Spring 2019

Multiframe coded computation for distributed uplink channel decoding

Brinell F. Monteiro
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

MULTIFRAME CODED COMPUTATION FOR DISTRIBUTED UPLINK CHANNEL DECODING

by
Brinell F. Monteiro

The latest 5G technology in wireless communication has led to an increasing demand for higher data rates and low latencies. The overall latency of the system in a cloud radio access network is greatly affected by the decoding latency in the uplink channel. Various proposed solutions suggest using network function virtualization (NFV). NFV is the process of decoupling the network functions from hardware appliances. This provides the flexibility to implement distributed computing and network coding to effectively reduce the decoding latency and improve the reliability of the system. To ensure the system is cost effective, commercial off the shelf (COTS) devices are used, which are susceptible to random runtimes and server failures. NFV coded computation has shown to provide a significant improvement in straggler mitigation in previous work. This work focuses on reducing the overall decoding time while improving the fault tolerance of the system. The overall latency of the system can be reduced by improving the computation efficiency and processing speed in a distributed communication network. To achieve this, multiframe NFV coded computation is implemented, which exploits the advantage of servers with different runtimes. In multiframe coded computation, each server continues to decode coded frames of the original message until the message is decoded. Individual servers can make up for straggling servers or server failures, increasing the fault tolerance and network recovery time of the system. As a consequence, the overall decoding latency of a message is significantly reduced. This is supported by simulation results, which show the improvement in system performance in comparison to a standard NFV coded system.

MULTIFRAME CODED COMPUTATION FOR DISTRIBUTED UPLINK CHANNEL DECODING

by
Brinell F. Monteiro

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Telecommunications

Helen and John C. Hartmann Department of Electrical and Computer Engineering

May 2019

APPROVAL PAGE

MULTIFRAME CODED COMPUTATION FOR DISTRIBUTED UPLINK CHANNEL DECODING

Brinell F. Monteiro

Dr. Joerg Kliewer, Thesis Advisor — Date
Professor, New Jersey Institute of Technology

Dr. Ali Abdi, Committee Member — Date
Professor, New Jersey Institute of Technology

Dr. Alexander Haimovich, Committee Member — Date
Distinguished Professor, New Jersey Institute of Technology

BIOGRAPHICAL SKETCH

Author: Brinell F. Monteiro

Degree: Master of Science

Date: May 2019

Undergraduate and Graduate Education:

- Master of Telecommunications,
  New Jersey Institute of Technology, Newark, NJ, 2019
- Bachelor of Electronics and Telecommunication Engineering,
  Don Bosco Institute of Technology, Mumbai University, India, 2016

Major: Telecommunications

From not being able to add 2+2 to completing a Master's Degree in Telecommunications with a thesis, you know I could not have done it without you! You always told me that "Jack of all trades, master of none" was not meant for me because I could master it all. And you sure were right! I did it 'ma'! With God's help and your support, I found the patience and perseverance to complete this work! I dedicate this work to you mom, I know it's not an Oscar but it definitely feels like one!

Brinell F Monteiro

ACKNOWLEDGMENT

When the going gets tough, the tough gets going! I learnt this the hard way and this journey would not have been possible without the guidance and support of my thesis advisor Dr. Joerg Kliewer, who motivated me to take on the toughest of problems. Thank you for giving me the opportunity to pursue a master's thesis, for believing in me and encouraging me to aim high. I have learnt so much more than the ability to do quality research such as patience, perseverance and how to use every failure as a stepping stone. Thank you to Dr. Alexander Haimovich and Dr. Ali Abdi for being a part of my thesis committee and an inspiration to us all through your work and support. A special thanks to Dr. Roberto Rojas Cessa for the exceptional career guidance and support throughout my Masters. It is very easy to lose sight of the destination but you helped me stay on track and gave me the hope to try harder, you have truly been the best mentor and guide. I would also like to thank Dr. James Geller, for all the support and advice.

I thank all the students and faculty in the Center for Wireless Information Processing for all the help, especially Salman Habib for helping me think about the problem in more detail, Malihe Aliasgari for guiding me through the background work, Sarah Obead for her advice and inspiration, Alireza Bagheri for the motivation, Ishhanie Majumdar and Chen Yi for all the help and Anushreya Ghosh for being a great friend through this journey. I thank my mom, dad and brother for everything they have done to make this a possibility and for encouraging me till the end. Thank you Sancho Felix for believing in me and helping me stay focused. And thank you to all my friends that stood by me even when I had lost all hope. Last but not the least I would like to thank NJIT for all I have learned at this institution and for shaping me from a student into a professional.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Literature Review

The advancement in wireless communication systems has led to an increasing demand for higher data rates and low latencies. The latest 5G technology aims at improving the processing speed and fault tolerance of a network. In the year 2010 China Mobile Research Institute introduced the cloud radio access network (C-RAN) architecture, which brought about an architectural evolution in distributed communication systems. The centralization of information processing and resource pooling in the C-RAN architecture opened various research opportunities to implement virtualization techniques in the cloud. Network virtualization techniques introduced in LTE (Long Term Evolution) technology brought about significant improvements to the network performance as seen in [37]. Virtual machines running network functions are replicated to improve the fault tolerance of the network. Results from [6] show the extensive application of cloud computing and virtualization. Cloud computation is an evolving paradigm that has greatly improved the processing speed of networks by implementing network function virtualization (NFV) techniques in distributed computing. The most essential characteristic of cloud computing as defined in [26] is its ability to provide computing capabilities such as processing time and data storage on-demand. In a cloud radio access network, NFV provides solutions to scalability problems by offloading baseband functions such as uplink channel decoding to the cloud. In terms of latency, the uplink channel decoding is an expensive baseband function as seen in [4]. To ensure the upcoming technologies are cost effective, commercial off-the-shelf (COTS) devices are used in order to provision cloud computing. The use of COTS hardware decreases the efficiency of the system as

the devices are more susceptible to malfunctions and random execution time. Coding techniques are used in distributed computing as a method to mitigate straggling servers and server failures as seen in [34]. Coded computation adds redundant overhead which increases the processing time while decoding the information in the cloud. However, in [21] we see that coded computation in distributed computing has proven to improve the overall system performance. Since the processing time of each server varies, the decoding latency is dependent on the processing speed of individual servers. This allows straggling servers to increase the decoding time significantly. In addition to straggling servers, the network is also susceptible to faults in the network such as server downtime as described in [20]. The decoupling of network functions from the hardware using NFV provides the flexibility to implement coding techniques to reduce the decoding latency in the uplink channel. NFV coded computation introduced in [3] has proven to reduce the decoding latency by mitigating the work done by straggling servers. In this work, the standard NFV coded model is improved using a multiframe coded computation scheme that allows each server to process multiple frames of the same message. The multiframe model further reduces the decoding latency and improves the fault tolerance of the network.

Various solutions have been suggested to improve the reliability of distributed computing such as replicating the work on multiple servers as described in [11] and [35] or replication of virtual machines to run network functions as proposed in [27]. Network function virtualization (NFV) leverages flexibility to open network capabilities to address various challenges, offering new strategies to design and manage networks. Coded NFV as suggested in [8] enhances the robustness of channel decoding by using the algebraic structure of the transmitted coded message. Diversity based solutions are proposed in [2] by using channel coding to alleviate problems in virtual network functions. Novel coded computing techniques are used to overcome this problem by systematically addressing the performance analysis in a standard

coded NFV model used for C-RAN uplink decoding. Using this scheme provides a substantial gain in using a software based virtual network such as the cloud over traditional hardware based platforms. One of the main advantage is that the fault tolerance of the network increases, making it more flexible to use COTS devices. The makes the system cost effective. However using these COTS devices as servers to perform decoding in the cloud causes processing delays due to their random runtimes. Using coded computing to achieve reliability and low decoding latency in cloud computing is analyzed in detail in [3]. Key concepts from [3] are used to model the system in this work. By using suitable erasure codes operating over a fraction of the original information, a function can be completed without having to wait for all the servers to complete decoding. While this scheme works well to overcome straggling servers, the system is tolerant to a small number of stragglers. One of the suggested solutions to this problem would be to use rateless codes for load balancing in distributed computing. Rateless codes mitigate the redundant work done by stragglers, making the system much more efficient as suggested in [23]. This is done by allowing servers with lesser processing time to decode more than one packet without the master server performing any dynamic load balancing.

