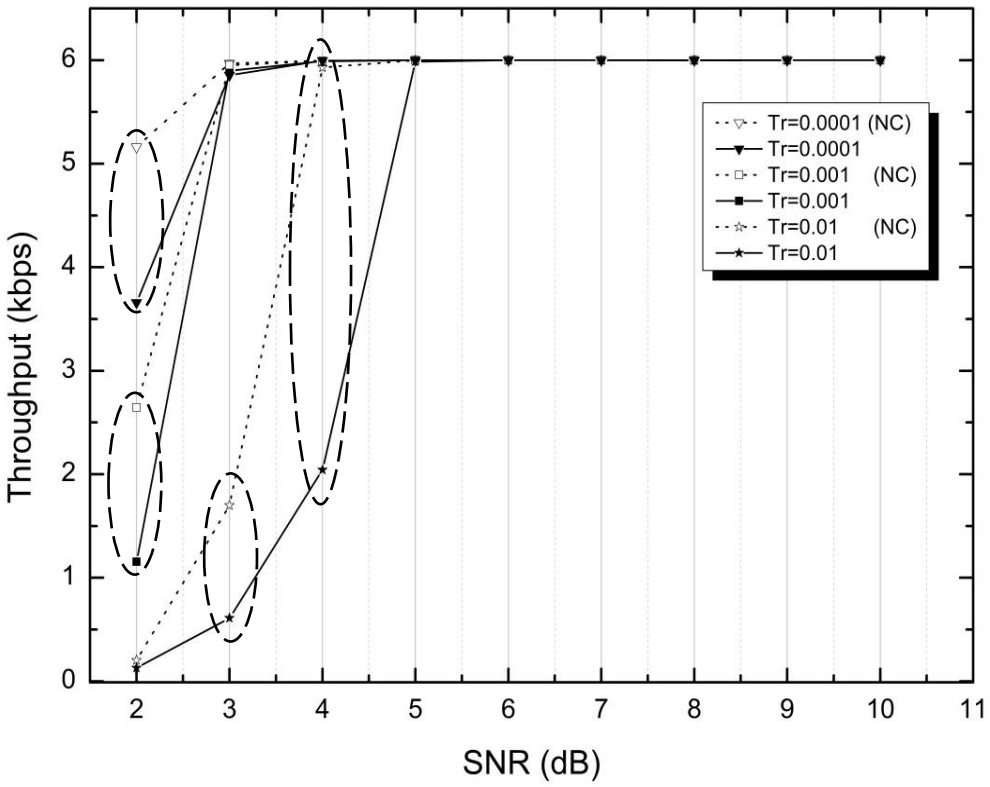


Figure 6.9: A throughput comparison between traditional and unsynchronized NC SR ARQ scheme

Since the bit error rate (BER) is inversely proportional to SNR, a low SNR increases packet loss and delays which in turn decrease throughput. The degradation in the network performance, when the SNR is low, is due to the additional constraint of the limited successful delivery of packets in succeeding links, causing the nodes' transmitters build up a backlog of packets; this highlights the need to alleviate the congestion using NC. Numerical results show that, with a low SNR (2dB), NC achieves a throughput advantage of between 41% and 129% in all scenarios. In addition, with SNR=4dB, NC achieved a 190% throughput gain when the transmission time was T_r=0.01 seconds.

e) **Different source data rates**

In order to determine the conditions under which the system is stable and can gain the greatest advantage from NC, it is necessary to understand the relationship between the traffic inter-arrival time and coding efficiency.

To increase coding opportunities at the coding nodes, a pre-set maximum delay at I_1 and I_2 can either be introduced, or define a network where packet arrival at the coding nodes is almost synchronized. During the delay period, in the first option, coding nodes perform coding when packets from all streams are present; otherwise, packets are transmitted without coding if the period expires. The second option requires deterministic and synchronized packet arrival at the coding nodes being investigated; this option is implemented by independently varying the packet inter-arrival distribution for all sources, S_1, S_2 and S_3.

A constant inter-arrival time is set with values ranging between [0.02, 0.2] on the link (S, S_1). The inter-arrival time between the generated packets over (S, S_2) uniformly takes the range [0.05, 0.15]. Furthermore, packets transmitted over link (S, S_3) are exponentially distributed with a mean value of 0.1. This arrangement provided the results shown in Figure 6.10, which demonstrates that the coding benefits are optimum when the packet inter-arrival times are close enough to maximize coding opportunities. The reason for this benefit is due to the larger inter-arrival delay (greater than 0.1), which forces the decoding block to pause until all packets are ready for coding; this causes fewer coding opportunities to occur, and consequently a lower throughput gain.

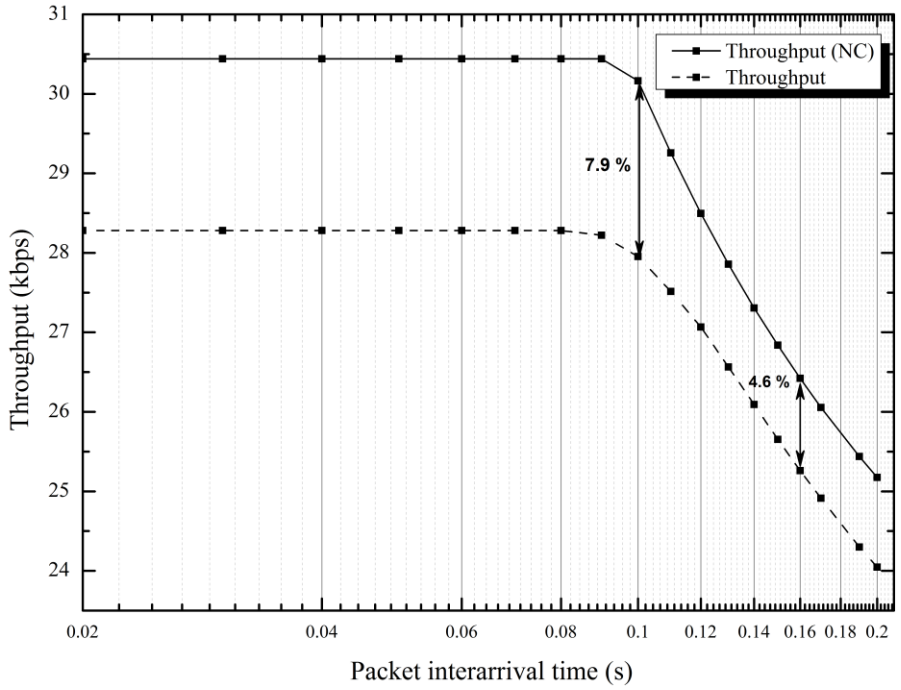


Figure 6.10: A throughout comparison at different data rates sources

f) **Synchronous NC scenario**

In unsynchronized networks, NC has shown that it offers significant improvements in throughput performance. However, it is common that, during packet transmission, a delay can occur when a packet of any generation takes a longer route to reach its destination. In an NC system, this forces the receiver to adopt a sophisticated method of synchronization, since it cannot construct the original packets unless all coded packets of the same generation are present. In this experiment, the *packet combiner* block is exploited to simulate a synchronized NC scheme. In this way, the *packet combiner* block can represent a decoding stage in real-world networks, since it waits until all necessary packets are ready for the combining operation to proceed, which fulfils the receiver requirement for decoding. Conversely, this particular arrangement will clearly cause degradation in the throughput performance, since any packet with a short path will have to be held at the receiver until all packets arrive, as shown in Figure.6.11. Nevertheless, NC still offers benefits for low SNR and high packet

transmission delay scenarios, such as underwater acoustic communications [57, 99]. Figure 6.11 indicates that NC still offers 78% and 86% throughput advantage when SNR values are 3 dB and 4 dB, respectively.

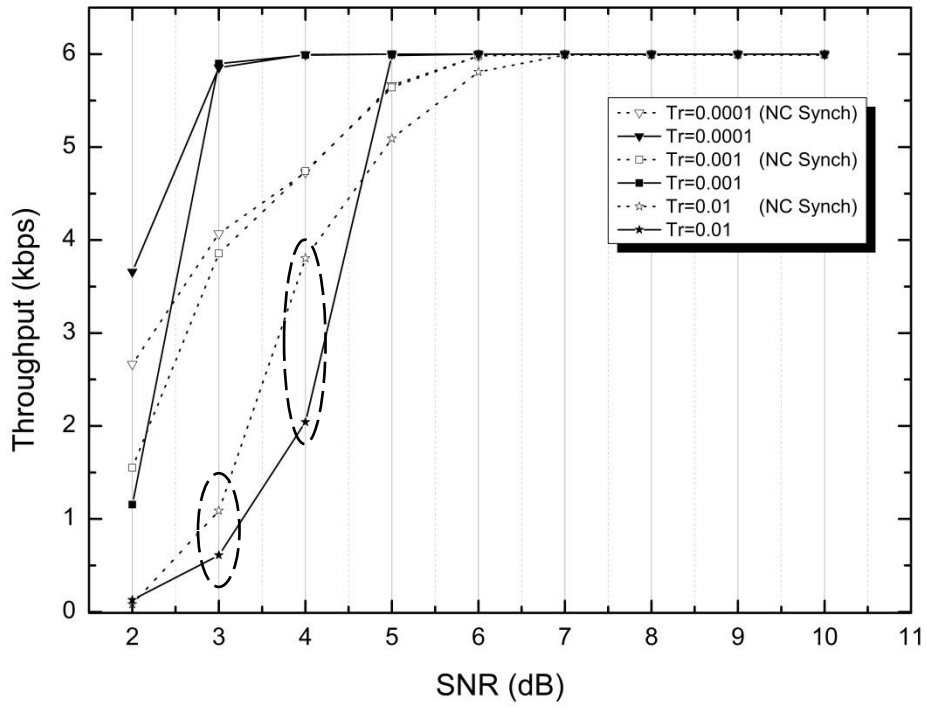


Figure 6.11: A throughput comparison between a traditional and synchronized NC SR ARQ scheme

6.6 Conclusions

This chapter has examined the performance of NC using simulation of a practical scenario to ascertain the extent to which the theoretical benefits of NC are realized. The performance of NC in a multicast network was compared with SR-ARQ-based classic routing. When the network was error-free, NC offered throughput increases of up to 48%, as the propagation time became much greater than the transmission delay, but offered no gain otherwise. However, in the presence of AWGN (and hence errors), NC offered a 50% throughput increase over a usable throughput range, i.e. before the throughput declined drastically because of queuing and retransmission delays.

Variation of the SNR in the AWGN channel showed that NC comes into its own for poor quality transmission conditions, enabling the throughput to be maintained as the SNR drops, and significantly outperforming classic routing.

The chapter also presents results for incoming packets of variable data rates; an aspect which has not been considered in the past. As one may expect, the combination of packets with dissimilar inter-arrival rates at intermediate nodes may not offer a throughput gain. It is thus essential to ensure that data sources are tuned to generate packets at rates that do not go beyond a value that guarantees NC gains. This leads to the work presented in **Chapter 7**, where a queuing model of traditional and synchronous NC intermediate nodes with Poisson packet arrivals is mathematically analysed and verified via simulation.

Chapter 7

Network Coding Based M/M/1 Queuing Model Analysis

7.1 Introduction

In Chapters 4, 5 and 6, the performance of two error control schemes under network coding (NC) has been investigated. It was concluded that the capacity gain of NC can be strongly influenced by properly synchronising the incoming packets of the NC node. However, the implementation of synchronous NC requires a thorough investigation of its impact on the relevant queuing systems, since the input buffers of coding nodes may become intractable. As shown by the literature provided in Chapter 2, the behaviour of an encoding node queue was demonstrated by [34]; it was concluded that the absence of synchronisation might seriously influence the performance of NC, i.e. the buffer size at the encoding nodes would increase to infinity.

This chapter analyses the performance of an M/M/1 queuing system at an intermediate node under synchronous NC without input buffers. The main contribution of this chapter is the derivation of the coded packets distribution, and their mean waiting time due to two and three packet streams. The results are then expanded to compare the packet loss in an M/M/1 queue limited to accommodate K packets.

A synchronized queuing model with identical Poisson packet arrivals is studied in [34]. The authors assume that packets are stored in input buffers before matching and coding are performed. They show that the output stream asymptotically follows a Poisson distribution with the same arrival rate, and that the system becomes unstable with input buffer sizes increasing to infinity. This observation has led to the proposal of a combined opportunistic scheduling and encoding (COSE) technique, in which a clearing stage is associated with an encoding stage to transmit unmatched packets, in order to clear non-empty buffers. This technique, however, can be regarded as an asynchronous NC scheme.

The results obtained by [34] are extended in this work by stating the necessary conventional queuing equations for the M/M/1, and then experimenting with different arrival rates for each stream. This is followed by a mathematical construction of a NC queuing model, where the number of incoming packet streams, k, is equal to 2 and 3, and is without input buffers. Coded packets arriving at the output queue are found to follow a general random distribution after matching. It is clear that in a NC node with input buffers, packets stored in the higher arrival rate buffers will be ready for advancement to the matching stage, whenever packets from the other buffer or buffers with lower arrival rates are present; this causes them to follow exactly the same distribution as that of the smallest arrival rate. In this chapter, the system is modelled without input buffers, by allowing the sources to suppress generating packets in the presence of a blockage. The study is further extended by investigating the packet loss probability, through studying the same model with limited buffering capacity.

7.2 System model and assumptions

A single server is considered, with an infinite queue size, a Poisson arrival process with arrival rate λ, and exponentially distributed service times with service rate μ. Figure 7.1 depicts the model, which consists of *k* packet sources connected to a FIFO queue and a server, through a matching/ coding block that performs coding. The interconnections are made through an error-free system with identical link capacities therefore no error control scheme is adopted in this model. For a given time step, at most one packet arrives at, and at most one packet departs from, the FIFO queue. Packets of the i^{th} stream leave the source individually, according to a Poisson process with rate λ_i. Therefore, the *pdf* of the inter-arrival times, denoted as $a(t)$, follows an exponential distribution with average time $1/\lambda$ between arriving packets. Packets are assumed to reach the coding/ merging stage as soon as they are released by the source and therefore the propagation delay is negligible. The FIFO queue is assumed to be placed at the output port of a network switch or router with lossless fabric - the service and arrival processes are independent. The description of the state of the M/M/1 system defines a Markov process because of the *memoryless* property of the exponential distribution [100].

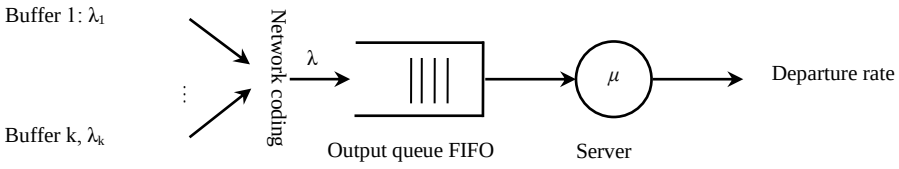


Figure 7.1: Queuing model of an intermediate node with *k* incoming streams without input buffers

In a traditional routing scenario, packets incoming from more than one input stream are merged into one stream, and then stored intact in the FIFO queue, to be served at the next available opportunity. On the other hand, in the studied synchronous NC scenario, packets have to be matched and combined algebraically over *GF(2)* with

the packets from other streams, prior to their entry to the queue, and then stored as coded packets. Coding is assumed to be performed after matching, according to an XOR function once, packets from each stream are ready to be coded. Packets leaving the matching stage are defined as "coded packets"; of particular interest is the distribution which they follow, and the time they spend in the FIFO queue.

Having completed one matching process, the matching stage immediately begins another, provided that at least one input packet of each stream is available; otherwise the matching stage becomes idle and no coded packets enter the FIFO queue in this time domain. All incoming packets from *k* sources are assumed to be sequentially sorted, so that pairs are easily identified for matching. Most studies assume that generated packets are stored in input buffers prior to their entry to the coding stage; in the studied system, however, packets travel directly from the source to the coding stage. In the case of a blockage due to synchronization - waiting for packets from other streams - the source simply stops transmission according to control signals received from the node, and restarts at the next available opportunity. Therefore, a delay is caused by waiting packets from other streams, T_d, to be ready for coding. This delay eventually causes throughput degradation; however, throughput analysis is neither the scope nor the focus of this chapter and will be carried out in further research. The term $\rho=\lambda/\mu$ is defined as the utilization ratio or the traffic intensity. The components of the equilibrium distribution vector of M/M/1 are given by

$$p_k = p_0 \rho^k \tag{7.1}$$

where $p_0 = 1 - \rho$ is the steady–state probability of finding zero packets in the system [69], where $\rho < 1$ is needed to ensure that the system is in equilibrium i.e. $\lambda<\mu$. Since *k* streams of packets are connected to the queue, the average number of

arrivals per unit time should not exceed the average number of packets processed per unit time when the server is busy. Upon arrival to the infinite FIFO queue, packets are served without delay if the server is available; otherwise, packets are forced to wait in the queue until the next possible delivery slot.

7.3 Queuing Analysis of NC-based M/M/1

In this section, the well-established traditional results of M/M/1 are stated in the case of packets being merged at an intermediate node; packets are accepted through any entity input port, and are sent intact through a single entity output port. The mathematical derivations needed to compute the average arrival rate are listed for synchronically coded packets.

7.3.1 The arrival rate of two merged Poisson streams

Traditional routing (TR) operates in a *store-and-forward* mode, where a router accepts and stores all incoming packets, and forwards them without modification to their intended destinations [37]. When data are merged, it is well known that the sum of several independent Poisson processes results in a new Poisson process, whose rate, λ, is the sum of the incoming arrival rates i.e. $\lambda = \lambda_1 + \lambda_2$; this may lead to congestion if the traffic generated by the sources exceeeds the avalible bandwidth at the intermidiate node.

7.3.2 Arrival rate of the coded packets without input buffers

In this section, the mean arrival time of coded packets is derived, and its distribution is then used to derive the mean waiting time which the packets spend in the FIFO queue. As shown in Figure 7.2, the random variable *T* defines the inter-arrival time between two successive coded packets; this R.V. is continous for Poisson traffic and is exponetially distributed [101]. In order to study the random variable *T* in

synchronous NC, the probabilty $p(0)$ that no packets arrive in a period t must be identified.

Theorem 7.1: Consider an encoding node with two independent input streams, as shown in Figure 7.2. Packets coming from streams 1 and 2 are Poisson-distributed with a mean arrival time λ_1 and λ_2, respectively. Furthermore, packets leaving the queue experience an exponential service time with mean $1/\mu$. The resulting coded packets - after matching - follow a random general distribution with a mean arrival rate given by:

$$\lambda = \frac{\lambda_1 \lambda_2 (\lambda_1 + \lambda_2)}{\lambda_1^2 + \lambda_1 \lambda_2 + \lambda_2^2} \tag{7.2}$$

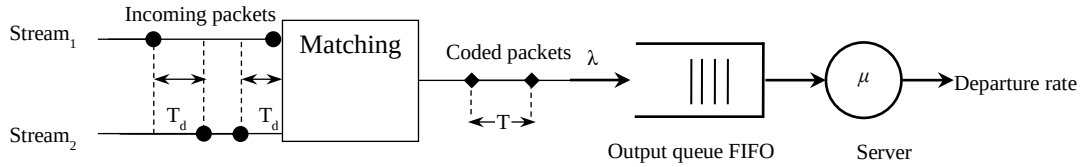


Figure 7.2: The time delay, *T*, incurred by matching

Proof: Let "*A*" be the event where no packets arrive at both streams, or at least one packet arrives at stream 1 but none arrives at stream 2, or at least one packet arrives at stream 2 but none arrives at stream 1, during time period *t*.