1.2 Cloud Radio Access Network (C-RAN)

In a traditional radio access network a connection is established between the base station and the user. In the C-RAN network, the model is shifted from individual baseband units (BBU) to a centralized control and processing terminal, referred to as a BBU pool. The baseband resources required by the remote radio units (RRU) are pooled together and made available at a centralized location. The BBU pool is connected to the RRUs using high speed optical fiber. C-RAN is cloud computing environment that dynamically handles interconnections within the network using hardware and interface cards. With the upcoming technologies such as 5G and IoT,

C-RAN plays a significant role the network architecture, enabling easy deployment, scalability and transition in technology as seen in [5]. C-RAN architecture supports more users and provides better management and control solutions as described in [14].

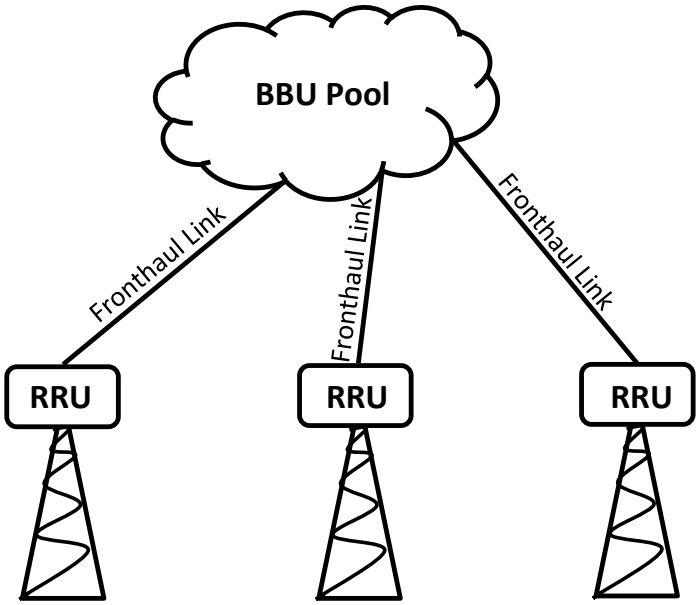

Figure 1.1 C-RAN Architecture.

The CRAN architecture introduced by China Mobile Research Institute is the concept of a collaborative, centralized, clean and cloud radio access network. The architecture consists of a distributed base station network. C-RAN network has a large coverage area, serving way more consumers than a traditional base station. As defined in [30] a centralized base band unit (BBU) is connected to a many remote radio units (RRU) via a fronthaul link. The BBU performs digital processing of the information such as baseband processing and packet processing. The RRU performs radio functions such as amplifying the signal, frequency conversion and analog to digital data conversion or vice versa. Centralization is the aggregation of more than one BBU into a single location. This improves the efficiency of the operation and control at the BBU by enabling resource pooling and advanced virtualization technologies. It also reduces the work load at the RRU, allowing deployment of

distributed RRUs especially in remote locations such as the edge of the network. Due to the centralization of the BBU, collaboration is possible between BBU processes such as information exchange and extensive computation. However this tends to increase the latency of the system due to the fronthaul transmission between the RRU and BBU. In such a network the control plane and the data plane are separated to reduce the latency. The RRU performs retransmission control decisions while the BBU performs data processing and decoding. The concept of integrating software and hardware in a logical and physical completeness separated from the consumer is adapted from cloud computing as in [14]. Distributed computing and virtualization techniques are used to process the information received at the BBU. By adopting the C-RAN architecture the spectral efficiency, as well as the energy efficiency of the network is improved as seen in [6], benefiting both the service providers as well as the consumers. The benefits of C-RAN as seen in [33] are reduction in cost, improved energy efficiency, better utilization of the spectrum, and transformation of the business model. The data processing depends largely on the quality of the communication channel and not solely on the information bits.

1.2.1 Uplink Channel

As seen in [33], let the number of BBUs in a network by K. The number of users in each BBU is M. The channel between the BBU and the users is given by a complex coefficient matrix H. The complex precoding matrix for the BBUs is W and the power allocation matrix is represent by P. Let x represent the signal for all M users. The received signal at a BBU is given by

$$y = HW\sqrt{P}x + n, \tag{1.1}$$

where the quantity $n =$ is an i.i.d. Gaussian noise vector such that $n \sim C\mathcal{N}(0, I)$. In order to calculate the weighting coefficients for the distributed antennas in different

BBUs, precoding and power allocations are performed. Iterative decoders that use capacity approaching codes such as low density parity check codes (LDPC) or turbo codes are commonly used. The system complexity is directly proportional to the number of decoding iterations. However, executing more iterations can potentially correct more errors thus allowing the system to operate at a low SNR. These decoders dominate the computational complexity for the uplink channel, requiring a lot more operations than a single process. The expected decoding complexity as proposed in [30] is given as

$$C(\gamma, \Delta_\gamma) \approx \frac{r(\gamma, \Delta_\gamma)}{\log_2(\zeta - 1)} \left[\log_2 \left(\frac{2 - \zeta}{K'} \log_{10}(\hat{\varepsilon}_{channel}) \right) - 2\log_2 l_k(\gamma, \Delta_\gamma) \right], \qquad (1.2)$$

where the quantity $\hat{\varepsilon}_{channel}$ is the channel outage constraint and $l_k(\gamma, \Delta_\gamma) = \log_2(1 + \gamma) - r(\gamma, \Delta_\gamma)$. The parameters ζ and K' are related to the connectivity of the decoder when represented as a graph. The computational effort required by a receiver with an iterative decoder is linear in the number of information bits and the number of iterations. To overcome the noise in the uplink channel the message is encoded using an LDPC code which is decoded using iterative decoding in the cloud.

The fronthaul is the link between the RRU and the BBU connected by optical fibers. A large amount of fibers are required for centralization on a large scale. Various schemes are suggested in [30] such as compression techniques, wavelength division multiplexing, optical transport networks and microwave transmission. Practical fronthaul has technology capacity or time delay constraints. As suggested in [29], compression and large scale pre-coding and decoding with low overhead is required to overcome the disadvantages of C-RANs imposed by the fronthaul constraints. Each RRU in the uplink forwards a compressed version of the received signal as a soft relay to the central BBU pool through the fronthaul link that has limited capacity. A combined decoding of all the users based on the received signals is performed by the centralized BBU pool. Signals received at multiple RRUs are often statistically

correlated as shown in [28]. Using distributed source coding stratergies instead of the traditional independent compression at each RRU has a significant benefit due to the correlation in the signals. The signals received from other RRUs is leveraged as side information, reducing the rate of the compressed stream by introducing resolvable uncertainty into the compressed signal. This improves the quality of the compressed signal received at the RRU. The quality of side information which is known to the encoder, determines the amount of reduction in rate that is allowed by the system without incurring any error during decompression. Recent work in [38] has been able to characterize the information theoretical capacity to within a constant gap. The uplink channel can be modeled as an instance of a general relay network with a single destination. In this work we consider only a single RRU communicating with a BBU to evaluate the decoding latency at the BBU for a single message. The suggested scheme can be extended to a model with multiple RRUs sending compressed information to the BBU pool.