Therefore,

$$p(A: T > t) = p_{\lambda_1,t}(0).p_{\lambda_2,t}(0) + \left(1 - p_{\lambda_1,t}(0)\right).p_{\lambda_2,t}(0)$$

$$+(1 - p_{\lambda_2,t}(0)).p_{\lambda_1,t}(0)$$

In order to find the *pdf* associated with the inter-arrival time, a complementary event $B: T \leq t$ has to be identified. This means that one or more packets have arrived at

stream 1 and 2 which satisfy the synchronous coding conditions in time period *t*. The probability associated with event *B* is

$$\begin{aligned} P(B: T \leq t) &= (1 - p_{\lambda_1,t}(0)).(1 - p_{\lambda_2,t}(0)) \\ &= 1 - p_{\lambda_1,t}(0) - p_{\lambda_2,t}(0) + p_{\lambda_1,t}(0).p_{\lambda_2,t}(0) \\ &= 1 - e^{-\lambda_1 t} - e^{-\lambda_2 t} + e^{-(\lambda_1+\lambda_2)t} \end{aligned} \tag{7.3}$$

Hence, the *cdf* for the random variable *T* is given by

$$F_T(t) = 1 - e^{-\lambda_1 t} - e^{-\lambda_2 t} + e^{-(\lambda_1+\lambda_2)t}$$

The *pdf* for this random variable is obtained by differentiating $F_T(t)$

$$f_T(t) = \lambda_1 e^{-\lambda_1 t} + \lambda_2 e^{-\lambda_2 t} - (\lambda_1 + \lambda_2) e^{-(\lambda_1+\lambda_2)t} \tag{7.4}$$

therefore, the average inter-arrival time between coded packets *T* can be obtained by

$$T = \int_{t=0}^{\infty} t f_T(t)\, dt = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2} \tag{7.5}$$

Thus, one can conclude that the inter-arrival time of the coded packets obeys a random general distribution with mean arrival rate λ given by

$$\lambda = \frac{1}{T} = \frac{\lambda_1 \lambda_2 (\lambda_1 + \lambda_2)}{\lambda_1^2 + \lambda_1 \lambda_2 + \lambda_2^2} \tag{7.6}$$

The *pdf* obtained is used to calculate the mean waiting time of the coded packets stated in Section 7.4.

7.3.3 Arrival rate of coded packets for *k*=3 streams

The probability *p (T > t)* for *k* streams can be given by

$p(A: T > t) = 1 - \prod_{i=1}^{K}\left(1 - p_{\lambda_i,t}(0)\right)$, which is equal to $p(A: T > t) = 1 - \prod_{i=1}^{K}\left(1 - e^{-\lambda_i t}\right)$. The complementary event "*B*" can then be given by $P(B: T \leq t) = \prod_{i=1}^{K}\left(1 - e^{-\lambda_i t}\right)$.

When three packet sources are adressing one intermidate node queue, the probability *p (T < t)* then takes the following form

$$P(B: T \leq t) = \prod_{i=1}^{3}\left(1 - e^{-\lambda_i t}\right) \tag{7.7}$$

The *cdf* for the random variable, *T*, is given from (7.7) by

$$\begin{aligned} F_T(t) = & -e^{-\lambda_1 t} - e^{-\lambda_2 t} - e^{-\lambda_3 t} + e^{-(\lambda_1+\lambda_2)t} \\ & + e^{-(\lambda_1+\lambda_3)t} + e^{-(\lambda_2+\lambda_3)t} \\ & + e^{-(\lambda_1+\lambda_2+\lambda_3)t} \end{aligned} \tag{7.8}$$

The *pdf* for this random variable is obtained by differentiating the above equation

$$\begin{aligned} f_T(t) = & \lambda_1 e^{-\lambda_1 t} - \lambda_1 e^{-(\lambda_1+\lambda_2)t} - \lambda_1 e^{-(\lambda_1+\lambda_3)t} \\ & + \lambda_1 e^{-(\lambda_1+\lambda_2+\lambda_3)t} \\ & + \lambda_2 e^{-\lambda_2 t} - \lambda_2 e^{-(\lambda_1+\lambda_2)t} \\ & - \lambda_2 e^{-(\lambda_2+\lambda_3)t} \\ & + \lambda_2 e^{-(\lambda_1+\lambda_2+\lambda_3)t} \\ & + \lambda_3 e^{-\lambda_3 t} - \lambda_3 e^{-(\lambda_1+\lambda_3)t} \\ & - \lambda_3 e^{-(\lambda_2+\lambda_3)t} \\ & + \lambda_3 e^{-(\lambda_1+\lambda_2+\lambda_3)t} \end{aligned} \tag{7.9}$$

Therefore, the mean inter-arrival time of the coded packets of three packet sources is obtained by

$$T = \int_{t=0}^{\infty} t f_T(\mathrm{t})\, \mathrm{d}t$$

$$T = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \frac{1}{\lambda_3} - \frac{1}{\lambda_1 + \lambda_2} - \frac{1}{\lambda_1 + \lambda_3} - \frac{1}{\lambda_2 + \lambda_3} + \frac{1}{\lambda_1 + \lambda_2 + \lambda_3} \tag{7.10}$$

In the special case when $\lambda_1 = \lambda_2 = \lambda_3$, the mean arrival rate for the coded packets is

$$\lambda = \frac{6}{11}\lambda_1 \tag{7.11}$$

This relation has been verified via simulation; a histogram of the distribution of inter-arrival coded packets with three identical inputs is shown in Figure 7.12.

7.4 Mean Waiting Time (MWT) in the FIFO Queue

7.4.1 The mean queue waiting time in traditional routing

Since merged packets follow a Poisson distribution with mean arrival rate $\lambda = \lambda_1 + \lambda_2$, the same formula for obtaining the mean waiting time in M/M/1 queue can be applied. The average number of the packets L in the system is given by $L = \sum_{n=0}^{\infty} nP_n = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu-\lambda}$, where P_n is the probability of finding n packets in the system. The average waiting time in the system can be obtained from Little's formula $W_s = \frac{L}{\lambda} = \frac{1}{\mu-\lambda}$. Thus, the average waiting time in the queue is given by

$$W_q = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\rho}{\mu(1-\rho)} \tag{7.12}$$

7.4.2 NC mean waiting time without input queues

To estimate the time the packets spend in the queue before they are served, the derived results listed above can be employed to analyse the mean waiting time in the

queue under consideration. The system corresponds to the case of an arbitrary inter-arrival time with *pdf a(t)* and *cdf A(t)*. Since the inter-arrival of coded packets is not exponentially distributed, the *embedded Markov chain* method is utilized as an analysis approach [102]. The embedded time points are considered to be the arrival instants of coded packets to the system. Since the service times are exponentially distributed, the number of departures within an inter-arrival time *t* follows a Poisson distribution, which can be found using

$$\alpha_n = \int_{t=0}^{\infty} \frac{(\mu t)^n}{n!} e^{-\mu t}\, \alpha(t) dt \quad n = 0,1,\dots,\infty \tag{7.13}$$

Note that α_n, $n \geq 0$, is the probability that exactly *n* packets have been served during a single inter-arrival period, assuming that more than *n* packets were present at the system upon the commencement of this inter-arrival period.

The number of packets in the system immediately before an arrival instant defines the state of the system. For the i^{th} arrival, let n_i be the number of packet in the system just before this arrival, and let S_{i+1} be the number of packets served between the i^{th} and the $(i+1)^{th}$ arrival. Thus

$$n_{i+1} = \begin{cases} n_i + 1 - S_{i+1} & \text{if } n_i + 1 - S_{i+1} > 0 \\ 0 & \text{if } n_i + 1 - S_{i+1} \leq 0 \end{cases} \tag{7.14}$$

n_i forms an embedded Markov chain for the number in the system at the arrival instants [69].

Under equilibrium conditions, where $i \rightarrow \infty$, let $n_{i+1} = k$ and $n_i = j$, the conditional transition probability p_{jk}, that $(j+1-k)$ packets are served between the consecutive arrival instants, is given by

$$p_{jk} = P\,(\,n_{i+1} = k \mid n_i = j\,)$$

$$= \begin{cases} P\,(\,S_{i+1} = j-k+1\,) & \text{if } k > 0, \\ P\,(\,S_{i+1} \geq j+1\,) & \text{if } k = 0, \end{cases} \tag{7.15}$$

To move from state j to state k, exactly *j+1-k* packets must have been served during the inter-arrival time, hence following (7.13)

$$P_{jk} = a_{j+1-k} \quad k > 0 \tag{7.16}$$

Clearly $p_{jk} = 0$ for all k > j+1, since there are at most j+1 packets present between two consecutive arrivals. Therefore, the transition probabilities matrix P takes the form

$$P = \begin{pmatrix} p_{0,0} & a_0 & 0 & \dots & \\ p_{1,0} & a_1 & a_0 & 0 & \dots \\ p_{2,0} & a_2 & a_1 & a_0 & 0 \\ p_{3,0} & a_3 & a_2 & a_1 & a_0 \\ \vdots & & & & \ddots \end{pmatrix}$$

where α_n denotes the probability of *n* departures in an inter-arrival time interval, assuming that the server is busy for the entire interval. The transition probability diagram is shown in Figure 7.3.

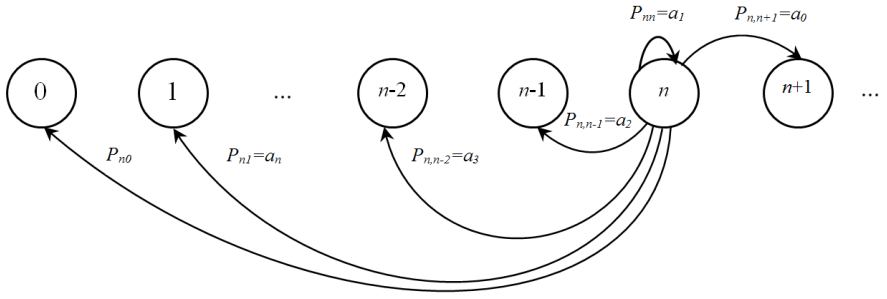


Figure 7.3: Transition probability diagram for embedded Markov chain

To obtain the equilibrium state probabilities p_j, where $j= 0,1,\dots\infty$ of finding *j* packets in the system just before an arbitrary arrival instant, the associated transition probabilities are used to solve a set of balance equations, which can be written as

$$p_j = p_{j-1}\alpha_0 + \sum_{k=0}^{\infty} p_{j+k}\,\alpha_{k+1} \tag{7.17}$$

To find *j* packets in the system upon a packet arrival between two time instants, there must be *j-1* packets in the queue, plus the packet that has arrived, given that no packet has been served between these time instants, or there must be *j* packets in the queue plus the packet that has arrived, given that one packet has been served between these time instants, and so on.

$$p_j = \sum_{k=0}^{\infty} p_{j-1+k}\,\alpha_k \qquad j \geq 0 \tag{7.18}$$

The solution of these equilibrium equations can be found by trying to find a solution in the form of a geometric series

$$p_j = \beta\sigma^j \qquad j \geq 0 \tag{7.19}$$

Substituting this form into equation (7.18), and dividing by $\beta\sigma^{j-1}$, yields

$$\sigma = \sum_{k=0}^{\infty} \sigma^k\,\alpha_k$$

By substituting α_k from (7.13), σ can be expressed as follows

$$\sigma = \sum_{k=0}^{\infty} \sigma^k \int_{t=0}^{\infty} \frac{(\mu t)^k}{k!} e^{-\mu t}\,\alpha(t)dt$$

$$\sigma = \int_{t=0}^{\infty} \sum_{k=0}^{\infty} \frac{(\mu t\sigma)^k}{k!} e^{-\mu t}\,\alpha(t)dt$$

$$= \int_{t=0}^{\infty} e^{-(\mu-\mu\sigma)t}\,\alpha(t)dt$$

Therefore σ can be recognized as the unique root of the Laplace Transform $L_A(\mu-\mu\sigma)$ of the *pdf* of the inter-arrival times i.e.

$$\sigma = L_A(\mu - \mu\sigma) \tag{7.20}$$

To obtain the stability conditions where $\rho>1$ [103] that σ should be a unique real value in the range $0 < \sigma < 1$. Finally, solution (7.19) needs to be normalized, yielding

$$p_j = (1-\sigma)\sigma^j \qquad j \geq 0 \tag{7.21}$$

where $\beta=(1-\sigma)$, since $\sum_{j=0}^{\infty} p_j = 1$. This means that the queue size is geometrically distributed with parameter σ.

A packet will be forced to wait for service with a probability $1 - p_0 = \sigma$. The waiting time in the queue W_q will be zero if packets find the system empty upon arrival. However, if *n* packets are found in the system, including the one currently being served, the mean waiting time can be obtained using the memoryless property of the exponential distribution as follows

$$W_q = E[p_n]\frac{1}{\mu} = \frac{1}{\mu}\sum_{n=1}^{\infty} n(1-\sigma)\sigma^n = \frac{\sigma}{\mu(1-\sigma)} \tag{7.22}$$

The Laplace transform for the inter-arrival time of the coded packets is then found by

$$L\left(a(t)\right) = \frac{\lambda_1}{s+\lambda_1} + \frac{\lambda_2}{s+\lambda_2} - \frac{\lambda_1+\lambda_2}{s+(\lambda_1+\lambda_2)}$$

Therefore, σ can be evaluated, following (7.20) as follows,

$$\sigma = \frac{\lambda_1}{\mu - \mu\sigma + \lambda_1} + \frac{\lambda_2}{\mu - \mu\sigma + \lambda_2} - \frac{\lambda_1+\lambda_2}{\mu - \mu\sigma + (\lambda_1+\lambda_2)} \tag{7.23}$$

This leads to a complicated quartic equation. For simplicity, only the roots when $\lambda_1 = \lambda_2 = \lambda$ are shown, as follows

$$\sigma_1 = 1, \sigma_2 = \frac{\lambda+\mu}{\mu}, \sigma_3 = \frac{3\,\lambda+\mu+\sqrt{\lambda^2+6\,\lambda\mu+\mu^2}}{2\mu}, \sigma_4 = \frac{3\,\lambda+\mu-\sqrt{\lambda^2+6\,\lambda\mu+\mu^2}}{2\mu}$$

The root σ_4 is chosen from the four possibilities since it fulfils the stability condition $0 < \sigma < 1$. For a particular NC queuing system, the ability to quantify the average waiting time in the queue for the coded packets from their average arrival rate is available. Therefore, the performance gap between traditional routing and NC can be exactly determined under certain system parameter settings; this would enable a decision to be made about what arrival rate to implement for each stream, when NC is taken into consideration.

7.4.3 Mean waiting time of coded packets with three input streams

To find the mean waiting time of coded packets with three input streams, the same steps carried out in section 4.3 are followed. The Laplace transform of (7.9) yields

$$\begin{aligned} L\left(a(t)\right) = {} & \frac{\lambda_1}{s+\lambda_1} + \frac{\lambda_2}{s+\lambda_2} + \frac{\lambda_3}{s+\lambda_3} - \frac{\lambda_1}{s+\lambda_1+\lambda_2} - \frac{\lambda_2}{s+\lambda_1+\lambda_2} \\ & - \frac{\lambda_1}{s+\lambda_1+\lambda_3} - \frac{\lambda_3}{s+\lambda_1+\lambda_3} - \frac{\lambda_2}{s+\lambda_2+\lambda_3} - \frac{\lambda_3}{s+\lambda_2+\lambda_3} \\ & + \frac{\lambda_1}{s+\lambda_1+\lambda_2+\lambda_3} + \frac{\lambda_2}{s+\lambda_1+\lambda_2+\lambda_3} + \frac{\lambda_3}{s+\lambda_1+\lambda_2+\lambda_3} \end{aligned}$$

Therefore σ can be obtained by

$$\begin{aligned} \sigma = {} & \frac{\lambda_1}{\mu-\mu\sigma+\lambda_1} + \frac{\lambda_2}{\mu-\mu\sigma+\lambda_2} + \frac{\lambda_3}{\mu-\mu\sigma+\lambda_3} - \frac{\lambda_1}{\mu-\mu\sigma+\lambda_1+\lambda_2} \\ & - \frac{\lambda_2}{\mu-\mu\sigma+\lambda_1+\lambda_2} - \frac{\lambda_1}{\mu-\mu\sigma+\lambda_1+\lambda_3} - \frac{\lambda_3}{\mu-\mu\sigma+\lambda_1+\lambda_3} \\ & - \frac{\lambda_2}{\mu-\mu\sigma+\lambda_2+\lambda_3} - \frac{\lambda_3}{\mu-\mu\sigma+\lambda_2+\lambda_3} + \frac{\lambda_1}{\mu-\mu\sigma+\lambda_1+\lambda_2+\lambda_3} \\ & + \frac{\lambda_2}{\mu-\mu\sigma+\lambda_1+\lambda_2+\lambda_3} + \frac{\lambda_3}{\mu-\mu\sigma+\lambda_1+\lambda_2+\lambda_3} \end{aligned}$$

In the special case where $\lambda_1 = \lambda_2 = \lambda_3 = \lambda$, $\mu=1$;

$$\sigma = \frac{3\lambda}{1-\sigma+\lambda} - \frac{6\lambda}{1-\sigma+2\lambda} + \frac{3\lambda}{1-\sigma+3\lambda} \tag{7.24}$$

Solving this equation gives the following roots

$$\sigma_1 = 1, \sigma_2 = \frac{1}{3}\zeta - \frac{-3\lambda^2-6\lambda-1}{3\zeta} + \frac{2}{3}(3\lambda+1)$$

where $\zeta = \sqrt[3]{-45\lambda^2 + 3\sqrt{3}\sqrt{-\lambda^6 - 6\lambda^5 + 62\lambda^4 + 18\lambda^3 + 2\lambda^2} - 9\lambda - 1}$

$\sigma_{3,4}$ are two conjugate complex numbers and have cumbersome representations that are omitted here. Out of these four roots, σ_2 is selected, since the stability condition $0 < \sigma < 1$ is fulfilled.

7.5 Model Validation and Numerical Results

In this section, the interarrival distribution of the coded packets and the mean waiting time spent in the queue are validated via simulation, and compared with the traditional routing in light of the analysis. The unit of the x-axis of the graphs showing the interarrival distribution of coded and merged packets will be arbitrary time units a time unit. Evidently, the MWT obtained from NC-based queueing will be significantly lower than that obtained from traditional routing; this gain in delay performance is achieved simply by having a smaller number of packets entering the FIFO queue in the NC scenario. Figure 7.4 shows the SimEvents model which comprises two sources of exponentially distributed packets that are transmitted over streams 1 and 2, and are addressing one output port. Therefore, in traditional routing, packets are mergerd using the *Path Combiner* block, which accepts entities through any entity input port, and outputs them through a single entity output port. On the other hand, *Entity Combiner* is utilized to act as a coding stage, where it detects

when all necessary component entities are ready for the combining operation to proceed. Merged and coded packets are then advanced to the FIFO queue, to be served at an exponentially increasing rate of service. The FIFO queue provides several statistical metrics: for instance, *Average wait* which provides a sample mean of the waiting times in the block for all entities that have departed via any port, and *Average queue length* which provides the average number of entities in the queue over time.

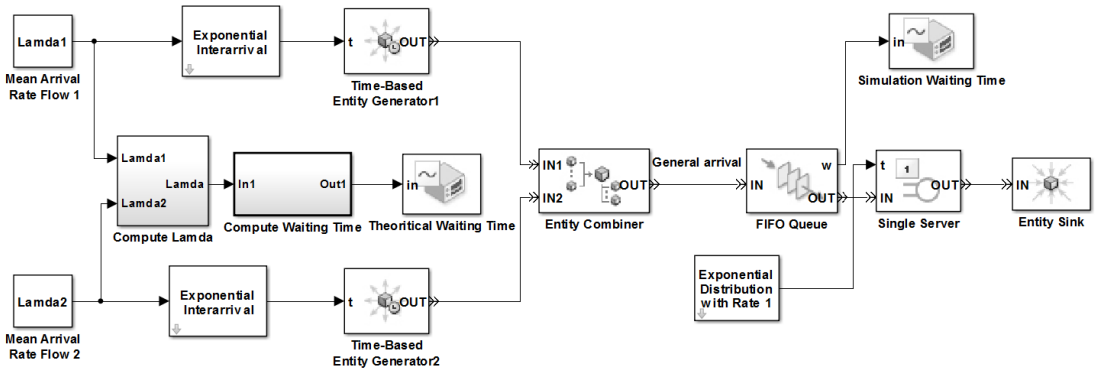


Figure 7.4: NC queuing model in SimEvents

As stated above, packets are set to arrive at the coding/ merging block in a Poisson-distributed way. However, they leave the block following Poisson and random general distributions, in the case of traditional routing and NC, respectively. The total arrival rate for both streams is set to be lower than the service rate required to maintain system stability.