1.2.2 Cloud Computing

The internet as we know it, is rapidly expanding, causing tremendous change in the world of computing. With the introduction of modern technologies such as machine learning and artificial intelligence the need for data storage and processing has increased exponentially. Cloud computing is used for efficient data access and storage as well as computation, moving the data processing and data storage from desktops and devices into large data centers as described in [16] and [9]. Cloud computing was introduced way back in the 1960s, by John McCarthy in [24] when he proposed the idea to one day organize computation as a public utility. As defined by the U.S. National Institute of Standards and Technology (NIST) in [26], cloud computing enables convenient, ubiquitous, and on-demand access to the network. The computing resources are shared by a pool of users and can be provisioned rapidly

without any interaction with the service provider. Cloud computing includes software services, hardwarde services as well as deployment models. The hardware and software present at the data centers is what constitutes the cloud as in [18]. A cloud can be available to a single organization, multiple organizations or both. Amazon Web Services is one of the largest cloud services currently available to multiple organizations. Cloud computing has provided opportunities for evolving technology to multiple companies in the Information Technology (IT) and Telecommunication industry. Companies such as Google are striving towards providing reliable, powerful and cost-effective cloud platforms. Computing service users do not need to invest in developing and maintaining IT infrastructure. Instead they pay providers only when they need to access the computing services. The various challenges that need to be addressed also open opportunities for research as described in [36]. An application is developed based on three essential building blocks, a model for computation, storage and communication. Cloud computing integrates existing technologies to provide an operational model that meets the technical and economic requirements of businesses across industries. Cloud computing is mainly aimed at improving the effective utilization of distributed resources and pooling the resources to achieve a much higher throughput and ability to process large scale computation. As seen in [16] cloud technology enables virtualization of resources, maintaining quality of service, providing scalable solutions and ensuring interoperability.

A cloud computing system can be explained with the help of a few essential characteristics as described in [26] and [10]. To facilitate a clear understanding of cloud computing we define key element of the system. The NIST has developed standards, guidelines and minimum requirements to provide consistency across the industry. The following characteristics are provided by NIST in [26] as a standard definition for cloud computing. As per [26] the essential characteristics are as follows.

1. On-Demand Self-Service

Cloud computing capabilities such as processing time and data storage are provisioned unilaterally as needed by a user without any interaction of a service provider. This is possible with the help of automation. A consumer is expected to perform all the necessary actions to acquire the resources by itself. This request is then automatically processed by the cloud and the resources are made available to the consumer to use. This requires a huge amount of planning and infrastructure management. With the help of virtualization technologies the process of on-demand self-service has become more flexible and feasible.

2. Broad Network Access

Computing capabilities hosted by a cloud should be accessible over a wide network range that promote its usage over various heterogeneous platforms such as smart-phones, desktops, laptops. Network access should be available to any location that offers online access. Creating a cloud with a vast network access raises a number of security concerns. However since most consumers use mobile devices the cloud network is intended to reach as many consumers as possible.

3. Resource Pooling

Infrastructure is quite expensive and difficult to modify once installed. Cloud computing resources such as processing time, bandwidth and data storage provided by a service provider are pooled using a multi-tenanat model to service the increasing demand of multiple consumers. Physical resources are virtualized so that based on the consumer needs they can be dynamically allocated. The consumer does not have any knowledge of the location of these resources but can specify their requirement at the desired location. Each consumer is isolated from the other but share the same underlying resource. This provides a layer of abstraction between the consumers and the resources. Large scale IT resources are pooled in this way to service a growing number of consumers. Advanced virtualization technologies have made this process scalable and efficient to a high degree. It enables service providers to use the existing

resources in a cost effective and efficient way.

4. Rapid Elasticity

The computation capability should be allocated and released automatically or elastically in such a way that the resources appear to be infinite to a consumer. At any given the time the consumer should be able to access any amount of resources. The resources need to be automated in a way to rapidly scale the outward and inward commensurate of resources with demand. The cloud provides access instantly on demand and releases the resources as soon as the task is complete to make it available to other consumers, providing high scalablilty services. Allocation and re-allocation of resources is modeled very strategically so that the provisioning is seamless to the consumer and resources are available instantaneously.

5. Measured Service

Usage of services such as bandwidth, storage, processing time and active users need to be constantly monitored and reported to service providers as well as consumers. Since most resources are pooled the management and control of the usage need to be transparent between the consumer and provider. Appropriate metering mechanisms are adopted to measure the provisions of each individual consumer for billing and also to evaluate the effective use of resources and planning of the system.

The advantages of cloud computing are seen in the services that the technology provides. The cloud computing service models serve as a foundation for important concepts in cloud computation. Each service can be developed and deployed independently.

1. Software as a Service (SaaS)

Service providers have combined infrastructure of hardware as well as software. Cloud computing provides the consumers capabilities to use applications running on the infrastructure of a service provider. Applications are released on a host device by the provider which can be accessible to the consumer through a program

interface or client interface from any client device such as a smartphone. The service provider controls and manages the network infrastructure. A few applications allow limited configuration access to consumers. Many different applications are maintained in a single logical environment achieving high speed, maintenance, and availability of services. Companies such Salesforce and Google provide Saas services like salesforce.com and Google Mail.

2. Platform as a Service(PaaS)

Cloud computing provides users a full development cycles for softwares which include programming languages, tools and libraries. Consumers can deploy acquired applications and consumer created infrastructure onto the cloud. Consumers have access to host applications and configure settings on the deployed applications. The service provider controls and manages the infrastructure of a network such as servers and operating systems.

3. Infrastructure as a Service(IaaS)

The ability to manage storage space, processing time, network access and other resources is given to the consumer. The user can deploy and run software applications and operating systems. The user has access to networking components such as firewalls and hosts, but does not control or manage infrastructure in the cloud.

Cloud computing makes the system cost effective and improves the flexibility for upcoming technologies. In this work we adopt the cloud computing model to deploy the uplink channel decoding in the cloud.

1.2.3 Distributed Computing

The pooling of resources such as data storage and computation capability requires a large amount of computation capacity. Since it is not practically possible for a single server to perform computation on such a large scale the processing is divided by multiple servers. A distributed computing system consists of one master server and

multiple slave servers. The master server performs administrative functions such as allocating the frames to the slave servers for processing. The work is divided among the slave servers and the information can be processed at a faster rate. The master server is responsible for all control and maintenance functions, while the slave servers process the information. Distributed computation is widely used in cloud computing to improve the efficiency of the system and reduce the latency.

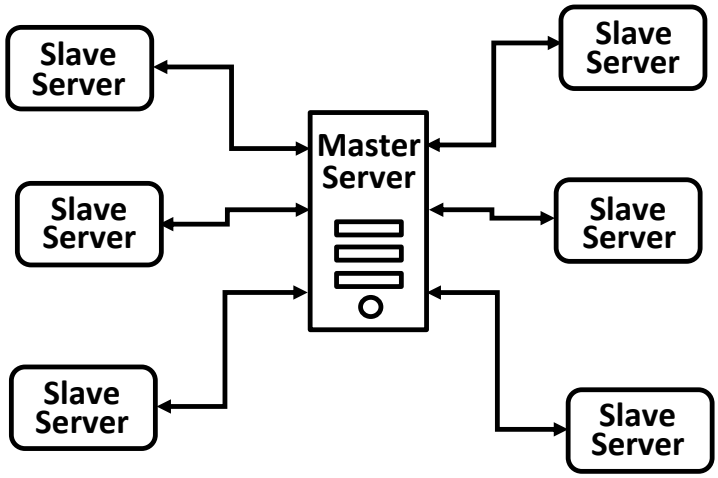

Figure 1.2 Distributed Computing.