7.5.1 Arrival rate validation

7.5.1.1 Merged packets distribution

By setting, for example, λ_1=0.3, λ_2=0.4 in the traditional routing model, Figure 7.5 shows a histogram of the inter-arrival time of the merged packets which clearly

follow an exponential distribution and in agreement with the theoretical *pdf*, which is given by $f(t) = (\lambda_1 + \lambda_2)e^{-(\lambda_1+\lambda_2)t}$.

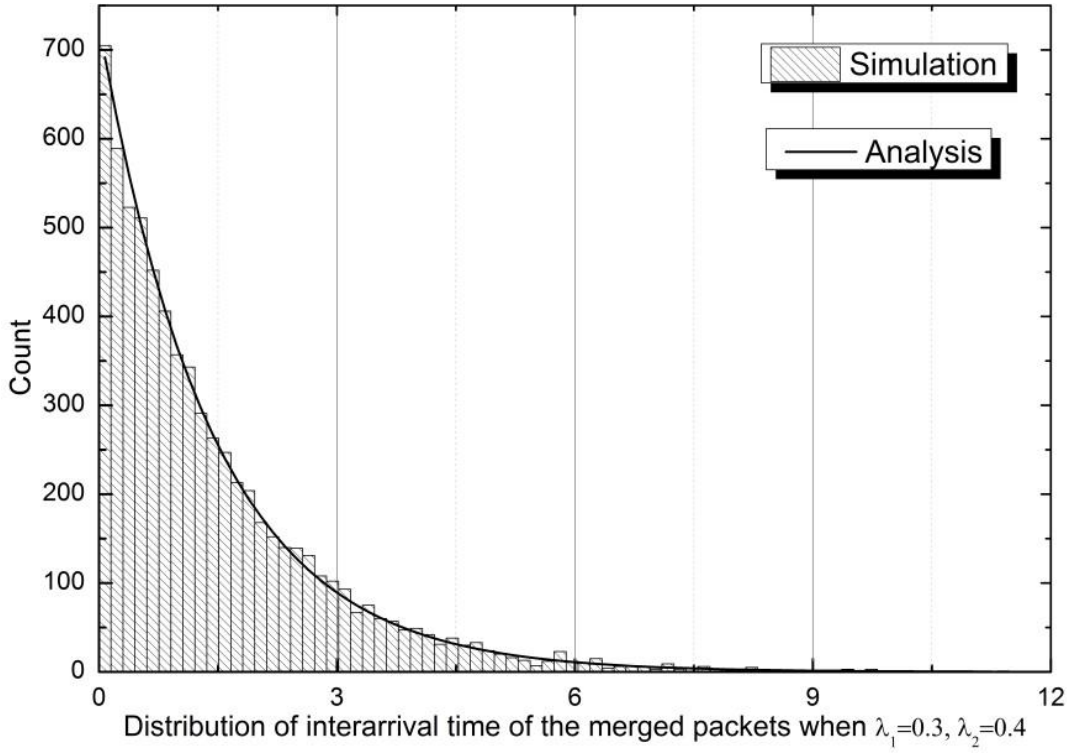


Figure 7.5: Histogram of inter-arrival time of merged packets with λ_1=0.3, λ_2 =0.4

7.5.1.2 Network coded packets with input buffers

In this scenario, input buffers were placed prior to the matching stage, and the simulation run with λ_1=0.3, λ_2=0.4 at buffer 1 and 2, respectively. Packets stored in buffer 2 have to leave in the same way as buffer 1 packets, since $\lambda_1 < \lambda_2$, consequently, coded packets will also follow a λ_1 distribution. Figure 7.6 shows that the inter-arrival time for the coded packets follows an exponential distribution with λ=0.3.

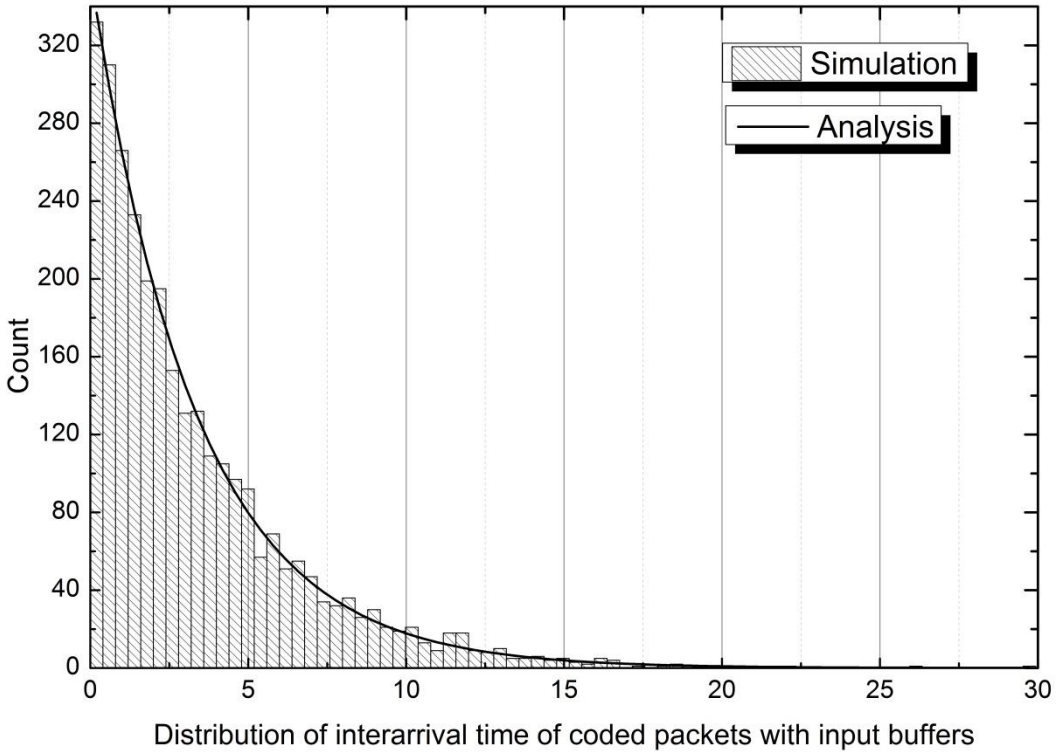


Figure 7.6: Histogram of inter-arrival time of coded packets with λ_1=0.3, λ_2 =0.4 with input buffers

The same model simulation was run with identical arrival rates $\lambda_1 = \lambda_2$; as expected, the resulting coded packets followed a Poisson distribution with the same input parameters $\lambda = \lambda_1 = \lambda_2$, as proved by [34].

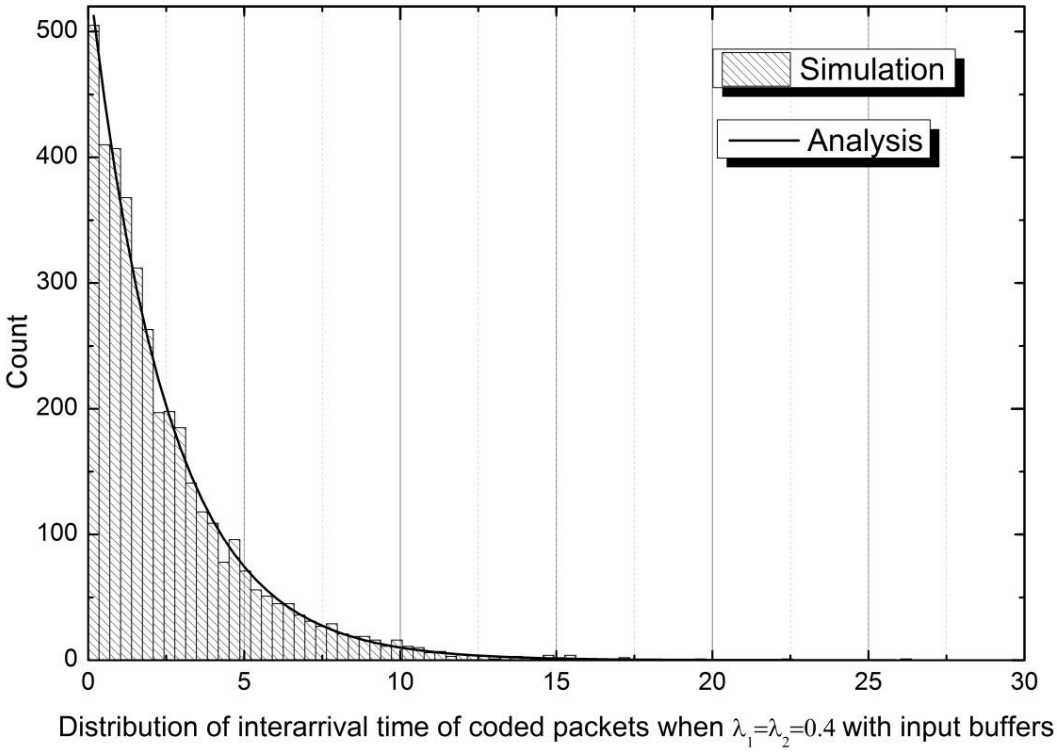


Figure 7.7: Histogram of inter-arrival time of coded packets with $\lambda_1=\lambda_2$ and input buffers

Evidently, with an unequal mean arrival time, buffer 2 connected to stream 2 will build-up quickly and become uncontrollable, since it receives more packets than buffer 1 per time unit, as illustrated in Figure 7.8. Therefore, buffer 2 will always have its packets ready for coding, and will be confined to the behaviour of buffer 1.

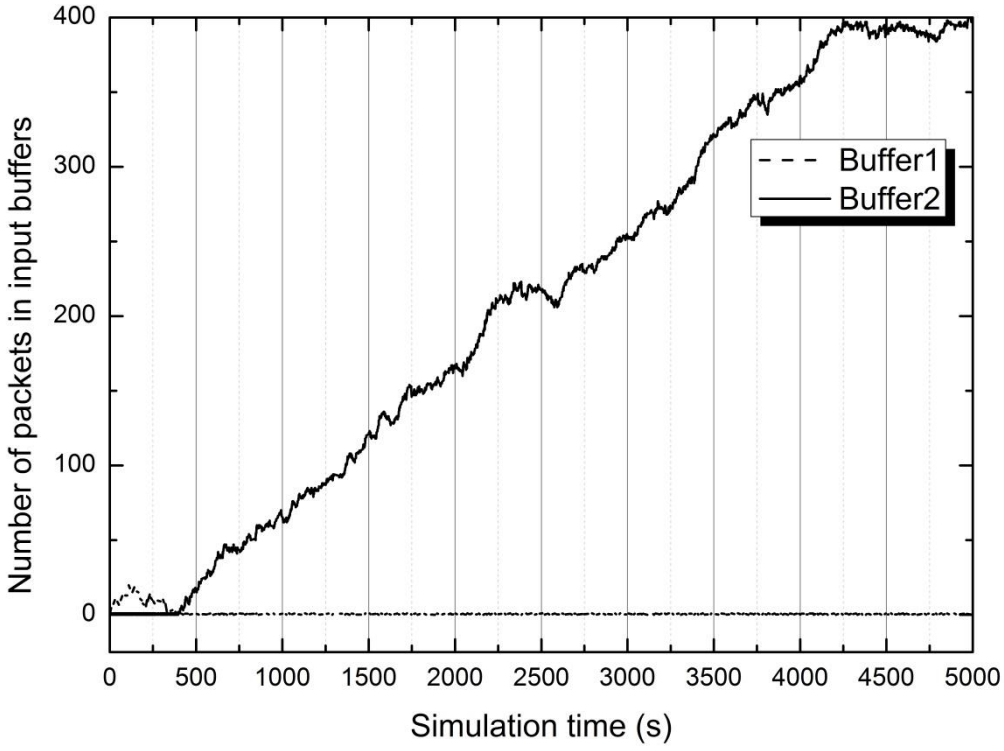


Figure 7.8: Length of each input buffer with different arrival rates

7.5.1.3 Distribution of network coded packets without input buffers

In this section, Theorem 7.1 is validated via simulation, and the results of coded packet distribution when λ_1=0.3, λ_2 =0.4 are shown. Figure 7.9 indicates that the inter-arrival distribution of the incoming packets, under an NC scenario for a given M/M/1 queue, follows a general distribution with a mean arrival rate given by (7.6).

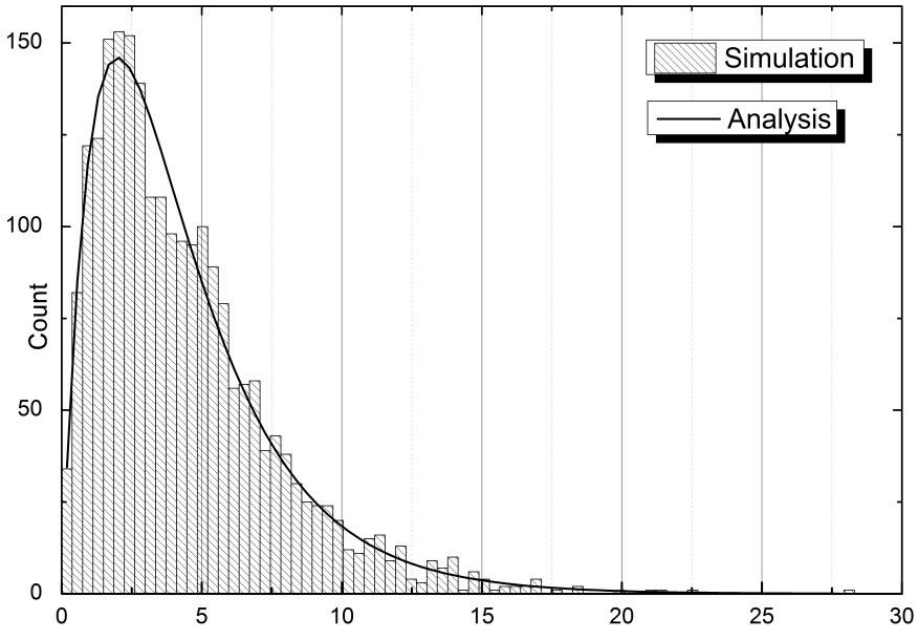


Figure 7.9: Histogram of inter-arrival time of the coded packets with λ_1=0.3, λ_2 =0.4.

Figure 7.10 shows that the simulated *pdf* of the coded packets is in agreement with the theoretical *pdf*, which given by (7.4)

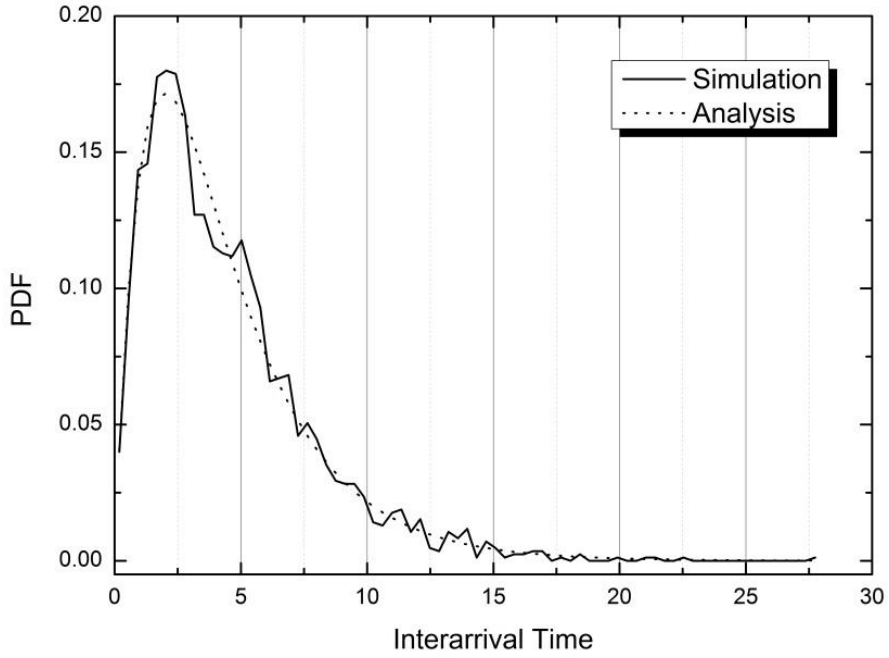


Figure 7.10: The pdf of inter-arrival time of the coded packets

In the special case when λ_1=λ_2, the coded packets still follow a general distribution, with the arrival rate governed by equation (7.6), i.e. λ= (2/3) λ_1 as shown in Figure 7.11.

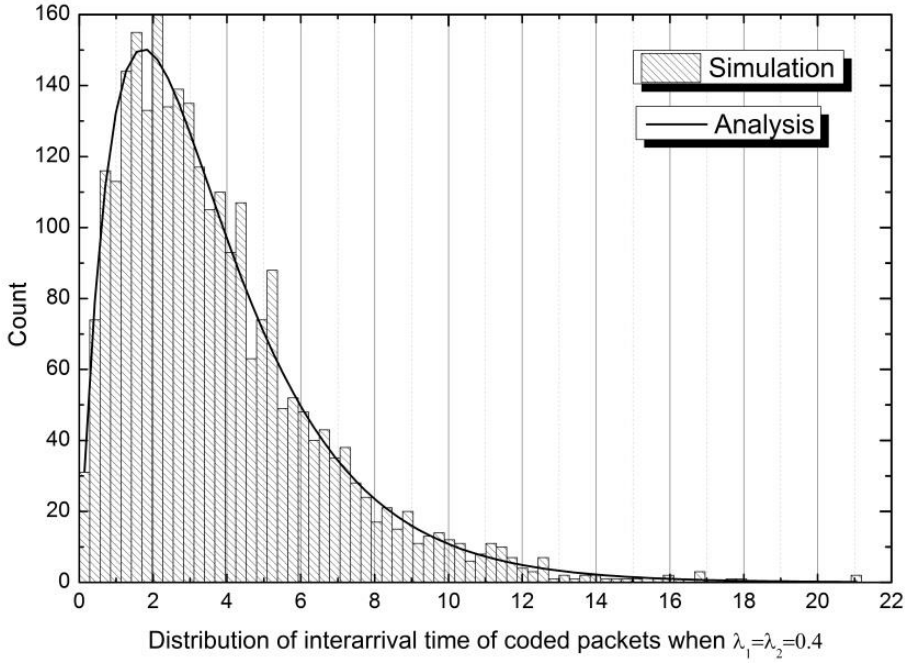


Figure 7.11: Histogram of inter-arrival time of coded packets with $\lambda_1=\lambda_2$

7.5.1.4 Distribution of network coded packets with three identical arrival rates

The model constructed was modified to include a third packet source, and used to validate the mean arrival rate relation given by (7.11). The packet arrival rate for all streams was set, for example, to 0.4 packets per second, i.e. λ_1, λ_2, λ_3=0.4. Figure 7.12 shows that coded packets with the three identical input streams model follow a general distribution with λ given by (7.11).

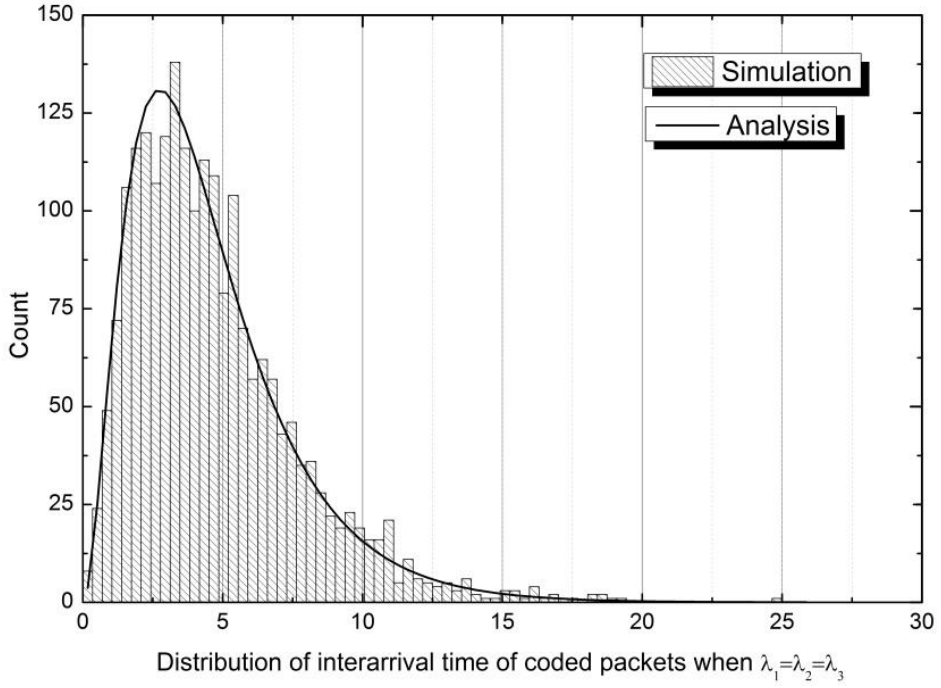


Figure 7.12: Histogram of inter-arrival time of coded packets with λ_1, λ_2, λ_3=0.4

7.5.1.5 Distribution of network coded packets with three different arrival rates

Figure 7.13 shows that coded packets with three different input streams, for example, λ_1=0.2, λ_2=0.3 and λ_3=0.4, follow a general distribution with λ given by the corresponding formula of (7.10).

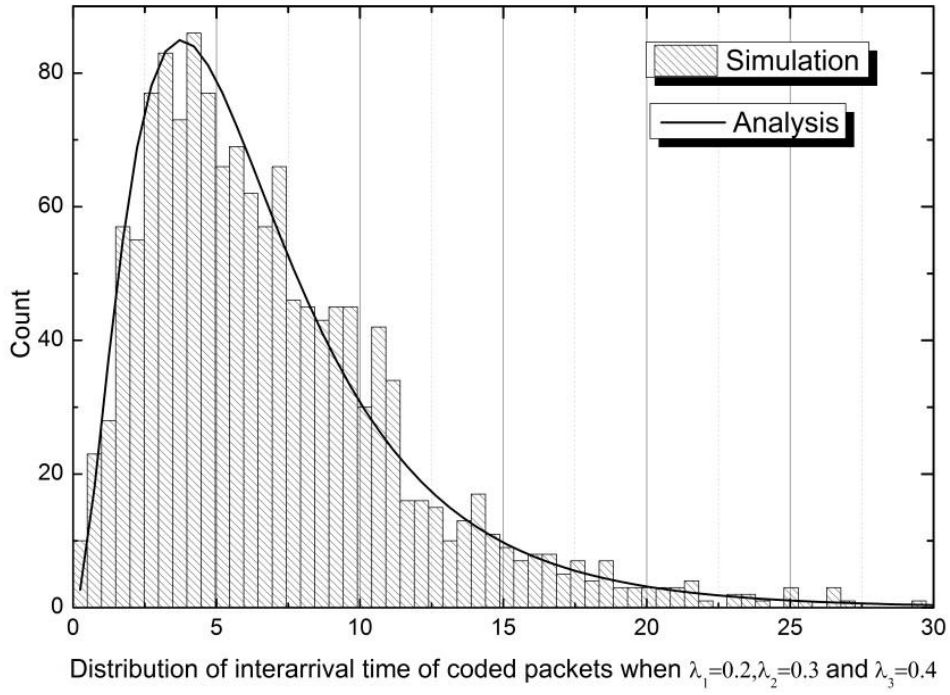


Figure 7.13: Histogram of inter-arrival time of coded packets with λ_1=0.2, λ_2=0.3 and λ_3=0.4

7.5.2 Queue mean waiting time validation

In this section, attention is given to the waiting time that the merged and coded packets spend in the intermediate node queue before they are served. A comparison between the analytical and simulation results for the MWT for merged and coded packets is conducted.

7.5.2.1 MWT for Merged packets

When setting λ_1=0.3 and λ_2=0.4, the MWT is obtained by applying the well-known equation (7.12), which is expressed as $W_q = \frac{1}{1-(0.3+0.4)} - \frac{1}{1} = 2.33$ seconds; this matches the simulation results very well, as shown in Figure 7.14.

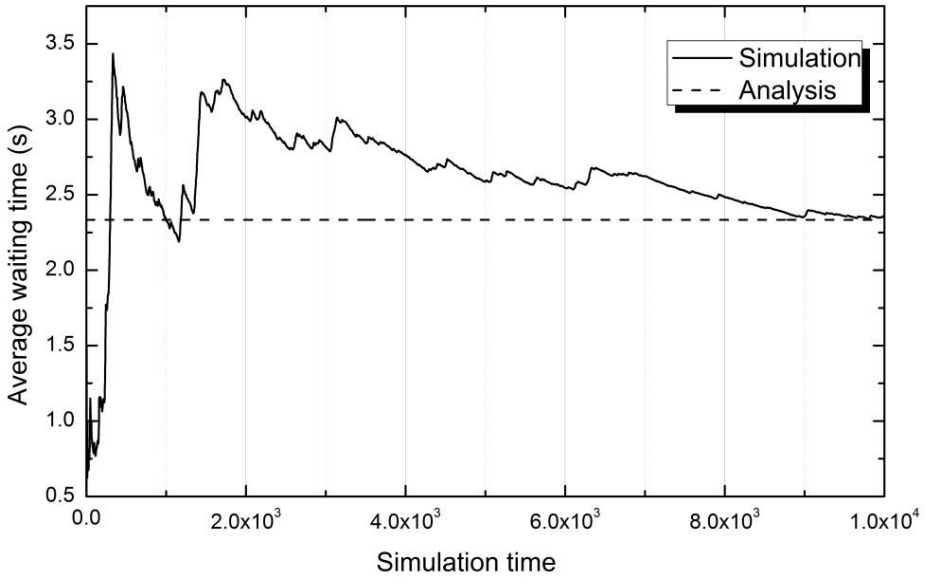


Figure 7.14: Average mean waiting time of merged packets in FIFO queue with λ_1=0.3, λ_2 =0.4

7.5.2.2 MWT for network coded packets without input buffers

Figure 7.15 shows that the analytical results obtained by applying the derived equation (7.22) when λ_1=0.3, λ_2=0.4 and μ=1, match the simulation results extremely well.

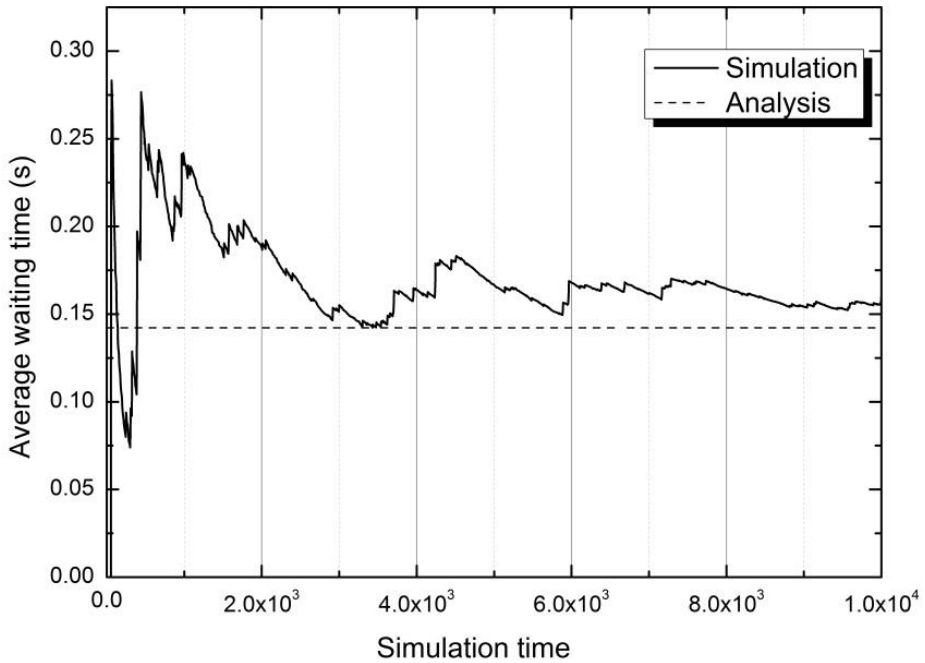


Figure 7.15: Average mean waiting time of coded packets in FIFO queue with λ_1=0.3, λ_2 =0.4 and without input buffers

7.5.2.3 MWT for network coded packets without input buffers when $\lambda_1 = \lambda_2$

Applying formula (7.22) for $\lambda_1 = \lambda_2 = 0.4$ and $\mu=1$, values of $\sigma_4 = 0.1566$ and $W_q = 0.1857$ seconds are obtained. The obtained average waiting time shown in Figure 7.16 follows the analytical results very closely.

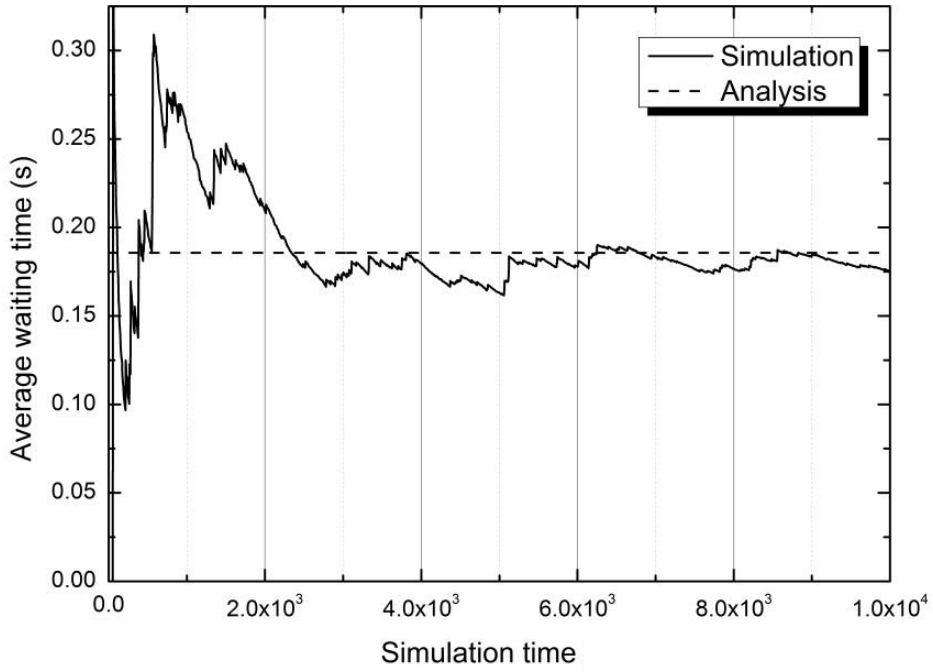


Figure 7.16: Average mean waiting time of coded packets in FIFO queue with $\lambda_1=\lambda_2 =0.4$ and without input buffers.

7.5.2.4 Network coded packets without input buffers with 3 input streams

Applying formula (7.24) when $\lambda_1=\lambda_2=\lambda_3=0.4$, $\mu=1$, $\sigma_1 = 1$, $\sigma_2 = 0.08$ and $\sigma_{3,4} = 2.16 \mp i0.39$ can be obtained. The parameter σ_2 meets the stability conditions, since it falls in the range $0 < \sigma < 1$, therefore a MWT $W_q = 0.09$ seconds is obtained from (7.22). Figure 7.17 verifies the obtained results for this section.

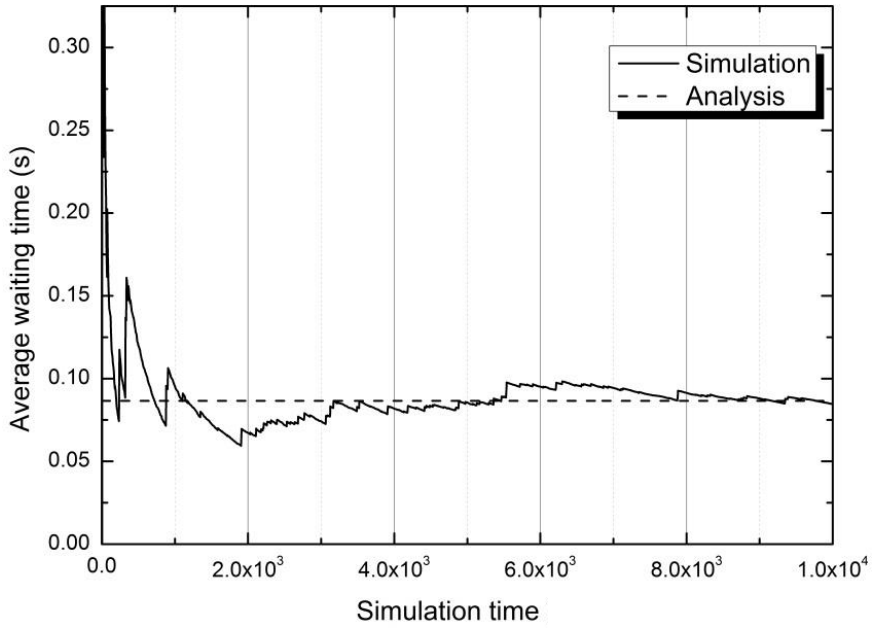


Figure 7.17: Average mean waiting time of coded packets in FIFO queue with $\lambda_1 = \lambda_2 = \lambda_3 = 0.4$.

7.5.2.5 Mean waiting time of network coded packets with input buffers

To evaluate the MWT performance of M/M/1, the model was modified by adding infinite input buffers at each input of the coding stage. Setting λ_1=0.3 and λ_2 =0.4, the resulting λ will follow an exponential distribution and be equal to λ_1, since it has a smaller value. Therefore applying equation (7.12) for λ=0.3, yields $W_q =$ 0.43 seconds, which was verified by simulation as shown in Figure 7.18.

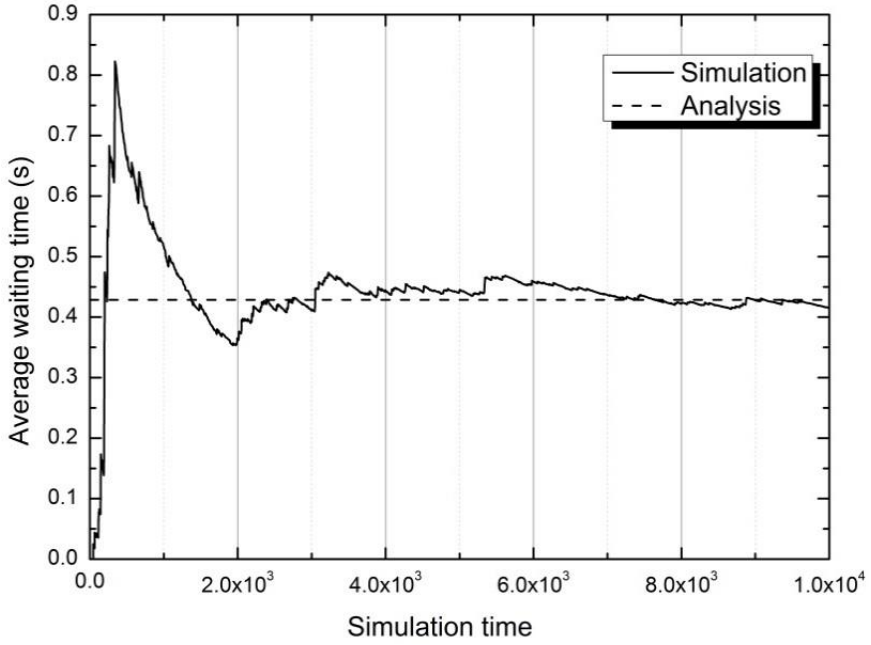


Figure 7.18: Average mean waiting time of coded packets in FIFO queue with λ_1=0.3, λ_2 =0.4 with input buffers

It is clearly noticed that the MWT of coded packets with input buffers placed at the coding node is much less than that without input buffers. This is because having input buffers at the input of coding nodes increases the coding opportunities and hence increases the number of coded packets entering and waiting in the FIFO queue.

7.5.2.6 Examining different arrival rates

In this section, more results regarding the MWT for both traditional routing and NC are provided. The arrival rate of the packets coming from stream 1 is fixed at a value 0.4, and λ_2 is set to range between [0.1, 0.55]. The NC technique brings a new solution for queue stability, as shown in Figure 7.19.

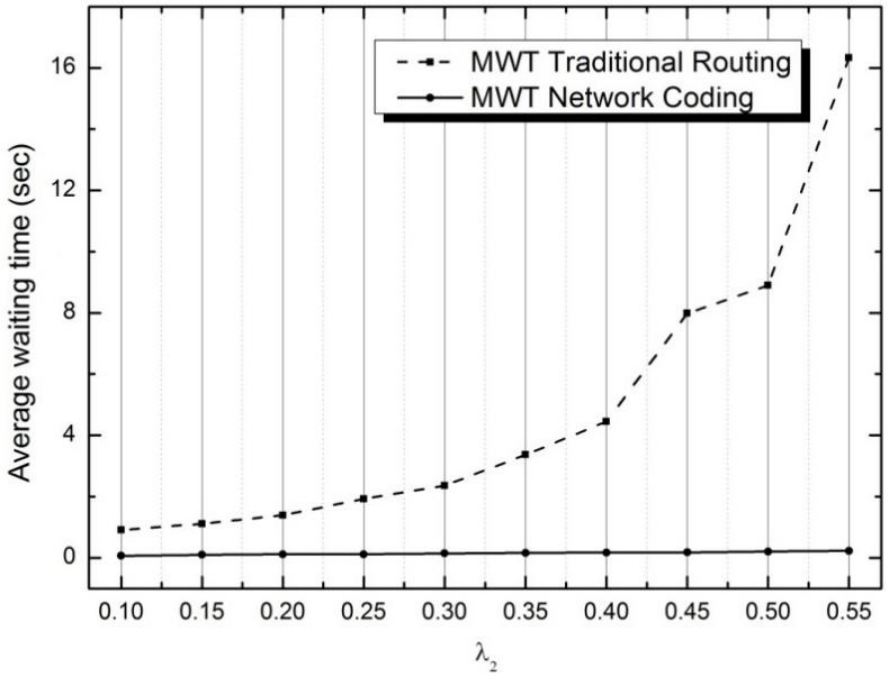


Figure 7.19: An average queue waiting times for NC-based node for a fixed λ_1 and varying λ_2

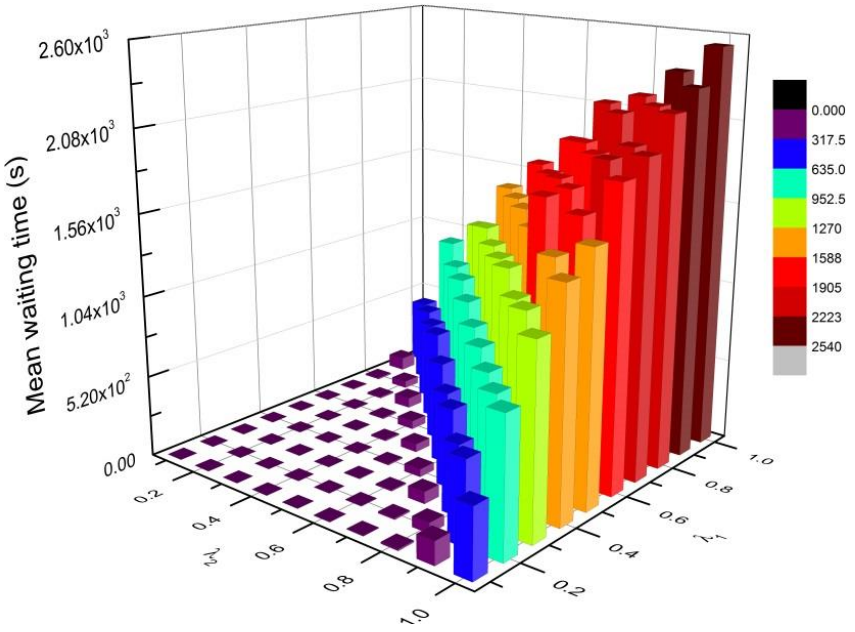


Figure 7.20: Average queue waiting times for traditional node for various mean waiting time values

To further enhance the visualisation of what happens in the intermediate node queue when merging two streams, both λ_1 and λ_2 were set to range between [0.1, 1], for both TR and NC techniques. Figure 7.20 indicates that a rapid change in the packet's MWT occurs with the traditional scenario when $\lambda_1 + \lambda_2$ becomes greater than or equal

to the service time, which leads to congestion problems. This means that with capacity-limited queues, packets at the intermediate node will be rejected and hence, the data transfer will be dominated by the retransmissions of the discarded packets, which leads to throughput wastage.