To enable distributed computation in cloud computing, commercial off the shelf devices (COTS) are used to ensure the technology is cost effective. The COTS devices are susceptible to random runtimes and server failures, making the system unreliable. Due to the random runtimes of the COTS devices the latency of the system is highly dependent on the processing time of each individual server. This reduces the overall latency of the system. The COTS devices are also highly susceptible to failures, which introduce faults in the network. Improving the fault tolerance of the network and reducing the latency introduced by stragglers is the main focus of our work. Work has been done in the recent years to mitigate stragglers and improve the fault tolerance of the network. One of the suggested approach to the problem is using network function virtualization (NFV), which we will discuss in more detail.

1.3 Network Function Virtualization (NFV)

Network Function Virtualization (NFV) is a revolutionizing technology which enables a new consumption model for network infrastructure. The cloud services as discussed earlier are decoupled from their respective hardware resource and virtualized into network functions. Service providers are transitioning into a new consumption model based on the demand for services as described in [13]. NFV reduces operating expenses, capital expenses and facilitates deployment of new services. The decoupling of physical network equipment from their respective functions such as dispatching a firewall to a service provider as an instance of plain software is an example of implementing NFV. It is possible to consolidate various network equipment onto high volume devices, storage and switches that can be located at the user, in distributed networks or in large data centers. NFV provides a more efficient solution to controlling and managing resources.

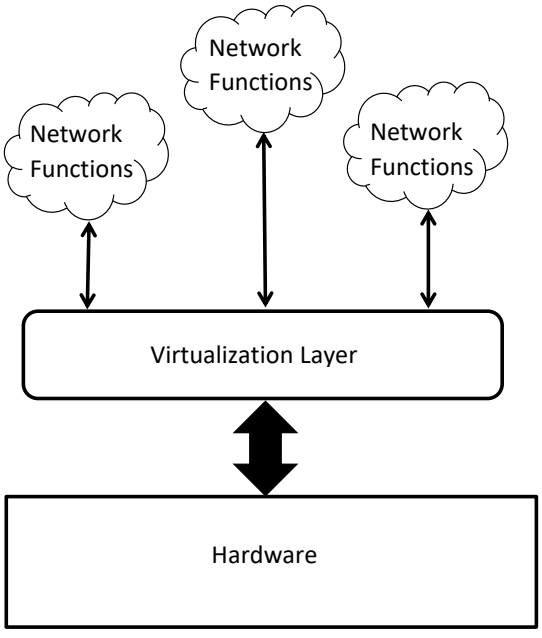

Figure 1.3 Network Function Virtualization.

1.3.1 Resource Virtualization

A given service is decomposed into virtual network functions (VNFs). The VNFs can later be used by software running on multiple servers as shown in [27]. Some of the fundamental differences in network services that NFV provides is decoupling of software from physical resources. Since the software and hardware are no longer integrated, both can evolve independently as network elements and do not need to depend on each other and be constrained by the limitations of one element. The software and hardware can have separate development and maintenance time lines. The virtualization of resources also provides flexible network function deployment. Reassignment and sharing of infrastructure resources is possible due to the detachment of software from hardware allowing each to services different functions at different times. This makes deploying new network services over the same physical infrastructure easier and faster for network operators. Resource virtualization also enables dynamic scaling of resources. The VNF performance can be dynamically scaled with great flexibility due to the decoupling of the network functions into instantaneous software components such as a network operator provisioning capacity based on live traffic.

1.3.2 NFV in 5G

As proposed in [31], NFV addresses various problems on virtualization by providing standard virtualized technologies to present multiple network devices. Clouds consisting of software implemented on high capacity devices can be utilized instead of using expensive routers and switches within the network. This helps to reduce the cost of the infrastructure, improve the power efficiency, reduce the geographical area required to deploy the equipment and the adaptation of the devices will be facilitated. The service operators can also have more flexibility with services based on geographical location and user demand. The resources from other operators can

be pooled more effectively at the same server. The advantages of moving from a traditional radio resources architecture, to a cloud based model for Ethernet RRUs and wireless BBUs is discussed in more detail in [39]. NFV overcomes a few problems in cloud computing as stated in [1] such as optimizing resource provisioning for reduced cost and power efficiency. It also mobilizes and scales VNFs between multiple hardware resources. The performance guarantees such as maximum allowable latency, maximum rate of failures for the operation of VNFs is ensured using the standards set by NFV. NFV provides great benefits in very dense networks which is a key requirement in the upcoming 5G technology. Unlike spare networks where cell sites make nearly autonomous radio resource management decisions as previously implement in 4G, in dense networks the terminal connects to the network through cluster of closest cells, which cooperatively minimize the impact of interference from neighbor clusters to which the terminal is not connected. In this way radio resource management decisions are logically centralized in a C-RAN architecture. Scalabilty issues in C-RAN are solved by deploying all control decisions that require cooperation of many cells in VNFs near the network core and rapid decisions in VNFs near the network edge. The logical centralization enables advanced algorithms to have access to an accurate update overview of the network status, interference maps, flow parameters and operator preferences as seen in [15]. Mobility management functions can base their decisions on network parameters beyond local radio quality at the cell cite, while still providing minimal service interruptions during handovers.

1.4 Channel Coding

A noiseless channel is depicted in Figure 1.4. The input binary digit is reproduced at the output exactly as it is seen at the input. Thus we can say that the transmitted bit is received without any error. During each transmission, we can reliably send

one binary digit to the receiver. In terms of capacity we see that the capacity of the channel is 1 bit.

Figure 1.4 Error Probability in a noiseless channel.

Communication systems usually consist of noisy channels. It is not practically possible to send information bits from the transmitter to the receiver without error. For this purpose we consider a binary symmetric channel (BSC) as illustrated in Figure 1.5. At the transmitter a binary input bit is sent through the channel and its

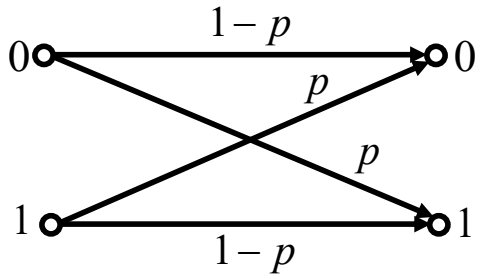

Figure 1.5 Error Probability in a BSC.

corresponding output received at the receiver is equal to the transmitted bit with a probability $1 - p$. The probability that transmitted bit is received in error is given by p. The capacity of the BSC channel in bits per transmission as given in [7] is calculated to be

$$C = 1 + plogp + (1 - p)log(1 - p) \tag{1.3}$$

The channel capacity is the limit on the rate of information communicated over a given channel. The limit can be achieved by using codes with infinite block length as shown in the channel coding theorem. Since practically implementing codes with infinite block length has various complexity limitations, it is not possible to achieve

full capacity of the channel. Near capacity approaching codes are used such as LDPC codes and turbo codes are used for practical implementation.

The operation on communication networks depends on the way information is processed. A communication network consists of nodes and links. Link in the channel in the network which can either be wired or wireless. A node is the point where two or more links are connected such as routers and switches. A node can forward received information and process independent incoming information. Independently produced data streams need not be kept separate as they traverse through the network. The information can be combined and later be extracted as individual elements. Combining the information allows better network traffic flow. Network coding utilizes COTS devices to increase the network throughput as described in [12]. Different coding techniques are used to encode and decode the original message. In this work we use coding not only to combat the channel noise by also to decode the message efficiently. Traditionally channel coding is used to transmit information bits through the channel to minimize error as described in [34].

Let us consider a message U consisting of k bits $u_1, u_2, ..., u_k$ is to be transmitted over a noisy channel. To combat the noise in the channel, the message is encoded at the transmitter using a (k, n) linear block code. The output of the encoder $X = x_1, x_2, ..., x_n$ is sent over the channel. The output of the channel is $Y = y_1, y_2, ..., y_n$. The receiver decodes the received message to obtain an estimate $\hat{U}$ of the original message. The output y_i at time i depends only on the input x_i at time i. Given that

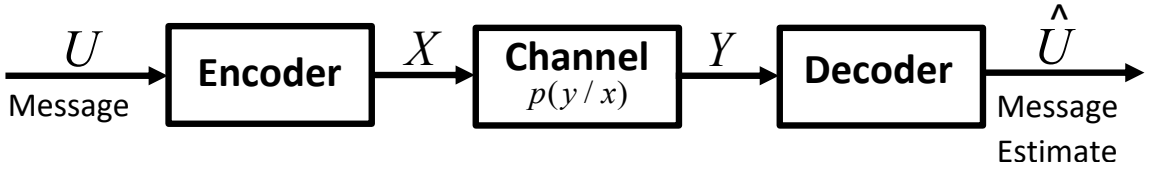

Figure 1.6 Channel Coding.

$x_1, x_2, ..., x_n$ is the input, the probability that $y_1, y_2, ..., y_n$ is the output of the channel

is given in [7] as

$$\prod_{i=1}^{n} p(y_i \,|x_i). \tag{1.4}$$

A symmetric channel has a transition matrix represented by Q. As described in [25], an $r-ary$ symmetric channel has a transition probability matrix $r \times r$ and $q_{xy} = \varepsilon$ if $x \neq y$, and $q_{xy} = 1-(r-1)\varepsilon$ if $x = y$, where $0 \leq \varepsilon \leq 1/(r-1)$. For $r = 2$ we get the binary symmetric channel with crossover probability ε. For $r = 4$ we get

$$Q = \begin{bmatrix} 1-3\varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 1-3\varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 1-3\varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 1-3\varepsilon \end{bmatrix} \tag{1.5}$$

The capacity of the r-ary symmetric channel is $\log r - H[1-(r-1)\varepsilon, \varepsilon, ..., \varepsilon] = \log r + (r-1)\varepsilon \log \varepsilon + (1-r\varepsilon+\varepsilon)\log(1-r\varepsilon+\varepsilon)$. For any $\varepsilon > 0$ and $R < C$, for a large n there exists a code of length n and rate $\geq R$ with at least 2^{Rn} distinct codewords and an appropriate decoding algorithm such that when the code is used on the channel, the probability of error is $< \varepsilon$, where C is the capacity of the channel. For an input X to the channel, the ith channel output Y is given by the equation

$$Y = X + Z, \tag{1.6}$$

where Z describes the channel noise. The channel capacity can be defined as

$$C = \max_{p(x)} I(X;Y), \tag{1.7}$$

where the maximum is taken over the input distribution $p(x)$.

CHAPTER 2

NFV CODED COMPUTATION

NFV provides the flexibility to implement coded computation in the uplink channel decoding as proposed in [3]. Coded computation is used to improve to reliability of the system. The coded message received in the cloud is further encoded by the master server to overcome the delays in processing and avoid straggling of servers. Using NFV coded computation is seen to have a notable gain in reducing the decoding latency at the cloud. The problem of straggling servers due to COTS devices is addressed in this chapter with the help of NFV coding solutions. NFV enables offloading critical baseband functions to the cloud. In terms of latency, uplink channel decoding is an expensive function. Continuous work has been done on reducing the overall latency in the uplink channel. This thesis addresses the problem of decoding latency in a standard NFV coded model. In this chapter we discuss the system model of a standard NFV coded model in a C-RAN network. We model the uplink channel and the decoding in the cloud to accommodate our proposed extension to the standard NFV model. In the standard NFV coded model the message is encoded by the user and transmitted over a binary symmetric channel (BSC) to the RRU. The RRU then sends the received message via the fronthaul link to the cloud to decode the message. The message is decoded in the cloud using NFV coded computation. An overview of the system model considered for this work is seen in Figure 2.1.

2.1 System Model

As illustrated in Figure 2.1, the mobile user sends an encoded message to the remote radio unit (RRU) via a binary symmetric channel. The message is then sent from the RRU to the cloud via the noiseless fronthaul link to decode the message. The BSC is a simple model for the uplink channel, while the the noiseless fronthaul link

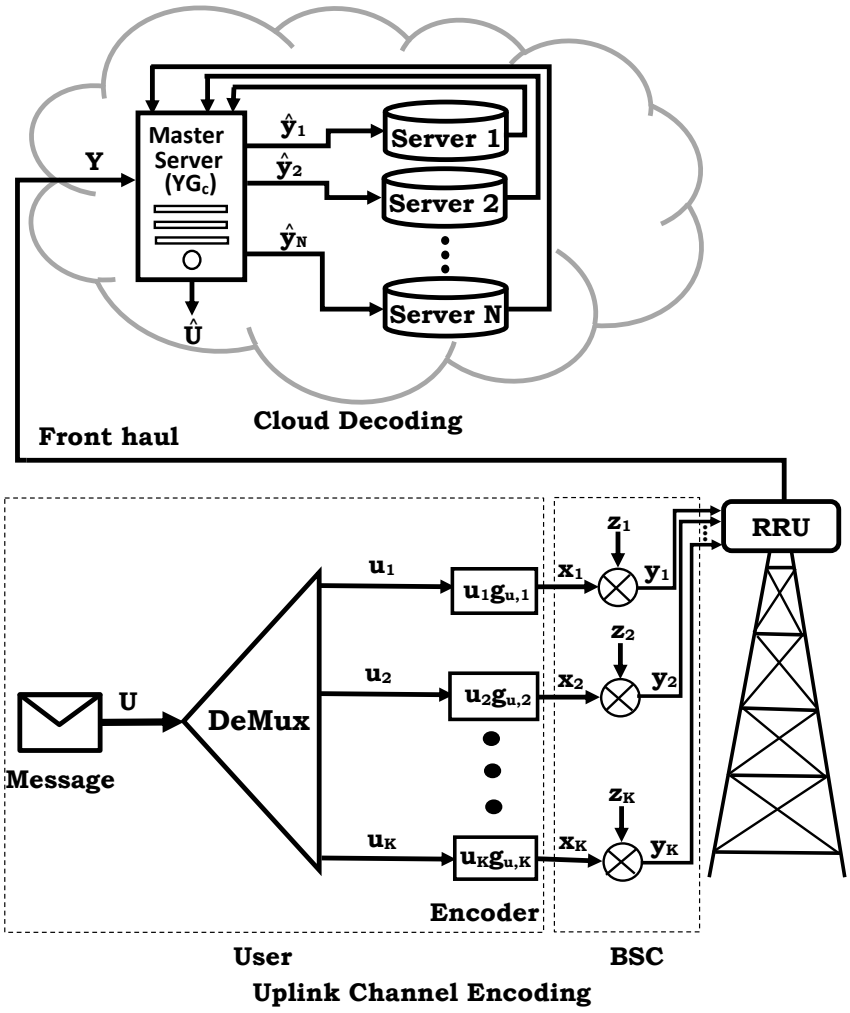

Figure 2.1 System model.

accounts for a deployment with higher capacity fiber optic cables. The cloud performs decoding of the information with the help of one master server and N slave servers. The master server encodes the received message into frames using a suitable linear block code with a minimum Hamming distance d_{min} operating over a fraction of the original data. A function can be computed as soon as any $N - d_{min} + 1$ frames have completed their operation, where d_{min} is the minimum Hamming distance of the generator matrix used by the master sever to encode the received message. We assume that the number of slave servers N available for decoding is greater than the number of received frames K. The master server then allocates one out of the $N \geq K$

frames to each slave server to decode. The slave servers processes each frame and are characterized by their random computing time. The system is highly dependent on the decoding latency of the servers. Using a suitable (N, K) linear block code makes the system tolerant to $N - d_{min}$ straggling servers and while increasing N increases the straggler tolerance, it also adds more redundant computation. The model is defined in further detail through the rest of this chapter.