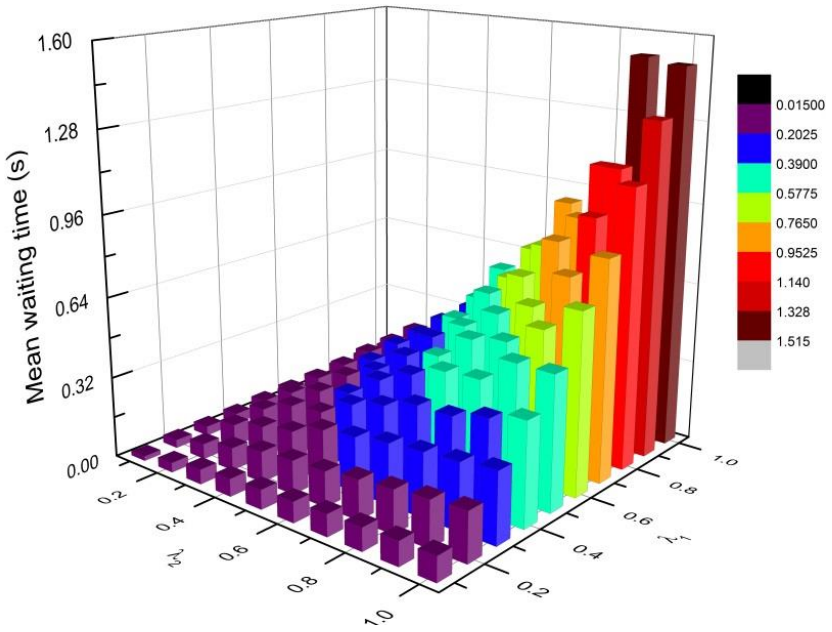


Figure 7.21: Average queue waiting times for NC-based node for various mean waiting time values

Figure 7.21 shows that NC acts like a control mechanism by reducing the incoming data rate, and hence helps in reducing the waiting time that the packets spend in the queue; consequently, the network does not become clogged with retransmissions.

7.6 Stability Condition

To ensure queuing stability, the total arrival rate of the incoming traffic should not exceed the service rate. For two input streams, the stability condition for a queue in a traditional routing scenario with two incoming streams is then given by

$$\lambda_1+\lambda_2 < \mu \tag{7.25}$$

From the relation (7.6), the stability condition for a queue when NC is adopted is given by

$$\lambda = \frac{\lambda_1\lambda_2(\lambda_1 + \lambda_2)}{\lambda_1^2 + \lambda_1\lambda_2 + \lambda_2^2} < \mu$$

$$\lambda_1\lambda_2(\lambda_1 + \lambda_2) < \mu(\lambda_1^2 + \lambda_1\lambda_2 + \lambda_2^2)$$

$$(\lambda_1 + \lambda_2) < \mu(\frac{\lambda_1}{\lambda_2} + 1 + \frac{\lambda_2}{\lambda_1}) \tag{7.26}$$

By comparing (7.25) and (7.26), NC allows the packet source to function at a rate higher than that of TR, whilst maintaining the property of generating packets in a controlled manner as a response to blockage. In the special case where $\lambda_1 = \lambda_2 = \lambda$, the stability condition is given by $\lambda < \frac{\mu}{2}$ and $\lambda < \frac{3\mu}{2}$ for TR and NC, respectively. This allows a wider stability domain for the FIFO queue, as previously shown in Figures 7.19, 7.20 and 7.21.

7.7 Server Utilization

To further validate the analytical model, the server utilization *U* for both scenarios is investigated, which is measured by the proportion of time when the server is busy serving an entity - this implies the time where the queue is not empty. Therefore,

$$U = 1 - p_0 = \rho \tag{7.27}$$

7.7.1 Traditional routing server utilization

Since $\mu=1$ and $\rho=\lambda/\mu$, the utilization *U* is equal to λ. In traditional routing, the total arrival rate λ can be found as $= \sum_{i=1}^{k} \lambda_i$ where *k* is the number of the incoming node links. Therefore, the utilization *U* is equal to 0.7, which is validated via the simulation results presented in Figure 7. 22.

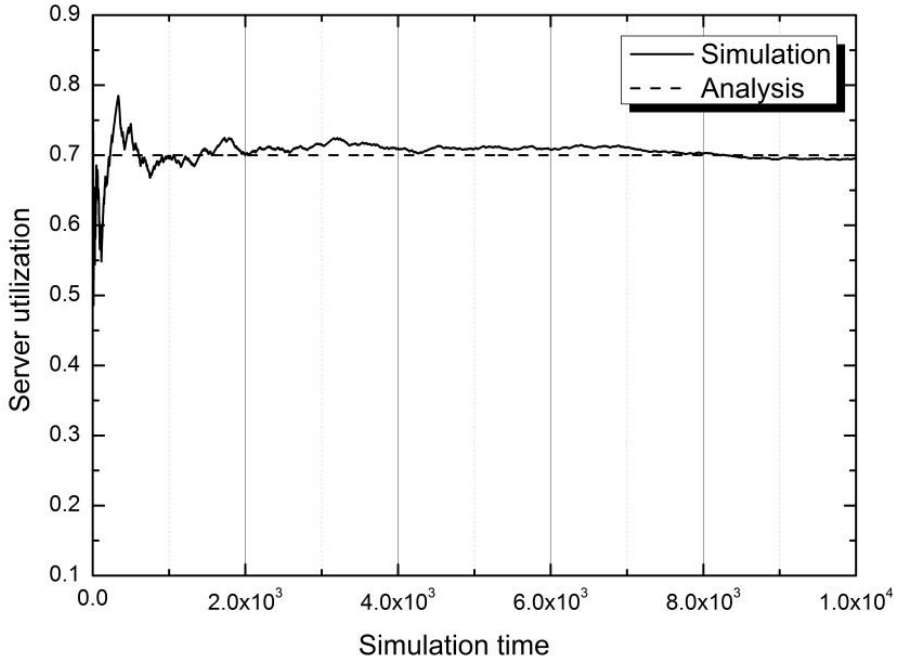


Figure 7.22: Server utilization of traditional routing when λ_1=0.4, λ_2=0.3

7.7.2 NC server utilization

Similar to traditional routing, the value for utilization is given by (7.27) for the total arrival rate λ of the coded packets. Applying (7.6) for λ_1=0.4, λ_2=0.5, $U = 0.295$ is obtained and shown in Figure 7.23. Comparing the results shown in Figures (7.22) and (7.23), one can conclude that the server will be required to work less in the case of NC, since it will receive fewer packets due to the effect of packet combining, which is desirable from the perspective of queue stability.

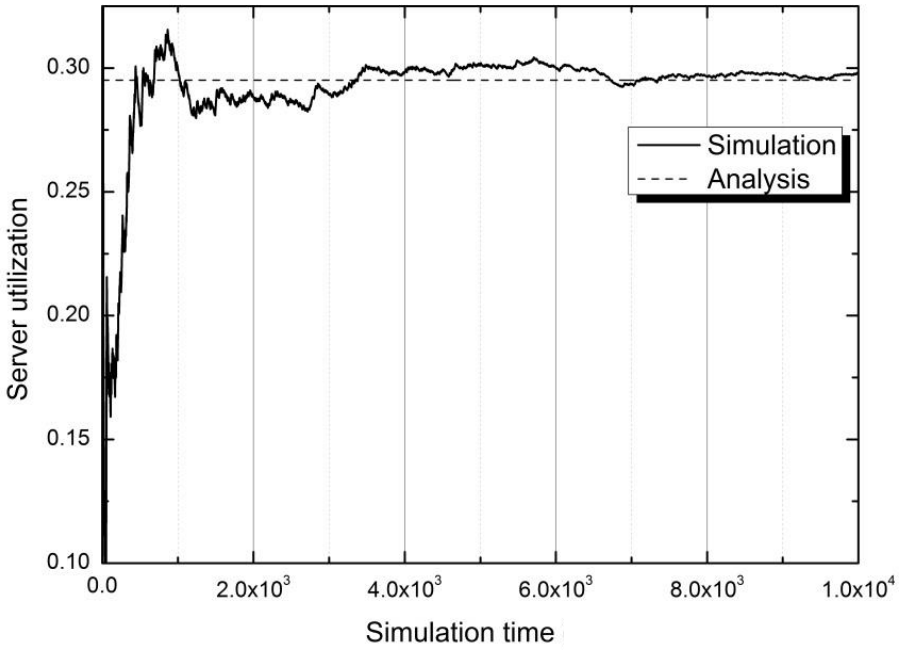


Figure 7.23: Server utilization of NC when λ_1=0.4, λ_2=0.5

7.8 NC-based M/M/1 with Finite Capacity *K*

The work presented in the previous sections is now extended to consider the case where the queue size is no longer infinite. In this case, the system can only accommodate *K* packets at a time. Newly arriving packets which find the system full will not be admitted, and they are considered as "lost" packets. Therefore, the effective arrival rate becomes $\lambda(1-p_k)$, where p_k is the probability that the system is full. In the case of traditional routing, p_k is given by [69]

$$p_k = p_0 \rho^k \tag{7.28}$$

where ρ is the distribution index and is given by $\rho = \lambda/\mu$ and $p_0 = \frac{(1-\rho)}{1-\rho^{k+1}}$. The packet loss probability *L* is the ratio of lost traffic, N (lost), to the overall traffic which comprises the lost and input traffic N(in), and hence is the expressed as $L = \frac{N(lost)}{N(lost)+N(in)}$.

$$L = \frac{\lambda p_k}{\lambda p_k + \lambda(1 - p_k)} = p_k \tag{7.29}$$

Following the previous analysis, the relation (7.15) takes the following form

$$n_{i+1} = \begin{cases} \min\,(n_i \ + \ 1 - S_{i+1}, K) & \text{if } n_i \ + \ 1 - S_{i+1} > 0 \\ 0 & \text{if } n_i \ + \ 1 - S_{i+1} \leq 0 \end{cases} \tag{7.30}$$

Using probabilities α_n, $n=0,1,2,\ldots$, defined in (7.13), the transition probability matrix P can be displayed as

$$P = \begin{array}{c} \\ 0 \\ 1 \\ \vdots \\ K-1 \\ K \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & \ldots & K-1 & K \end{array} \\ \begin{pmatrix} p_{0,0} & a_0 & 0 & & & \\ p_{1,0} & a_1 & a_0 & & & \\ \vdots & \vdots & & & & \\ p_{K,0} & a_{K-1} & a_{K-2} & \ldots & a_1 & a_0 \\ p_{K,0} & a_{K-1} & a_{K-2} & \ldots & a_1 & a_0 \end{pmatrix} \end{array} \tag{7.31}$$

If a packet C_n arrives to find a full system, then it will be lost, whereas if it arrives to find K-1 packets in the system, then it queues up and makes the system full. Due to the memoryless property of the service time distribution, these two cases are equivalent in terms of what the next packet, C_{n+1}, sees, thus the last two rows of the matrix P are identical. The limiting distribution of the embedded chain $p=(p_0, p_1, \ldots, p_k)$ can be determined through the following equations, which are obtained from the *stationarity* property of the distribution [104].

$$p_j = \sum_{i=0}^{K} p_i P_{ij} \quad j = 0,1,2,\ldots, \tag{7.32}$$

Given that $P_{ij} = a_{i-j+1}\ j > 0$, $P_{i0} = \sum_{n=i+1}^{\infty} a_n$, p can be evaluated through

$$p_0 = \sum_{i=0}^{K} p_i \left(\sum_{n=i+1}^{\infty} a_n \right)$$

$$p_1 = p_0 a_0 + p_1 a_1 + \cdots + p_{k-1} a_{k-1} + p_k a_{k-1}$$

$$p_2 = p_1 a_0 + p_2 a_1 + \cdots + p_{k-1} a_{k-2} + p_k a_{k-2}$$

$$\vdots$$

$$p_{k-1} = p_{k-2} a_0 + p_{k-1} a_1 + p_k a_1$$

$$p_k = p_{k-1} a_0 + p_k a_0$$

Solving these simultaneous equations, along with the normalization condition $\sum_{j=0}^{K} p_j = 1$, can be considered computationally practical if *K* is not too large, otherwise, a computational recursion can be developed as an alternative procedure [104]. To verify the analysis, the system is set, for example, to fit three packets, i.e. K=3, hence p_j can be written in a matrix form as follows

$$\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & a_0 & a_{0-1} \\ 0 & a_0 & a_1 - 1 & a_1 \\ a_0 & a_1 - 1 & a_2 & a_3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{7.33}$$

The target is then to find *L* for both NC and traditional routing scenarios. The simulation was run with a queue size *K* selected to fit three packets. Packets from stream 1 arrive according to a fixed data rate λ_1=0.4, and packet arrival rate of stream 2, λ_2, are set to range from 0.1 to 0.9. The theoretical results of solving equation (7.33) and simulation analysis are shown in Figure 7.24 which concludes that, with the increase of λ_2, the packet loss probability of un-coded packets increases steadily and linearly. However, the packet loss probability when NC is adopted only increases slightly, for two reasons: Firstly, due to synchronous coding, a delay is imposed on packets prior to their entry to the coding stage to match other streams' packets. This gives the server more time to serve the awaiting packets and

empty the queue for new arrivals. Secondly, fewer packets will compete for space in the queue, since the queue receives a combination of several packets, due to coding.

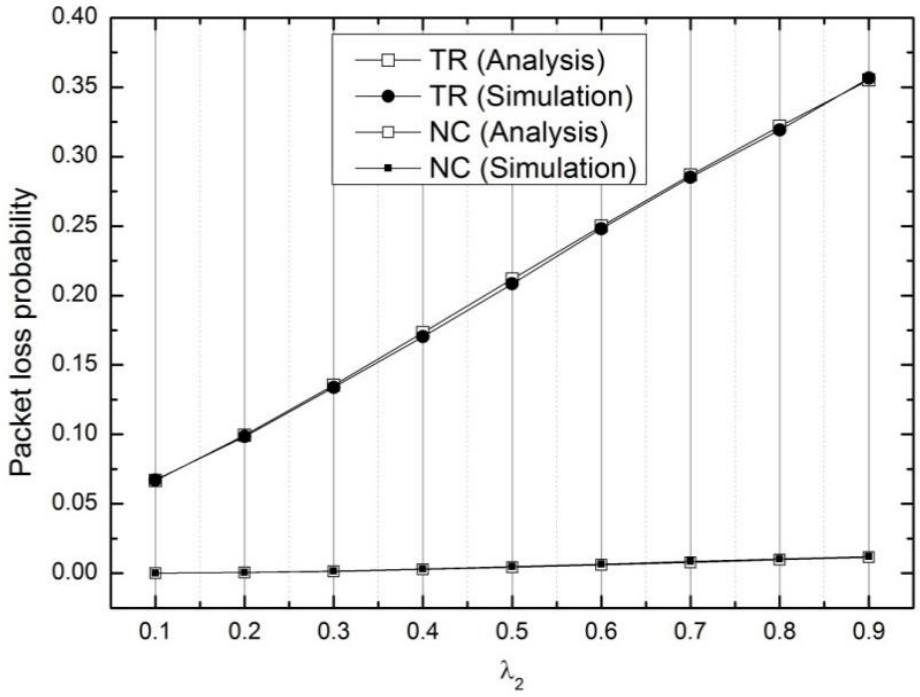


Figure 7.24: M/M/1/K packet loss probability in both traditional and NC routing scenarios

7.9 Conclusions

For an intermediate multicast network node, this chapter has compared a NC-based M/M/1 queuing model with traditional routing under different packet arrival rates. The distribution behaviour of coded packets is constructed. Having different packet arrival rates in the presented model removes the drawbacks of some existing works on queuing analysis. The analysis also proves that adopting NC at an intermediate network node reduces the number of packets entering the queue from different streams, and hence yields satisfactory network performance; this is evident when the MWT for both NC and TR scenarios is compared. The analysis is further verified by studying the server utilization, and to be close to a real-world application, the analysis was extended to encompass capacity-limited queues. It is thereby shown

that NC is able to maintain a very low packet loss probability compared with TR. In the following chapter, a simulation-based analysis is carried out to examine the behaviour for coded packets in a queuing model with deterministic service.

Chapter 8

Performance of a Network Coding Queuing Model with Deterministic Service

8.1 Introduction

Chapter 7 has described the analysis of a NC-based intermediate node queue with exponential packet inter-arrival and service times, M/M/1. The theoretical and simulation results showed that incoming packets from different streams follow a general random distribution after coding. Additionally, NC was shown to achieve significant queuing waiting time stability.

As an extension of Chapter 7, this chapter studies the behaviour of an M/D/1 queuing system under a synchronous NC scenario without input buffers; an intermediate node represented by a First-In-First-Out (FIFO) queue and a single server, is configured so that it receives merged and coded packets from two streams. The same derivation of the inter-arrival relationship of the coded packets presented in Chapter 7 is followed. The coded packet distribution expression is stated and verified via simulation. Performance measures of interest include the comparison of the FIFO queue mean waiting time (MWT) and server utilization for merged and coded packets. Results show that the mean waiting time of the coded packets is significantly lower than in traditional routing (TR), and servers are lightly utilised when synchronous NC is employed since coded packets may comprise two or more un-coded packets.

8.2 System Model and Assumptions

An intermediate node with an infinite queue size, a single server, a Poisson arrival process with arrival rate λ_k, and deterministic service times with service rate μ, is considered as shown in Figure 8.1.

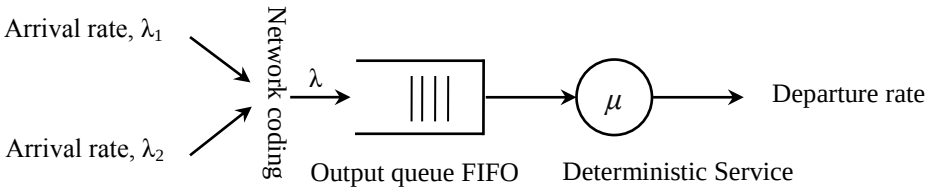


Figure 8.1: A NC-based queuing system with two incoming links and without input buffers

The model consists of two packet sources connected to an output FIFO queue and a server. The interconnections are made through an error-free system with identical link capacities therefore no error control scheme is adopted in this model. Packets from the i^{th} stream leave the source individually, according to a Poisson process with rate λ_i; consequently the inter-arrival time follows an exponential distribution with an average time of $1/\lambda_i$ between arriving packets. Packets reach the coding/ merging stage as soon as they are released by the source; therefore the propagation delay is assumed to be negligible. The FIFO queue is assumed to be placed at an output port of a network switch or router with a lossless fabric - the service and arrival processes are independent. The description of the state of the M/D/1 system defines a Semi-Markovian process because of the *memoryless* property of the exponential distribution [105].

In a traditional routing scenario, incoming packets from more than one input stream are merged into one stream, and stored intact in the queue to be served at the next available opportunity. However, in a synchronous NC scenario, packets must be matched and combined with the packets from other streams prior to their entry to the queue, and then stored as coded packets. Packets are combined algebraically over

GF(2), and then stored in the queue after matching is established, and are referred to as “coded packets”; the distribution of these packets is interesting, as is the time spent in the FIFO queue. All incoming packets from sources 1 and 2 are assumed to be sequentially sorted so that pairs are easily identified for matching.

After the completion of one matching process, another starts straight away if one or more input packet from each stream exists; if this condition is not met the matching stage waits with no coded packets going into the FIFO queue during the time period.

Most studies assume that generated packets are stored in input buffers prior to their entry to the coding stage [7, 8, 34]. In the system studied here however, packets travel directly from the source to the coding stage. In the case of source blockage due to synchronization, caused by waiting for packets from other streams, the source pauses packet generation until the subsequent port becomes available. The term $\rho=\lambda/\mu$ is defined as the utilization ratio or the traffic intensity; $\rho<1$ is needed to ensure that the system is in equilibrium i.e. $\lambda<\mu$. Since the queue has two incoming streams of packets, the total average number of arrivals per unit time should not exceed the average number of packets processed per unit time when the server is busy. Upon coding, packets arriving at the infinite FIFO queue are served without delay if the server is available, otherwise packets are forced to wait in the queue until the next possible availability.

8.3 Queuing Analysis of NC-Based M/D/1

In this section, the well-established traditional results for the case of packets being merged at the FIFO queue are stated; in this scenario, packets are accepted through any entity input port, and output - intact - through a single entity output port, due to merging.