2.1.1 Message Encoding

The first stage of the system model is the encoding of the message at the user to overcome noise in the BSC channel. An LDPC code is chosen as the user code due to its reduced decoding complexity at the cloud. The user send a single message U consisting of L bits. The message U is divided into K blocks of length n bits, $U = [u_1, u_2, ..., u_K]$ with the help of a demultiplexer. In order to overcome the noise

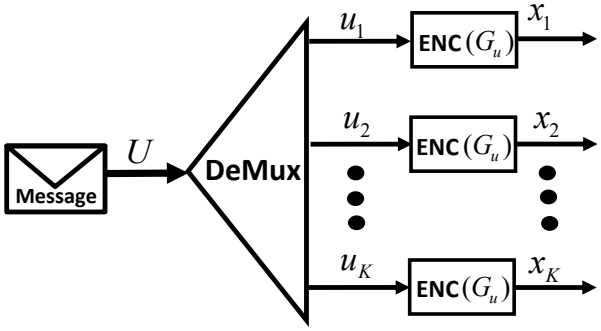

Figure 2.2 Encoded Message.

in the BSC, the message is encoded using a generator matrix $G_u \in \mathbb{F}_2^{n \times k}$, where $k = \frac{L}{K}$ is the length of each frame before encoding and $n = \frac{L}{(rK)}$ is the length of each encoded frame. Thus an (n, k) binary linear code C_u with rate $r = \frac{k}{n}$ is used to encode u_l, $l = 1, 2, ..., K$. The output of the encoder is given as

$$x_l = u_l G_l, \tag{2.1}$$

where x_l has length n bits. The encoded frames $x_1, x_2, ..., x_K$ are sent to the RRU.

2.1.2 Channel Characteristics

The uplink channel is a binary symmetric channel with a crossover probability of δ. Let $Z = [z_1, z_2, ..., z_K]$ represent the noise in the BSC. Z is a vector of i.i.d. Bernoulli random variables with crossover probability δ. The user sends the encoded message to the RRU. The output of the BSC is $Y = \{y_1, y_2, ..., y_K\}$, where each frame y_j

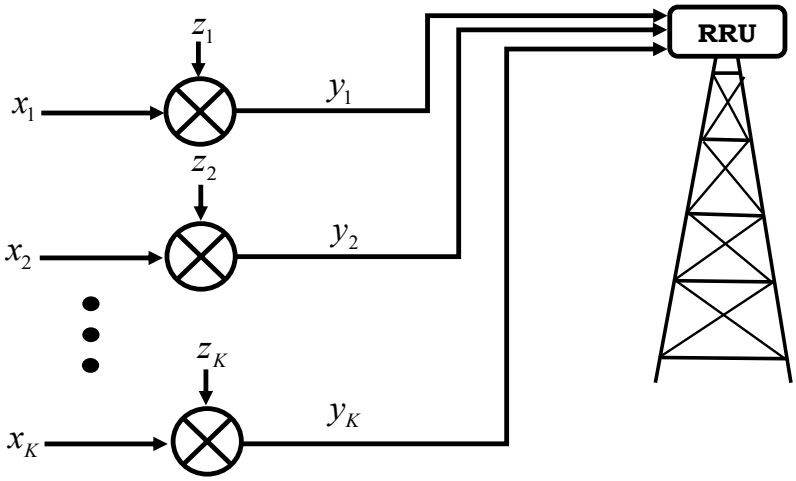

Figure 2.3 BSC Channel.

consists of n bits. The input to the RRU is seen as

$$Y = \sum_{j=1}^{K} y_j = \sum_{j=1}^{K} x_j + \sum_{j=1}^{K} z_j. \tag{2.2}$$

These received frames Y are sent to the cloud for decoding via the front haul link which is assumed to be noiseless.

2.1.3 Cloud Decoding

The signal $Y = [y_1, y_2, ..., y_K]$ received at the RRU is sent to the cloud to decode. The cloud consists of a distributed computing network, a master server that performs administrative functions and N number of slave servers that process the decoding of the frames. Y can be seen as an $n \times K$ binary matrix. Assuming that the number of servers N is greater than the number of frames K, the frames are linearly encoded

by the master server into N blocks, using an (N, K) binary linear NFV code with generator matrix $G_c \in \mathbb{F}_2^{N \times K}$.

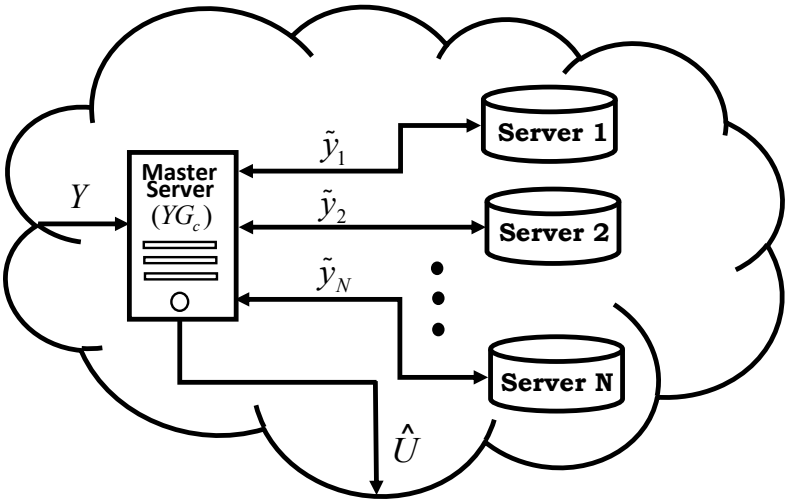

Figure 2.4 Cloud decoding.

2.1.4 NFV Decoder

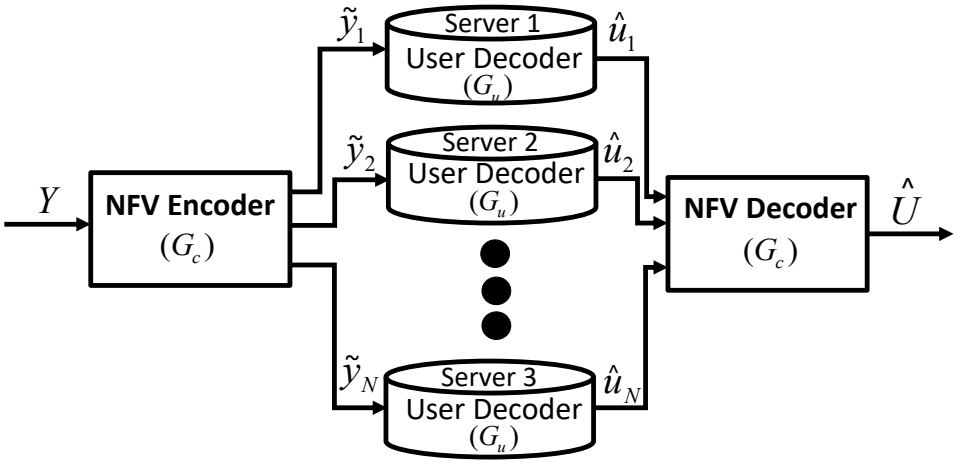

Figure 2.5 NFV Decoder.

The output of each server $\hat{u}$ is an estimate of a fraction of the original message U. An (N, K) NFV code with minimum Hamming distance d_{min} is chosen, such that the message U can be decoded once $N - d_{min} + 1$ frames have completed decoding. The time that a server takes to complete decoding a single frame is represented by

T_i, defined by a shifted exponential distribution that has a cumulative distribution function (CDF) given as

$$F(t) = \left\{ \begin{array}{ll} 1 - \mathrm{e}^{\left(-\frac{\mu}{n}(t-an)\right)}, & t \geq an \\ 0, \quad \text{otherwise} & \end{array} \right\}. \tag{2.3}$$

The time T_i has a shift of an, where a represents the minimum processing time per input bit and n represents the number of bits in the frame. The average time needed to process one bit is given by $\frac{1}{\mu}$. We assume that the processing time T_i for each server is mutually independent.

We now derive the probability of error of the system as a function of time. The probability that i servers complete decoding by time t is given as

$$\alpha_i(t) = F(t)^i (1 - F(t))^{N-i}, \tag{2.4}$$

where $F(t)$ is the CDF of time T_i. The parameter $\alpha_i(t)$ is used to derive the bounds on the probability of error as see in [17].

2.2 Straggler Mitigation

NFV coded computation was proposed by [3] as an effective strategy to mitigate straggling servers in cloud decoding. The COTS devices used have random runtimes. Each server i completes decoding a frame in a random time T_i. As illustrated in Figure 2.6, the processing time for each server is different. The servers with very large processing times introduce a significant latency into the system. The overall latency of the system in a distributed computing network is dependent on the latency of individual servers. NFV coded computation requires only $N - d_{min} + 1$ servers to complete decoding to obtain an estimate of the message. Thus the system is tolerant to $d_{min} - 1$ straggling servers. In Figure 2.6, once server 2 and 3 have completed decoding the information can be retrieved and thus server 1 does not need to complete

decoding the message. As seen, the reliability of the system is significantly improved using NFV coded computation.

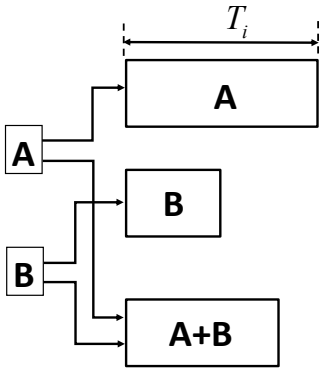

Figure 2.6 Random runtime of servers.

2.3 Linear Block Codes

A desirable property of linear block codes is its systematic structure of the codewords as shown in Figure 2.7, in which a codeword is divided into two parts, the message and the redundant checking part. The message part consists of k unaltered information bits and the redundant checking part consists of $n-k$ *parity-check* bits, which are linear sums of the information bits. A linear block code with this structure is referred to as a linear systematic block code as given in Figure 2.7.

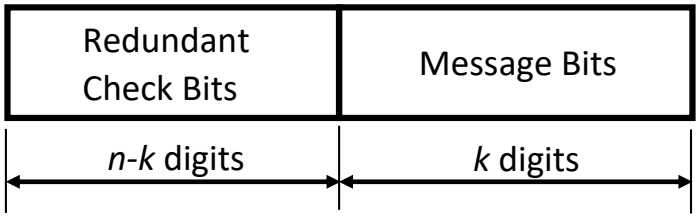

Figure 2.7 Codeword with a systematic structure.

A linear systematic (n, k) code is specified by a generator matrix G in the form

$$G = [P \quad I_G], \tag{2.5}$$

where P is a $k \times (n-k)$ matrix and I_G is the $k \times k$ identity matrix of G. For any $k \times n$ generator matrix G with k linearly independent rows, there exists an $(n-k) \times n$ parity check matrix H with $n-k$ linearly independent rows such that any vector in the row space of G is orthogonal to the rows of H, and any vector that is orthogonal to to the rows of H is in the row space of G as described in [32]. The parity check matrix H is given as

$$H = [I \quad P^T], \tag{2.6}$$

where P^T is the transpose of the parity matrix P in G with dimension $(n-k) \times k$ matrix and I is the $(n-k) \times (n-k)$ identity matrix. Linear block codes are generally defined in terms of *generator* and *parity-check* matrices. The minimum Hamming distance of a code d_{min} is defined as the smallest number of columns of H which need to be summed up such that they are equal to the all zero vector. As an example let N be the number of servers and K be the number of blocks the message is divided into. The (K, N) generator matrix for $G_c(2,3)$ is given as

$$G_c = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}_{2\times 3} \tag{2.7}$$

and the corresponding parity check matrix is given as

$$H_c = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}. \tag{2.8}$$

As we can see from H_c, the minimum Hamming distance for $G_c(2,3)$ is $d_{min} = 2$. The generator matrix can also be represented by by a dependency graph. The dependency graph consists of N nodes and has a link between each column in G_c that has a 1 in common in the same row. The dependency graph from $G_c(2,3)$ as defined above can be represented as shown in Figure 2.8.

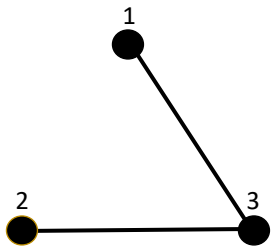

Figure 2.8 Dependency graph for $G_c(2, 3)$.

From the dependency graph we can obtain the chromatic number $\chi(G_c)$ of the code. The chromatic number is defined as the minimum number of colors needed to color the vertices of G_c such that no two adjacent vertices share the same color. For the purpose of this book we represent each color by a shape, a different shape represents a different color for illustration purposes only. The above dependency graph can be colored as shown in Figure 2.9.

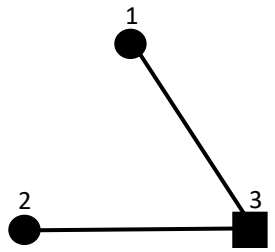

Figure 2.9 Chromatic number of the dependency graph for $G_c(2, 3)$.

As seen in Figure 2.9, two distinct shapes representing two distinct colors that are required to color the dependency graph such that no two adjacent nodes share the same color. Thus the chromatic number $\chi(G_c)$ for the code is $\chi(G_c) = 2$. Here we see that the chromatic number is equal to the minimum Hamming distance of the code. For a general case it can be shown easily that the minimum Hamming distance of a code is the lower bound to the chromatic number of the code.

2.3.1 LDGM Codes

Low-density parity-check (LDPC) codes represent the leading edge in modern channel coding. Their near-capacity performance on a large variety of data transmission and storage channels have drawn attention of coding theorists and practitioners. Since their decoders can be implemented with minimum complexity as seen in [22], LDPC codes are preferred over other codes such as turbo codes. In this work we use low density generator matrix (LDGM) codes, which are the dual codes of LDPC codes, as NFV codes. We construct an (8,16) LDGM code with low minimum distance and chromatic number. The generator matrix for a systematic LDGM code is given by

$$G = [P \quad I], \tag{2.9}$$

Like LDPC codes, LDGM codes are represented and decoded using a dependency graph as in [19]. The master server receives the input Y and re-encodes the message using a coded NFV constructed with the help of LDGM. The large deviation bounds derived in [17] show that the error probability of the system decreases with increasing minimum Hamming distance of the NFV code and increases with increasing chromatic number of the NFV code. In order to find an optimal trade off between d_{min} and $\chi(G_c)$ we designed a code that has $d_{min} = \chi(G_c)$. The parity matrix of the code was designed by generating a binary random 8×8 parity matrix, and then computing d_{min} and $\chi(G_c)$. An exhaustive search was performed over all parity matrices until a code with $d_{min} = \chi(G_c) = 3$ was obtained. The LDGM code used to encode the

message is

$$G_c = \left[\begin{array}{cccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]. \tag{2.10}$$

The minimum distance of the code $d_{min} = 3$ and the chromatic number of the code is given by the dependency graph in Figure 2.10. The minimum number of distinct

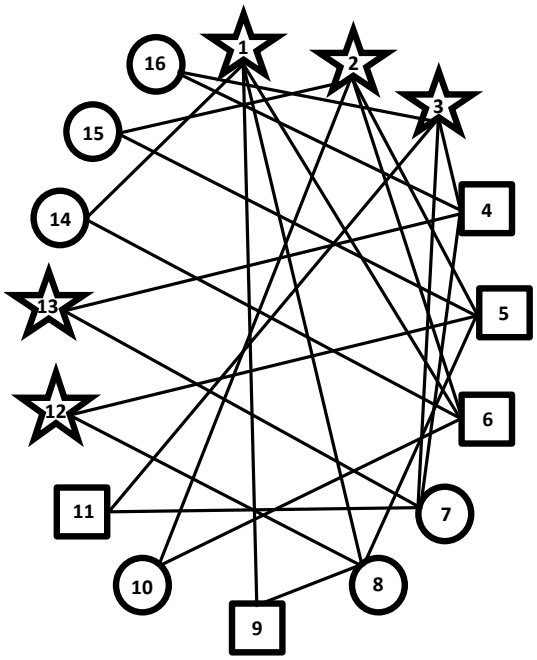

Figure 2.10 Dependency graph of the (8,16) LDGM code.

colors required to color the graph is 3. Thus the chromatic number is $\chi(G_c) = 3$.