8.3.1 Arrival rate of merged Poisson-distributed streams

Traditional routing (TR) functions by a *store and forward* method; packets enter a router where they are stored, and then sent without change to the destination [37, 103]. The sum of several independent Poisson processes for merged data gives a new process, where the rate is the sum of the incoming arrival rates. This results in the arrival rate λ of two incoming streams becoming $\lambda = \lambda_1 + \lambda_2$.

8.3.2 Arrival rate of coded packets without input buffers

The inter-arrival time, *T*, between two successive coded packets can be constructed using the same analysis followed in Chapter 7 section 7.3.2, which is given by

$$T = \int_{t=0}^{\infty} t f_T(\mathrm{t})\, \mathrm{d}t \; = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2} \tag{8.1}$$

and the mean arrival rate λ is then can be given by

$$\lambda = \frac{1}{T} = \frac{\lambda_1 \lambda_2 (\lambda_1 + \lambda_2)}{\lambda_1^2 + \lambda_1 \lambda_2 + \lambda_2^2} \tag{8.2}$$

8.4 Model Validation and Numerical Results

In this section, the distribution of the coded packets is validated via simulation, and compared with that of traditional routing, considering the analysis in section 8.3. A simulation of the MWT in the FIFO queue for both merged and coded packets is then presented. Since merged packets follow a Poisson distribution with a mean arrival rate $\lambda = \lambda_1 + \lambda_2$, the characteristics of this queue are originated from the Pollaczek-Khintchin formula [69]; since the service time of the system is constant, the variance is zero, i.e. $\sigma^2 = 0$. The statistical measures are thereby developed and listed below.

$$L_q = \rho^2 / [2(1 - \rho)] \tag{8.3}$$

$$W_q = \frac{L_q}{\lambda} = \frac{1}{2(\mu - \lambda)} - \frac{1}{2\mu} \tag{8.4}$$

where the ρ is the utilization ratio, and is equal to λ/μ. L_q and W_q are the queue length and the mean waiting time in the queue, respectively.

To be able to validate the analysis of the model studied, a discrete-time single-server queue, with infinite capacity and a deterministic service time is considered. In SimEvents, sources are represented by a "*Time-Based Entity Generator*" block, and the merging and coding stages are represented by "*Path Combiner*" and "*Entity Combiner*" blocks, respectively. The "*Path Combiner*" block accepts entities through any entity input port, and outputs them through a single entity output port. The "*Entity Combiner"* block acts as a coding stage, detecting when all necessary component entities are ready for the combining operation to proceed. Figure 8.3 shows the SimEvents model, which comprises two sources of exponentially-distributed packets that are transmitted over streams 1 and 2, and addressing one output port. The FIFO queue provides several statistical metrics; for instance, the "*Average Wait"*, which provides a sample mean of the waiting times in the block for all entities that have departed via any port, and the "*Average Queue Length"*, which provides the average number of entities in the queue over time. As stated above, packets generated in a Poisson-distributed manner leave the coding/ merging block following a Poisson distribution in the case of traditional routing, and a random general distribution in the case of NC.

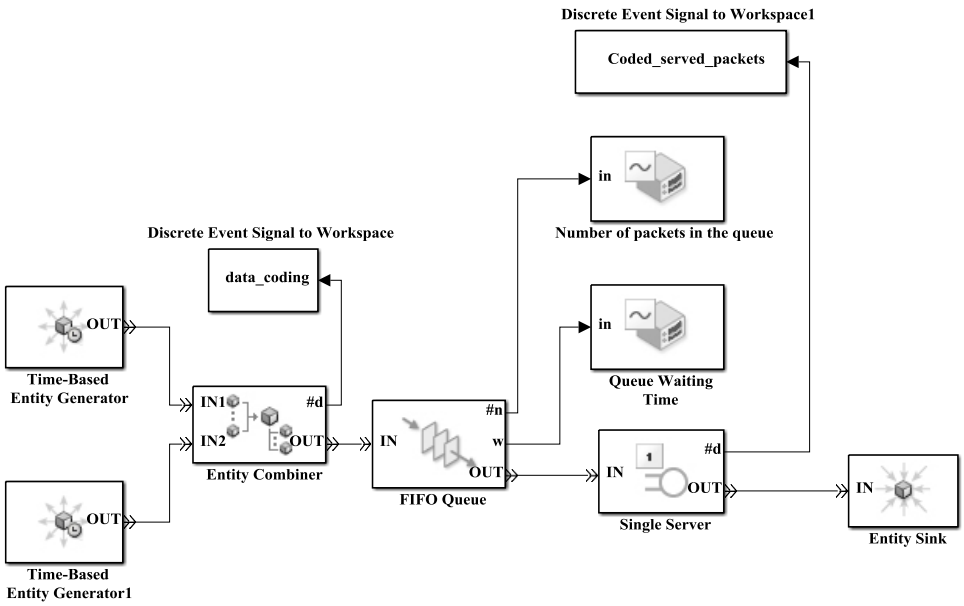


Figure 8.2: NC queuing model in SimEvents

8.4.1 Arrival rate validation

8.4.1.1 Merged packets distribution

The performance of the traditional routing model stated above is evaluated via simulation using a model with settings λ_1=0.4, λ_2=0.5 and μ=1. Figure 8.3 shows a histogram of the inter-arrival time of the merged packets which clearly follows an exponential distribution with mean $\lambda = \lambda_1 + \lambda_2$. Figure 8.4 shows that the simulated *pdf* of the merged packets is in agreement with the theorical pdf, which is given by $f(t) = (\lambda_1 + \lambda_2)\, e^{-(\lambda 1 + \lambda 2)\, t}$.

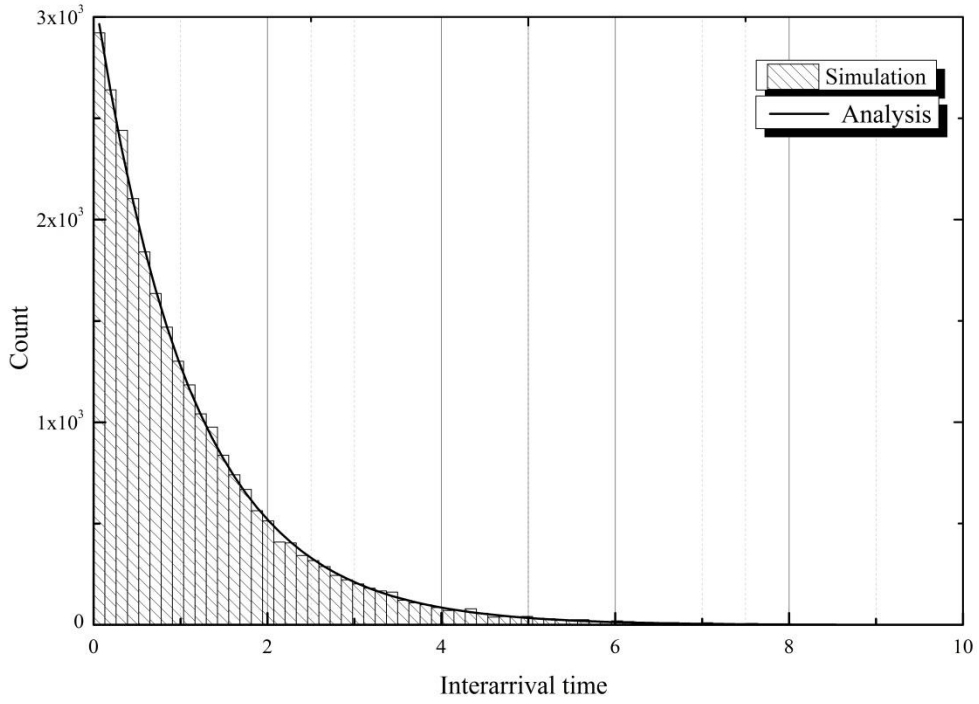


Figure 8.3: Histogram of inter-arrival time of merged packets with λ_1=0.4, λ_2 =0.5

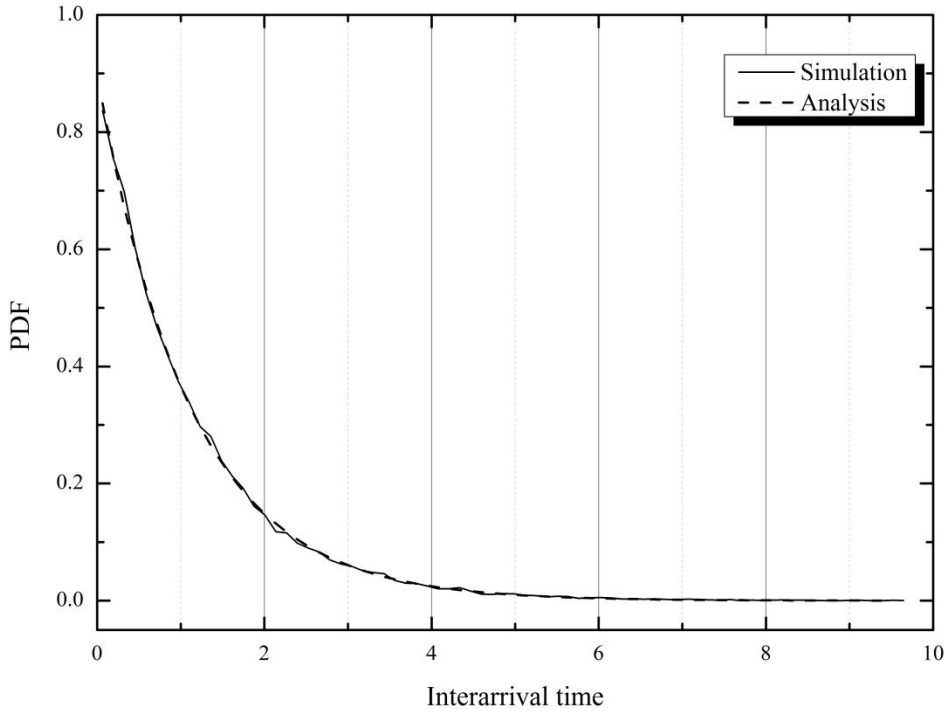


Figure 8.4: A comparison between the analytical and simulated pdf of the merged packet inter-arrival time

8.4.1.2 Network coded packets distribution without input buffers

The system considered in this section is shown in Figure 8.2. Streams 1 and 2 are set to generate packets at λ_1=0.3 and λ_2 =0.4, respectively. Figures 8.5 and 8.6 indicate

that the inter-arrival distribution of the incoming packets, under a NC scenario for a given M/D/1 queue, follows a general random distribution with a mean inter-arrival time given by (8.2). Figure 8.6 shows that the derived and simulated pdf of the coded packets are in excellent agreement.

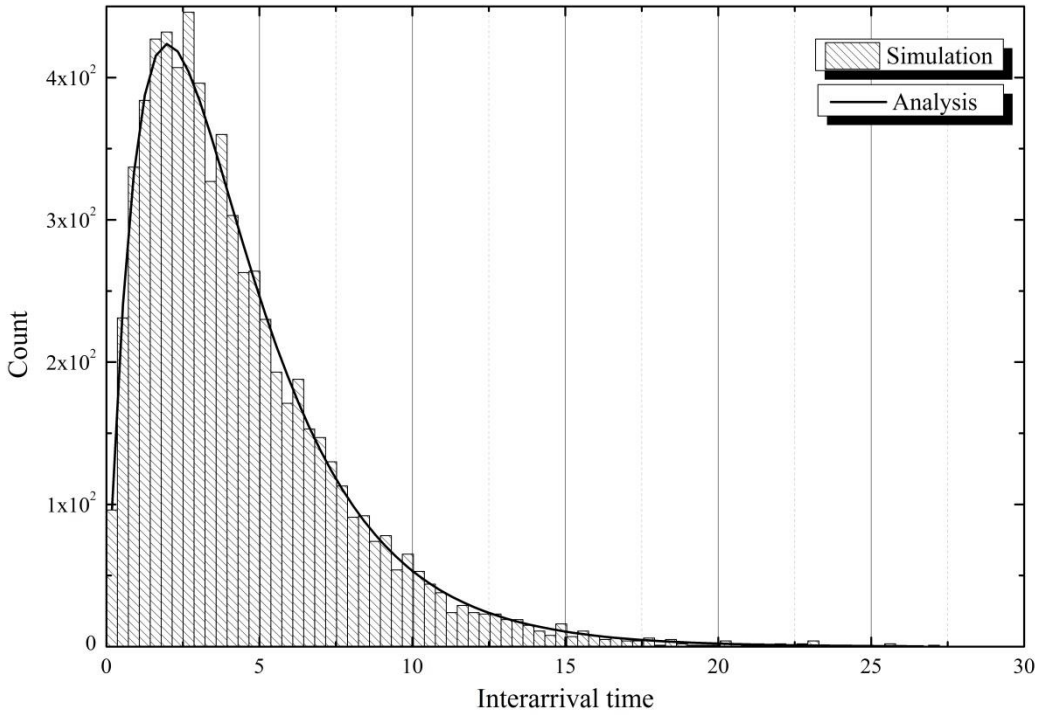


Figure 8.5: Histogram of inter-arrival time of coded packets with λ_1=0.3, λ_2 =0.4.

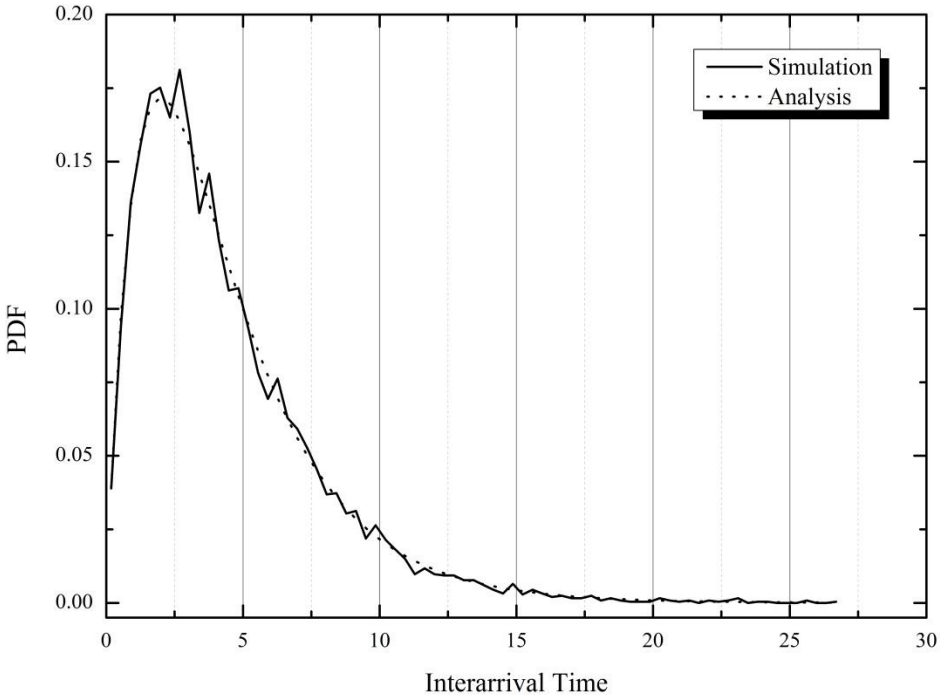


Figure 8.6: The pdf of the coded packet inter-arrival time.

8.4.1.3 Distribution behaviour of coded packets with various input buffer sizes

This section now provides a comparison of the coded packet distributions of four scenarios with various input buffer sizes, K. Packets are first set to travel from the sources to the coding node without input buffers, then the buffer is set to accommodate 1, 2 and 3 packets. Figure 8.13 shows how the coded packet distribution starts to take the exponential form as K increases.

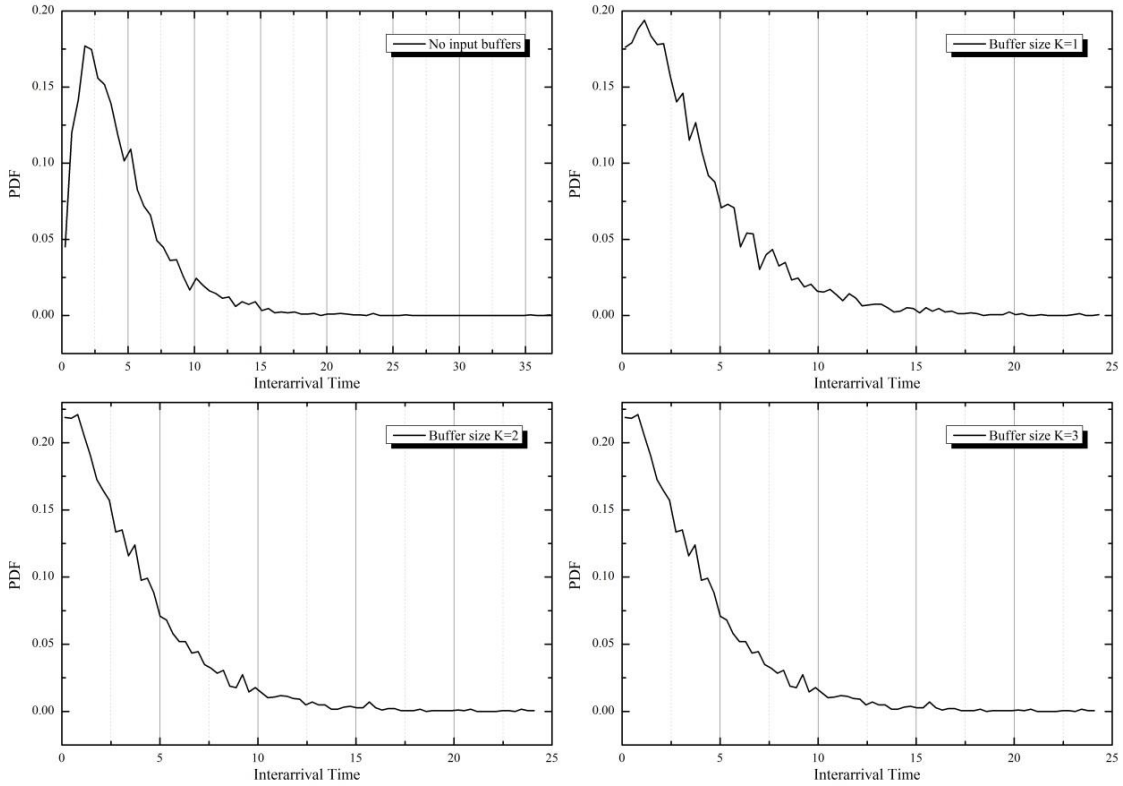


Figure 8.7: A pdf comparison of four coding scenarios with various input buffer sizes

8.4.2 Queue mean waiting time

In this section, a simulation-based comparison of the MWT for the merged and coded packets is shown, for selected values of λ; a more thorough simulation for a wider range of values of λ is then given in the next section.

A. MWT for merged packets

With values of λ_1=0.4, λ_2=0.5 and μ=1, the MWT value of merged packets in the FIFO queue is obtained by applying the well-known equation in (8.4), which is expressed as $W_q = \frac{1}{2(1-0.9)} - \frac{1}{2} = 4.5$ seconds. This value matches the simulation results presented in Figure 8.8 very well.

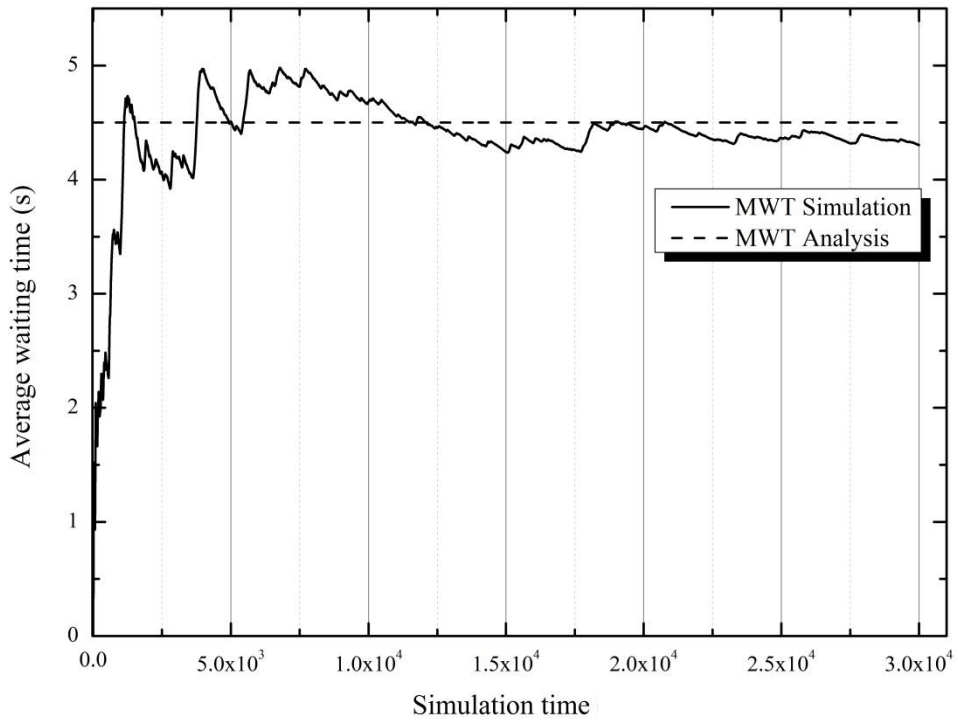


Figure 8.8: Average mean waiting time of merged packets in FIFO queue with λ_1=0.4, λ_2 =0.5

B. MWT for network coded packets without input buffers

Figure 8.9 shows the MWT of the coded packets obtained via simulation when, for example, λ_1=0.3, λ_2=0.4 and μ=1. In the following section, a range of different values of λ is used to show a full overview of the MWT for both traditional and NC nodes. It is evident that coded packets have to wait a much shorter time than the merged packets, due to the smaller number of combined packets that the server must deliver. It can be noted, by comparing Figures 7.15 and 8.9, that the MWT of coded packets with deterministic service is smaller compared to that of exponential service.

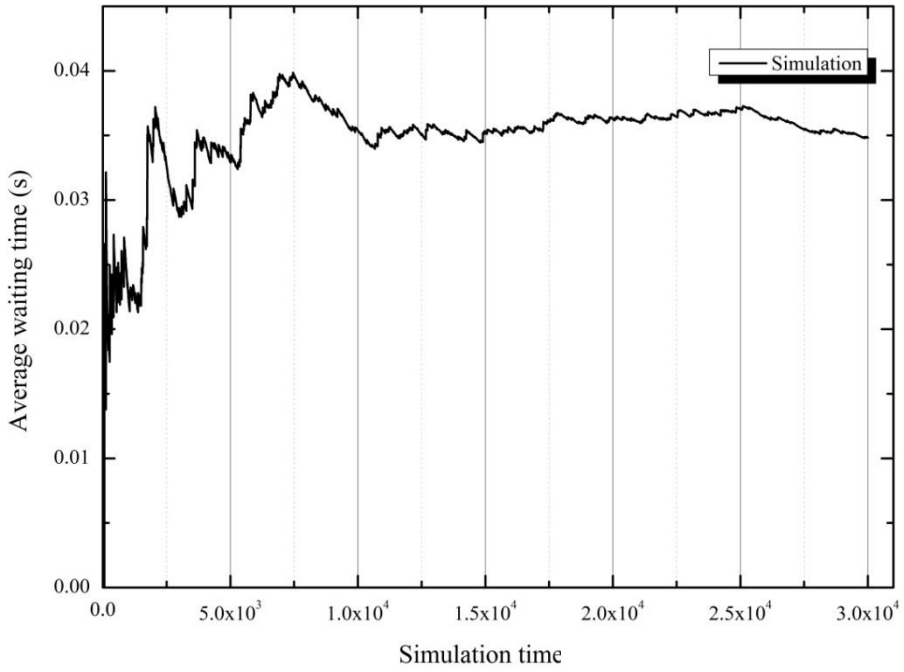


Figure 8.9: Mean waiting time of coded packets in the FIFO queue with λ_1=0.3, λ_2 =0.4, without input buffers

C. Examining MWT for different arrival rates

Here, further extensive simulation results are provided regarding the MWT for both traditional routing and NC. The arrival rates λ_1 and λ_2 are set to a range between [0.1, 1], for both TR and NC techniques. Evidently, the MWT obtained from NC-based queuing is significantly lower than that obtained from traditional routing; this gain in delay performance is achieved simply by having a lower number of packets entering the FIFO queue in the NC scenario.

The NC technique offers a new solution for queuing stability, as shown in Figures 8.10 and 8.11. There are several observations to be made: Firstly, one can observe that a sudden change in a packet's MWT under the traditional scenario occurs when $\lambda_1 + \lambda_2$ become equal to or greater than the service time and hence a traditional node system stops functioning, as demonstrated by Figure 8.10. However, the FIFO queue under NC increases gradually, remaining stable for a larger set of values for λ, as long as the stability condition discusses in Section 7.6 is met. Secondly, the MWT

experienced by the coded packets is considerably shorter than the value for the merged packets; this is due to the fact that fewer packets satisfy the entry condition of the coding stage, hence the server will have to serve fewer packets.

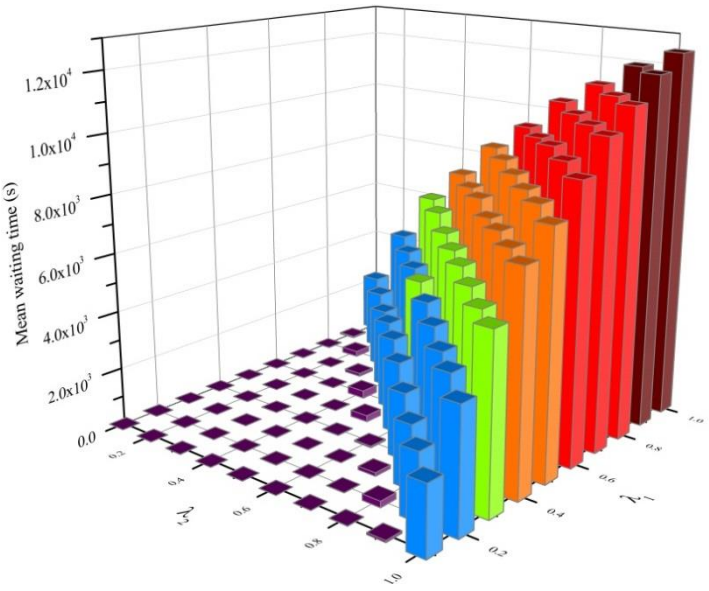


Figure 8.10: Average queue waiting times for traditional node for various arrival rates

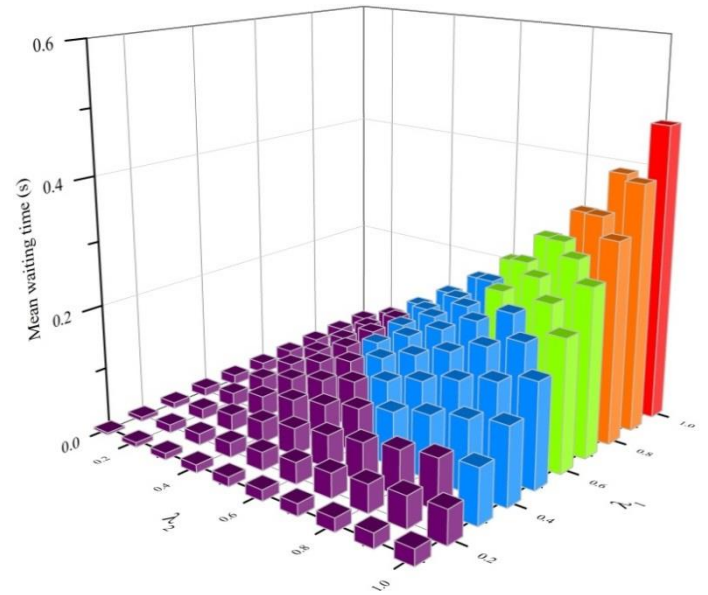


Figure 8.11: Average queue waiting times for NC-based node for various arrival rates

It is worthy of noting that the maximum value of MWT of the coded packets with deterministic service did not exceed the value of 0.5 seconds whereas it was about 1.6 seconds when $\lambda_1 = \lambda_2=1$ as shown in Figures 8.11 and 7.21, respectively.

8.4.3 Server utilization

To further validate the analytical model, the server utilization *U* for both scenarios is simulated; this is measured by the proportion of time that the server is busy serving an entity, therefore

$$U = 1 - p_0 = \rho \tag{8.5}$$

where p_0 is the probability the queue is empty.

A. Traditional routing server utilization

Since μ=1 and $\rho=\lambda/\mu$, the utilization *U* is equal to λ. In TR, the total arrival rate λ can be defined as $\lambda = \sum_{i=1}^{k} \lambda_i$, where *k* is the number of the incoming node links. Therefore, when *k=2,* λ_1=0.3 and λ_2=0.4, the value of *U* is 0.7, which is confirmed by the simulation results presented in Figure 8.12.

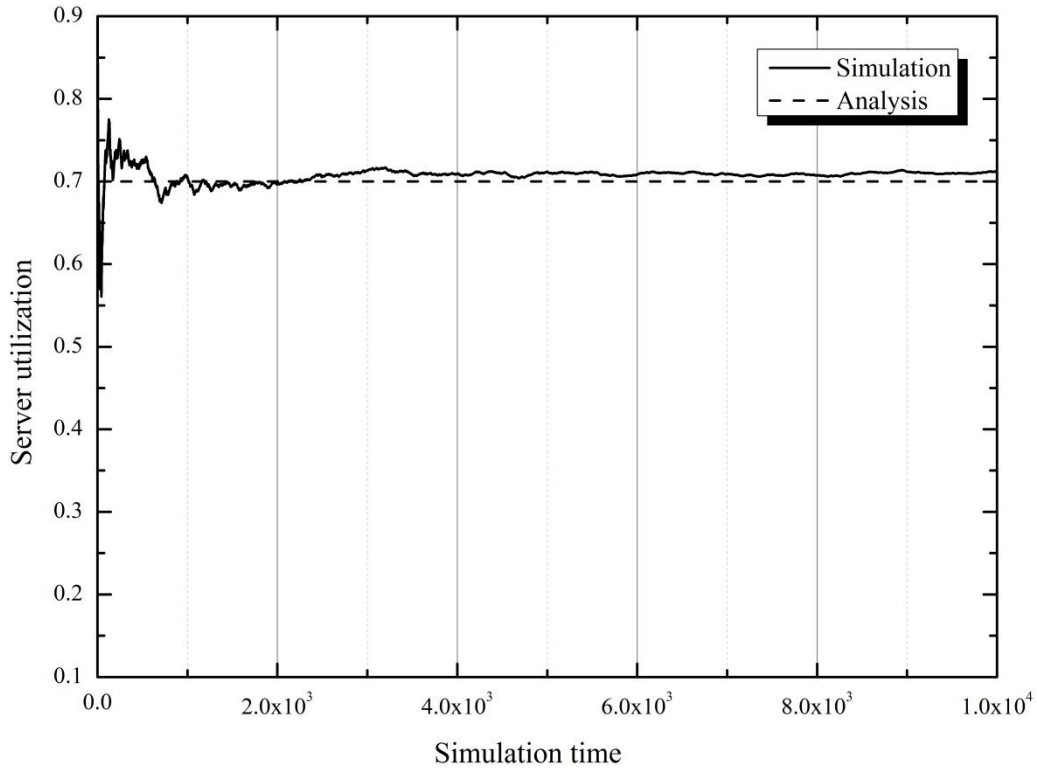


Figure 8.12: Server utilization of traditional routing when λ_1=0.4, λ_2=0.3

B. Network coding server utilization

Similar to TR, NC utilization is given by (8.5) for the total arrival rate λ of the coded packets. Applying (8.5) for λ_1=0.4 and λ_2=0.5, a value of U=0.295 is obtained and presented in Figure (8.13). Comparing the results shown in Figures 8.12 and 8.13, one can conclude that the server will be required to work less in the case of the NC system, since it will receive fewer packets due to packet combining, which is desirable for efficient packet handling.

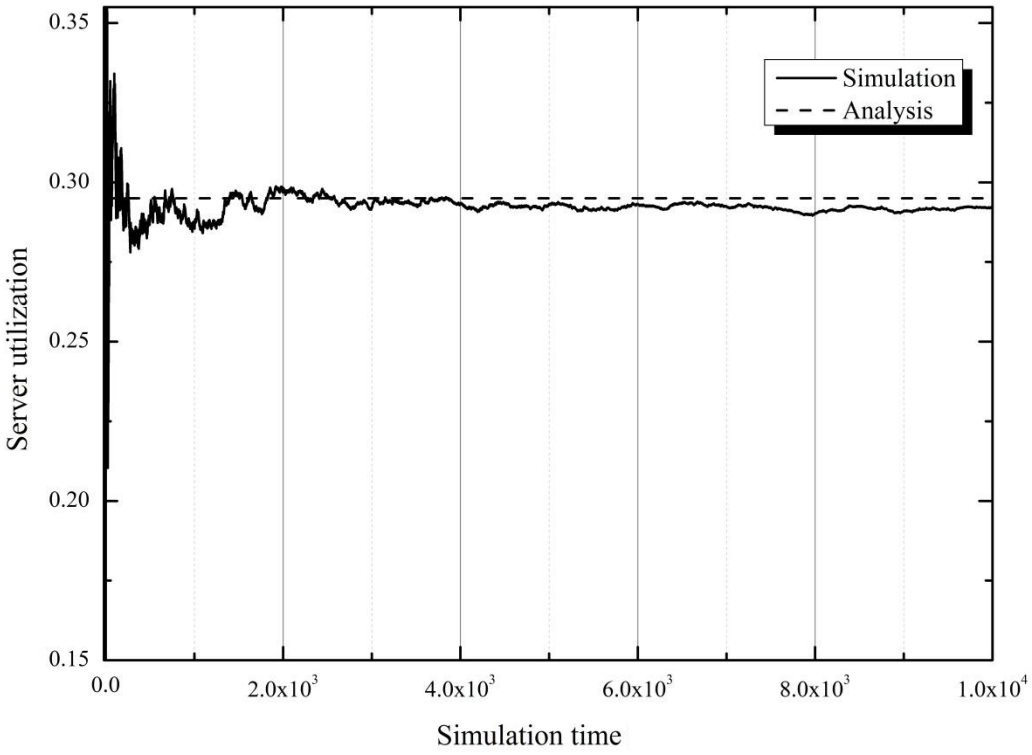


Figure 8.13: Server utilization of NC when λ_1=0.4, λ_2=0.5

8.5 Conclusions

In this chapter, a NC-based M/D/1 queuing model has been developed and compared with the traditional routing scheme under different packet arrival rates. The mean arrival rate of the coded packets is derived, and verified via simulation. The analysis is further validated by simulating the mean waiting time and the server utilization for both TR and NC scenarios. The results have many implications in several aspects of entity-controlled computer systems, communication networks and manufacturing systems; in all of these systems input buffers prior to the assembly stage are not applicable.

Chapter 9

Conclusions and Future Work

Throughout the preceding chapters, it has been shown that network coding (NC) can greatly influence the throughput gain of a network, by exploiting its ability to mix packets at certain intermediate nodes. Through allowing the algebraic combination of packets at intermediate nodes, NC can be employed as a replacement for traditional *store-and-forward* routing; in other words, there are ways to combine - and later extract - the independent data streams produced and consumed in the network. This is particularly evident when source messages share several common links as they move to multiple destinations.

When employed in a network adopting error-control schemes, NC can reduce the number of ACK/NAK signals needed at each transmission, leading to better network performance. In addition, adopting NC can reduce the number of packets needed to convey a set of information, thereby reducing congestion at particular nodes. This consequently reduces the amount of time a packet remains in the intermediate node queue before being served. Therefore, in this thesis, the focus is mainly on the performance of NC from the perspective of error-control techniques and queuing theory.

Overall, the key aims of this research project pertaining the achievement of higher throughput when network coding is implemented along with error-control protocols and stable queuing systems at intermediate nodes have been fulfilled. **Chapter 2**

provides a research background to NC and its application in both error-control schemes and queuing systems. Some examples are stated to illuminate the need for NC in some specific network situations; principally when more than two information flows overlap to share a common link to their destination. When implementing NC along with error control schemes, the number of ACK/NAK signals can be reduced, hence improving the network throughput. Additionally, from the perspective of coded packets, adopting synchronous NC at an intermediate node reduces the number of packets required to convey a set of information, and consequently coded packets will spend less time in the intermediate node's output queue before they are served.

Chapter 3 illustrates the three main ARQ schemes, namely stop-and-wait (SW), go-back-N, and selective repeat (SR). A purpose built control unit was constructed to model the SW ARQ protocol using SimEvents®. The major advantages of SW ARQ are its simplicity and the fact that it requires minimum buffer storage. However, when the propagation time becomes larger than the transmission time, SW ARQ becomes inefficient in its use of transmission bandwidth, due to the increase in the sender idle time. This motivates the additional exploration of the benefits that NC can provide to a continuous packet transmission scheme, SR ARQ, and hence a detailed theoretical background on this scheme is provided within this chapter.

Chapter 4 analyses the benefits of performing NC in a SW-ARQ-based point-to-point communication system using a Vandermonde full rank matrix. A communication model was built and simulated, where a packet source conveys a collection of coded packets and receives an acknowledgment for this collection instead of individual packets. Results show that, in the presence of higher data rates

and longer round-trip delays, conventional SW ARQ becomes inefficient due to the increased retransmissions and transmitter idle time.

An increase in throughput of between 2.5% and 50.5% at each link is found when NC is used along with SW-ARQ, and minimal decoding delay is seen. The advantage of NC is found for packet error rates of up to 0.4 and a number of coded packets k > 4; above this, the throughput of SW-ARQ begins to decrease because of the retransmission of large generations.

Chapter 5 focuses on the architecture modelling and performance study of adopting NC along with Selective Repeat (SR) ARQ in a unicast networks, and in a simple butterfly multicast network. A detailed simulation model for the network topology and elements necessary was, therefore, built. The results conclude that NC is useful in applications with low-bandwidth channels, having low values of α (the ratio of the propagation to the transmission time), and also in those prone to errors with high values of α. In a multicast network, NC showed a significant advantage over SR-ARQ in terms of throughput, with improvements of around 100% at lower bandwidths. It is clear that NC can be employed to decrease the congestion that is found at a shared server, in the situation where several packets targeting the same destination must be processed. NC also increases the usable range from $\alpha \leq 10$ to $\alpha \leq 90$, when packet errors occur at a non-coding link.

In Chapter 6, the butterfly network evaluated in **Chapter 5** is extended to include two coding nodes and three sinks. A total of 6 scenarios are examined, considering both traditional routing and NC-based SR ARQ, resulting in several conditions where NC shows better performance.

The model implemented provides throughput performance results for both routing-based and NC-based scenarios, subject to various data rate, signal to noise ratio and transmission delay values.

It can be seen that NC provides a throughput improvement of 3% to 48% when operating in error-free conditions, and a mean improvement of 50% in erroneous links. The results also demonstrate the conditions under which the advantages of NC can be maximized. The chapter further shows that a throughput improvement of 13% to 50% can be achieved in networks having variable packet lengths or transmissions delays. The new synchronized NC technique described suffers from some reduction in performance, but with long packet delays and low SNR conditions, an overall throughput gain of 78% to 86% is possible.

Chapter 7 predicts the performance of a computer network employing synchronous NC at its intermediate nodes, from a queuing theory perspective. Mathematical derivations and analytical results for an M/M/1 model reveal that the derived *pdf* of the coded packets obeys a random general distribution, and extensive simulations are presented to substantiate the constructed distribution.

Furthermore, the mean waiting time which the packets spend in the output queue of an intermediate node was constructed and verified via simulation. The results show that NC offers a new solution towards improving network stability, since it significantly reduces the amount of time the coded packets must wait in the queue. The analysis was further verified by investigating the server utilization for both traditional routing and NC scenarios. Besides, the packet loss probability of an M/M/1 model with limited queue capacity was also examined.

The results obtained in **Chapter 7** are utilized in **Chapter 8** to carry out a simulation-based study on a special case of an M/G/1 queue, M/D/1. The mean waiting time and the server utilization for the traditional routing and NC were investigated. Similar results were obtained in this work, which suggests that NC is useful in network applications where nodes have more than one incoming input addressing one output port.

Future Work

In this thesis, only multicast scenarios with one and two coding nodes were considered. Hence, a possible avenue for future research is to study NC with a larger and random network topology. The work presented in this thesis can also be implemented on dynamic networks where any link can be disconnected at any time [106].

The work in the first part of the thesis has mainly concentrated on SW and SR ARQ schemes. As future research, this work can be extended to encompass the GBN ARQ scheme in both the unicast and multicast scenarios. The ARQ schemes in this thesis can as well be combined with Forward Error Correction (FEC) techniques, forming Hybrid ARQ (HARQ).

The work in the second part of the thesis focused mainly on the queuing aspect of NC. The packets are assumed to arrive safely at the intermediate nodes without considering any error control schemes. Therefore, the error control schemes studied in the first part of the thesis can be integrated with the queuing systems representing real world scenarios.

Moreover, the optimum time that an intermediate node should wait before taking a *classic routing* decision also seems worthy of consideration. This will reduce the effect of intractability at intermediate node input buffers resulted from waiting all the required packets to arrive and allow a maximum number of suitable packets to be coded. This work can be implemented by having input buffers with only one packet capacity. When all input buffers have packets, the coding node performs NC and sends out the coded packet. Whereas, when one buffer is empty and the others are

not, the coding node can wait for a pre-set optimised delay to increase the coding opportunities.

The study in this thesis has dealt solely with a single queuing system having different input links. A natural extension to this study is to look at a network with a collection of interactive queues where the departure process of some queues feeds into others. Hence the analysis approach carried out in Chapters 7 and 8 can be applied on Tandem networks in which the service facilities are located in series and packets or customers pass through them sequentially [104].

Moreover, since a simulation-based analysis was carried out to investigate the mean waiting time coded packets spend in the M/D/1 queue in Chapter 8, a mathematical analysis could be carried out to verify the results obtained via modelling. This requires new methods for studying the special case of G/G/1 system leaving behind the simplification driven from the Markovian property as it will no longer be applicable. In this system, the inter-arrival and service times between packets are independent and given by an arbitrary distribution. One of the methods that can be suggested to solve such system is the spectrum factorization method [107].

References

[1] R. Ahlswede, C. Ning, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, pp. 1204-1216, 2000.

[2] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard, "The importance of being opportunistic: Practical network coding for wireless environments," 2005.

[3] S. Katti, H. Rahul, H. Wenjun, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *IEEE/ACM Transactions on Networking*, vol. 16, pp. 497-510, 2008.

[4] D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker, "Applications of error-control coding," *IEEE Transactions on Information Theory*, vol. 44, pp. 2531-2560, 1998.

[5] T. V. Ramabadran and S. S. Gaitonde, "A tutorial on CRC computations," *Micro, IEEE*, vol. 8, pp. 62-75, 1988.

[6] C. Fragouli, D. Lun, M. Medard, and P. Pakzad, "On Feedback for Network Coding," in *41st Annual Conference on Information Sciences and Systems (CISS)* 2007, pp. 248-252.

[7] P. Parag and J. Chamberland, "Queueing Analysis of a Butterfly Network for Comparing Network Coding to Classical Routing," *IEEE Transactions on Information Theory*, vol. 56, pp. 1890-1908, 2010.

[8] B. Shrader and A. Ephremides, "Queueing delay analysis for multicast with random linear coding," *IEEE Transactions on Information Theory*, vol. 58, pp. 421-429, 2012.

[9] MathWorks, "SimEvents User's Guide.," 2005.

[10] M. A. Gray, "Discrete Event Simulation: A Review of SimEvents," *Computing in Science & Engineering*, vol. 9, pp. 62-66, 2007.

[11] X. Xianchao and W. Zhongjie, "Networked modeling and simulation based on SimEvents," in *7th International Conference on System Simulation and Scientific Computing (ICSC)*, 2008, pp. 1421-1424.

[12] M. I. Clune, P. J. Mosterman, and C. G. Cassandras, "Discrete Event and Hybrid System Simulation with SimEvents," in *8th International Workshop on Discrete Event Systems (WODES)*, 2006, pp. 386-387.

[13] http://www.isi.edu/nsnam/ns/, The Network Simulator v2.

[14] O. Users' Manual, "OPNET Architecture, OV. 415," ed, 2010.

[15] G. F. Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M. J. Reed, "Opnet modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed," *WSEAS Transactions on Computers*, vol. 2, pp. 700-707, 2003.

[16] T. Tat and A. Girinsky, "Evolution of Store-and-Forward Techniques," *IEEE Transactions on Communications*, vol. 22, pp. 1297-1301, 1974.

[17] S. Y. R. Li, R. W. Yeung, and C. Ning, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, pp. 371-381, 2003.

[18] Z. Li, B. Li, D. Jiang, and L. C. Lau, "On achieving optimal throughput with network coding," in *Proceedings IEEE of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)* 2005, pp. 2184-2194.

[19] M. Langberg, A. Sprintson, and J. Bruck, "The encoding complexity of network coding," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 2386-2397, 2006.
[20] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proceesings of the IEEE International Symposium on Information Theory*, 2003, p. 442.
[21] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, *et al.*, "Network coding for wireless applications: A brief tutorial," in *Workshop wireless Ad-hoc Sensor Network*, 2005.
[22] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, *et al.*, "Polynomial time algorithms for multicast network code construction," *IEEE Transactions on Information Theory*, vol. 51, pp. 1973-1982, 2005.
[23] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 782-795, 2003.
[24] D. S. Lun, M. Médard, and R. Koetter, "Efficient operation of wireless packet networks using network coding," in *International Workshop on Convergent Technologies (IWCT)*, 2005.
[25] D. S. Lun, M. Medard, R. Koetter, and M. Effros, "Further results on coding for reliable communication over packet networks," in *Proceedingsof the International Symposium on Information Theory (ISIT)* 2005, pp. 1848-1852.
[26] P. Chou, Y.Wu, and K.Jain, "Practical network coding," in *Allerton Conference on Communication, Control and Computing* 2003.
[27] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *IEEE Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)* 2005, pp. 2235-2245.
[28] C. Fragouli, J.-Y. L. Boudec, and J. Widmer, "Network coding: an instant primer," *SIGCOMM Computter Communication Review*, vol. 36, pp. 63-68, 2006.
[29] D. Katabi, S. Katti, H. Wenjun, H. Rahul, and M. Medard, "On Practical Network Coding for Wireless Environments," in *International Zurich Seminar on Communications* 2006, pp. 84-85.
[30] D. S. Lun, M. Médard, R. Koetter, and M. Effros, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, pp. 3-20, 2008.
[31] Y. Wu, P. A. Chou, and S.-Y. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," in *39th Annual Conference on Information Sciences and Systems (CISS)*, 2005.
[32] C. Fragouli, J. Widmer, and J.-Y. Le Boudec, "A Network Coding Approach to Energy Efficient Broadcasting: From Theory to Practice," in *INFOCOM*, 2006.
[33] R. W. Yeung and Z. Zhang, "Distributed source coding for satellite communications," *IEEE Transactions on Information Theory*, vol. 45, pp. 1111-1120, 1999.
[34] Y. Ma, W. Li, P. Fan, K. B. Letaief, and X. Liu, "On the characteristics of queueing and scheduling at encoding nodes for network coding," *International Journal of Communication Systems*, vol. 22, pp. 755-772, 2009.
[35] R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang, "Network Coding Theory Part I: Single Source," *Foundations and Trends® in Communications and Information Theory*, vol. 2, pp. 241-329, 2006.

[36] S. Jaggi, P.A.Chou, and K.Jain, "Low complexity optimal algebraic multicast codes," *International Symposium on Information Theor, Yokohama, Japan*, 2003.
[37] I. Woungang, S. Misra, and S. C. Misra, *Selected topics in information and coding theory* vol. 7: World Scientific, 2010.
[38] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, *et al.*, "Toward a random operation of networks," *IEEE Transactions on Information Theory*, 2004.
[39] T. Ho, M. M´edard, R. Koetter, D. R. Karger, M. E.ros, J. Shi, *et al.*, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, pp. 4413–4430, 2006.
[40] S. Katti, S. Gollakota, and D. Katabi, "Embracing wireless interference: analog network coding," in *ACM SIGCOMM Computer Communication Review*, 2007, pp. 397-408.
[41] M. Wang and B. Li, "Network coding in live peer-to-peer streaming," *IEEE Transactions on Multimedia*, vol. 9, pp. 1554-1567, 2007.
[42] S. Jaggi, M. Langberg, S. Katti, H. Tracey, D. Katabi, M. Medard, *et al.*, "Resilient Network Coding in the Presence of Byzantine Adversaries," *IEEE Transactions on Information Theory*, vol. 54, pp. 2596-2603, 2008.
[43] S. A. Aly and A. E. Kamal, "Network coding-based protection strategies against a single link failure in optical networks," in *International Conference on Computer Engineering & Systems (ICCES)* 2008, pp. 251-256.
[44] L. Baochun and N. Di, "Random Network Coding in Peer-to-Peer Networks: From Theory to Practice," *Proceedings of the IEEE*, vol. 99, pp. 513-523, 2011.
[45] A. A. Bruen and M. A. Forcinito, *Cryptography, Information Theory, and Error-Correction: A Handbook for the 21st Century*: John Wiley & Sons, 2011.
[46] J. Walrand, *Communication Networks: A First Course*: WCB/McGraw-Hill, 1998.
[47] A. Leon-Garcia and I. Widjaja, *Communication Networks: Fundamental Concepts and Key Architectures*: McGraw-Hill Education, 2006.
[48] M. Ghaderi, D. Towsley, and J. Kurose, "Network Coding Performance for Reliable Multicast," in *IEEE Military Communications Conference (MILCOM)* 2007, pp. 1-7.
[49] J. K. Sundararajan, D. Shah, and M. Medard, "ARQ for network coding," in *IEEE International Symposium on Information Theory (ISIT)* 2008, pp. 1651-1655.
[50] N. Dong, T. Tran, N. Thinh, and B. Bose, "Hybrid ARQ-random network coding for wireless media streaming," in *Second International Conference on Communications and Electronics (ICCE)* 2008, pp. 115-120.
[51] N. Dong, T. Tran, N. Thinh, and B. Bose, "Wireless Broadcast Using Network Coding," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 914-925, 2009.
[52] F. Chiti, R. Fantacci, R. A. Johnson, Crnojevic, x, V., *et al.*, "End-to-End Delay Analysis for Reliable Communications over Lossy Channels: Integrating Network Coding and ARQ Schemes," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2009, pp. 1-5.

[53] Q. Yang and Y. Lie-Liang, "Throughput Analysis of General Network Coding Nodes Based on SW-ARQ Transmission," in *72nd IEEE on Vehicular Technology Conference (VTC)*, 2010, pp. 1-5.
[54] Q. Yang and Y. Lie-Liang, "Throughput comparison of automatic repeat request assisted Butterfly networks," in *7th International Symposium on Wireless Communication Systems (ISWCS)*, 2010, pp. 581-585.
[55] A. Antonopoulos and C. Verikoukis, "Network Coding-Based Cooperative ARQ Scheme," in *IEEE International Conference on Communications (ICC)*, 2011, pp. 1-5.
[56] G. Mingsheng and W. Mingwei, "On the delay performance of selective-repeat ARQ for underwater acoustic channels," in *1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (VITAE)*, 2009, pp. 727-731.
[57] Z. Guo, B. Wang, P. Xie, W. Zeng, and J.-H. Cui, "Efficient error recovery with network coding in underwater sensor networks," *Ad Hoc Networks*, vol. 7, pp. 791-802, 2009.
[58] W. Chen, K. B. Letaief, and Z. Cao, "Opportunistic network coding for wireless networks," in *IEEE International Conference on Communications (ICC'07)* 2007, pp. 4634-4639.
[59] Y. E. Sagduyu and A. Ephremides, "Cross-Layer Optimization of MAC and Network Coding in Wireless Queueing Tandem Networks," *IEEE Transactions on Information Theory*, vol. 54, pp. 554-571, 2008.
[60] S. Chieochan, E. Hossain, T. Issariyakul, and D. Niyato, "Opportunistic network coding and dynamic buffer allocation in a wireless butterfly network," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2009, pp. 1-6.
[61] H. Xiang and A. Yener, "On the energy-delay trade-off of a two-way relay network," in *42nd Annual Conference on Information Sciences and Systems (CISS)*, 2008, pp. 865-870.
[62] O. H. Abdelrahman and E. Gelenbe, "Performance Trade-Offs in a Network Coding Router," in *Proceedings of 19th International Conference on Computer Communications and Networks (ICCCN)*, 2010, pp. 1-6.
[63] H. Seferoglu and A. Markopoulou, "Network Coding-Aware Queue Management for Unicast Flows over Coded Wireless Networks," in *IEEE International Symposium on Network Coding (NetCod), 2010* 2010, pp. 1-6.
[64] J. T. Charith Gunasekara, A. S. Alfa, and P. Yahampath, "A queueing theoretic model for opportunistic network coding," in *Computing, Networking and Communications (ICNC), 2013 International Conference on*, 2013, pp. 999-1004.
[65] D. Umehara, T. Hirano, S. Denno, M. Morikura, and T. Sugiyama, "Wireless network coding in slotted ALOHA with two-hop unbalanced traffic," *IEEE Journal on Selected Areas in Communications*, vol. 27, pp. 647-661, 2009.
[66] D. E. Lucani, M. Medard, and M. Stojanovic, "Random linear network coding for time-division duplexing: queueing analysis," in *IEEE International Symposium on Information Theory (ISIT)*, Piscataway, NJ, USA, 2009, pp. 1423-7.
[67] O. H. Abdelrahman and E. Gelenbe, "Queueing performance under Network Coding," in *IEEE Information Theory Workshop on Networking and Information Theory (ITW)* 2009, pp. 135-139.

[68] Y. Yuan, K. Wu, W. Jia, and Y. Peng, "On the queueing behavior of inter-flow asynchronous network coding," *Computer Communications,* vol. 35, pp. 1535-48, 2012.
[69] L. Kleinrock, *Queueing Systems: Theory*: Wiley, 1976.
[70] X. Yijun and H. Bruneel, "Buffer contents and delay for statistical multiplexers with fixed-length packet-train arrivals," *Performance Evaluation,* vol. 17, pp. 31-42, 1993.
[71] A. Karasaridis and D. Hatzinakos, "Network heavy traffic modeling using α-stable self-similar processes," *IEEE Transactions on Communications,* vol. 49, pp. 1203-1214, 2001.
[72] B. Kim, Y. Chang, Y. C. Kim, and B. D. Choi, "A queueing system with discrete autoregressive arrivals," *Performance Evaluation,* vol. 64, pp. 148-161, 2007.
[73] C. Yu, L. Xinhua, S. Wei, L. X. Cai, Z. Weihua, and S. Xuemin, "A cross-layer approach for WLAN voice capacity planning," *IEEE Journal on Selected Areas in Communications,* vol. 25, pp. 678-688, 2007.
[74] J. A. Schormans and J. M. Pitts, "Decay rate (ER) modelling of G/D/1 queue, and results for ATM telecommunications," *Electronics Letters,* vol. 34, pp. 943-945, 1998.
[75] S. Heckmuller and B. E. Wolfinger, "Reconstructing arrival processes to G/D/1 queueing systems and tandem networks," in *International Symposium on Performance Evaluation of Computer & Telecommunication Systems (SPECTS)* 2009, pp. 361-368.
[76] S. Y. R. Li, Q. T. Sun, and S. Ziyu, "Linear Network Coding: Theory and Algorithms," *Proceedings of the IEEE,* vol. 99, pp. 372-387, 2011.
[77] W. Stallings, *Computer Networking With Internet Protocols and Technology*: Pearson/Prentice Hall, 2004.
[78] B. A. Forouzan and S. C. Fegan, *Data Communications and Networking*: McGraw-Hill Higher Education, 2003.
[79] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*: Pearson-Prentice Hall, 2004.
[80] J. F. Kurose and K. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*: Addison-Wesley Longman Publishing Co., Inc., 2002.
[81] F. Halsall, *Data Communications, Computer Networks and Open Systems*, 4th ed.: Addison-Wesley, 1996.
[82] W. Stallings, *Data and Computer Communications, 8/e*: Pearson Education India, 2007.
[83] G. Khazan and M. A. Azgomi, "A distributed attack simulation for quantitative security evaluation using SimEvents," in *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)* 2009, pp. 382-385.
[84] K. Valigura, M. Foltin, and M. Blaho, "Transport System Realization in SimEvents Tools," *Technical Computing Prague,* 2009.
[85] J. Banks, J. Carson, and B. Nelson, *DM Nicol, Discrete-Event System Simulation*: Prentice Hall, 2000.
[86] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*: Springer, 2008.

[87] M. Ş. Kuran, H. B. Yilmaz, T. Tugcu, and B. Özerman, "Energy model for communication via diffusion in nanonetworks," *Nano Communication Networks*, vol. 1, pp. 86-95, 2010.
[88] X. Guo, M. R. Frater, and M. J. Ryan, "A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks," in *OCEANS 2006-Asia Pacific*, 2007, pp. 1-6.
[89] Q. Yang and Y. Lie-Liang, "Throughput Analysis of Stop-and-Wait Automatic Repeat Request Scheme for Network Coding Nodes," in *71st IEEE Vehicular Technology Conference (VTC)*, 2010, pp. 1-5.
[90] H. Shojania and L. Baochun, "Parallelized Progressive Network Coding With Hardware Acceleration," in *Fifteenth IEEE International Workshop on Quality of Service*, 2007, pp. 47-55.
[91] H. X. Nguyen and M. Roughan, "Rigorous Statistical Analysis of Internet Loss Measurements," *IEEE/ACM Transactions on Networking*, vol. 21, pp. 734-745, 2013.
[92] J. Chen, L. Liu, X. Hu, and F. Xu, "Hop-by-hop transport for satellite networks," in *Proceedings of IEEE Aerospace conference*, 2009, pp. 1-7.
[93] N. Abedini, S. Sampath, R. Bhattacharyya, S. Paul, and S. Shakkottai, "Realtime streaming with guaranteed QoS over wireless D2D networks," presented at the Proceedings of the 14th ACM international symposium on Mobile ad hoc networking and computing, Bangalore, India, 2013.
[94] G. Mingsheng, S. Wee-Seng, and T. Meixia, "A Transmission Scheme for Continuous ARQ Protocols over Underwater Acoustic Channels," in *IEEE International Conference on Communications (ICC '09)* 2009, pp. 1-5.
[95] F. Pingyi, Z. Chen, W. Chen, and K. Ben Letaief, "Reliable relay assisted wireless multicast using network coding," *IEEE Journal on Selected Areas in Communications*, vol. 27, pp. 749-762, 2009.
[96] Q.-T. Vien, L.-N. Tran, and H. X. Nguyen, "Efficient ARQ retransmission schemes for two-way relay networks," *Journal of Communication Software and Systems*, vol. 7, pp. 9-15, 2011.
[97] Z. Guo, P. Xie, J.-H. Cui, and B. Wang, "On applying network coding to underwater sensor networks," presented at the Proceedings of the 1st ACM international workshop on Underwater networks, Los Angeles, CA, USA, 2006.
[98] CISCO. (Jan 2008, April 2015). *Wireless Site Survey FAQ*. Available: http://www.cisco.com/c/en/us/support/docs/wireless-mobility/wireless-lan-wlan/68666-wireless-site-survey-faq.html#q26
[99] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad hoc networks*, vol. 3, pp. 257-279, 2005.
[100] H. Kobayashi and A. G. Konheim, "Queueing Models for Computer Communications System Analysis," *IEEE Transactions on Communications*, vol. 25, pp. 2-29, 1977.
[101] F. Gebali, *Analysis of computer and communication networks*: Springer, 2008.
[102] D. G. Kendall, "Some problems in the theory of queues," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 151-185, 1951.
[103] L. Takacs, "Introduction to the Theory of Queues. 1962," ed: Oxford University Press, New York.
[104] U. N. Bhat, *An introduction to queueing theory: modeling and analysis in applications*: Springer, 2008.

[105] C.-H. Ng and S. Boon-Hee, *Queueing modelling fundamentals: With applications in communication networks*: John Wiley & Sons, 2008.
[106] X.-B. Hu, M. S. Leeson, and E. L. Hines, "An effective genetic algorithm for network coding," *Computers & Operations Research,* vol. 39, pp. 952-963, 2012.
[107] W. L. Smith, "On the distribution of queueing times," in *Mathematical Proceedings of the Cambridge Philosophical Society*, 1953, pp. 449-461.