2.3.2 Rateless Coding

Rateless fountain coding is a coding strategy proposed in [23] to mitigate straggling processors in distributed matrix-vector multiplication. Rateless coding is used for load balancing in systems with varying processing speeds, minimizing the overall computation time. A perfect load balancing scheme dynamically assigns one row vector product computation task to each other worker node as soon as the node finishes its current task. Thus faster nodes complete more tasks than slower nodes collectively completing the entire task. The rateless scheme achieves load balancing without communication overhead of dynamically allocating tasks. MDS codes use results from K nodes, and ignore the remaining $N-K$ nodes and thus do not adjust to the different degrees of node slowdown. Rateless codes achieve a much lower decoding latency as compared to MDS codes. An (N, K) MDS coded computation is robust to $N-K$ stragglers. Reducing K, increases the straggler tolerance but adds redundant computation. Rateleass coding can tolerate up to $N-1$ stragglers in the best case, with negligible redundant computation overhead. In order to use fountain codes we would require an infinite block length of information, which is not practically implementable. Thus we use the concept of rateless coding to develop a multiframe coded computation technique to reduce the decoding latency in distributed coded computation.

CHAPTER 3

MULTIFRAME CODED COMPUTATION

We propose a multiframe coded computation scheme as an extension to the standard NFV coded computation model. The multiframe coded computation scheme aims at improving the straggler tolerance of the system and reducing the decoding latency. The standard NFV coded model does not exploit the processing time of individual servers. In stead of each server decoding a single frame as seen in the standard NFV model, what if each server decodes multiple frames? By exploiting the variance in the server runtimes, we propose a model in which each server can decode multiple frames. In this way, the servers with low processing time can decode more frames achieving a significant gain in the overall decoding latency of the system.

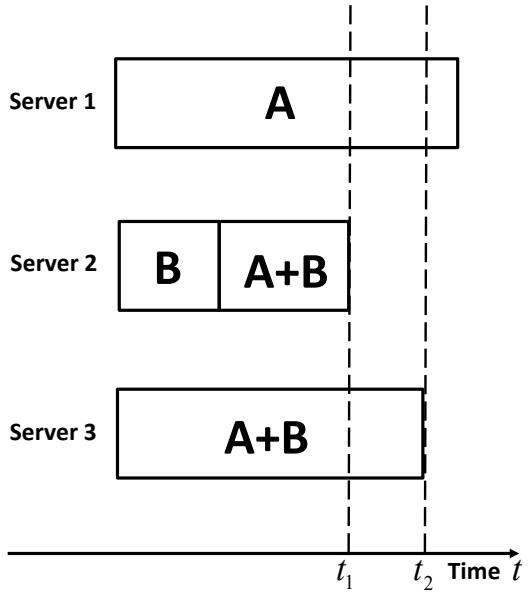

Figure 3.1 Multiframe coded computation.

Figure 3.1 illustrates the timing diagram for 3 servers. Each server begins decoding for a random time T_l^i, where l is the frame number and i is the server number. Once a server has completed decoding the first frame a new frame is allocated. We

assume a genie model that performs perfect dynamic allocation of frames. As seen in Figure 3.1, at time t_1 server 2 completes decoding two frames thus the master server can now retrieve the original message at time t_1. In a standard NFV model the master would complete decoding the message at time t_2. Thus we see that the decoding latency of the system is reduced. The standard NFV coded model does not exploit the processing time of individual servers. Although it uses coding to mitigate stragglers, servers with very high processing time increase the decoding latency. Since we need $N - d_{min} + 1$ frames to decode the message, in the multiframe scheme we allow each server to decode upto $N - d_{min} + 1$ frames. In the multiframe model if only one server is running and all others fail, the message can still be decoded. The multiframe model is tolerant to $N - 1$ stragglers in the best case, in comparison to the $d_{min} - 1$ straggler tolerance of the standard NFV model.

In this chapter we design a decoding algorithm in a C-RAN network based on the NFV coded computation model discussed in Chapter 2. We model the uplink channel and the decoding in the cloud based on a multiframe decoding scheme. As illustrated in Figure 2.1, the mobile user sends an encoded message to the remote radio unit (RRU) via a binary symmetric channel. The message is then sent from the RRU to the cloud via the noiseless fronthaul link to decode the message. The uplink channel decoding is a critical baseband function that is offloaded to the cloud. The BSC is a simple model for the uplink channel as discussed in Chapter 2, while the noiseless fronthaul link accounts for a deployment with higher capacity fiber optic cables. We assume that the number of slave servers N available for decoding is greater than the number of received packets K. The cloud performs decoding of the information with the help of one master server and N slave servers. The master server encodes the received message into frames using a suitable linear block code operating over a fraction of the original data such that a function can be computed as soon as any $N - d_{min} + 1$ frames have completed their operation, where d_{min} is

the minimum Hamming distance of the generator matrix used by the master sever to encode the received message. The master server then allocates one out of the $N \geq K$ packets to each slave server to decode. The slave servers process each frame and are characterized by their random computing time. Once a server has completed decoding the first frame, another frame out of the N frames is allocated to the server until the master server receives $N - d_{min} + 1$ frames. The proposed multiframe decoding scheme allows each server to decode multiple frames. If the servers have a large variance in processing time this model shows a significant gain in the decoding latency. The multiframe model can tolerate upto $N - 1$ stragglers with negligible redundant computation overhead improving not only the straggler tolerance but also increasing the fault tolerance of the system.

3.1 Decoding Latency

The decoding latency of the system is the overall decoding time for the master server to complete decoding the received message. The master server re-encodes the received coded message into frames using an NFV code and allocates the frames to the slave servers to decode. The master server completes decoding a message as soon as any $N - d_{min} + 1$ frames have completed decoding. The output of the master server is

$$\tilde{Y} = [YG_c], \tag{3.1}$$

where $\tilde{Y} = [\tilde{y_1}, \tilde{y_2}, ..., \tilde{y_N}]$ is an $n \times N$ matrix and $Y = [y_1, y_2, ..., y_K]$ is an $n \times K$ matrix. Let $g_{c,ji}$ be the $(j, i)^{th}$ entry of matrix G_c for the j^{th} packet where $j \in \{1, 2, ..., K\}$ and i^{th} server where $i \in \{1, 2, ..., N\}$. Therefore we get

$$\tilde{y_i} = \sum_{j=1}^{K} y_j g_{c,ji} = \sum_{j=1}^{K} x_j g_{c,ji} + \sum_{j=1}^{K} z_j g_{c,ji}. \tag{3.2}$$

Since $x_j g_{c,ji}$ is a linear combination of d_i codewords for a binary linear code with generator matrix G_u and rate r, it is a code word of the same code. The quantity Z

is the noise vector of i.i.d. Bern(γ_i) elements defined by the transition matrix of the BSC Q that has a crossover probability δ. The bit flipping probability γ_i is given by

$$\gamma_i = 2\sum_{j=0}^{d_i-1} \delta^{2j+1}(1-\delta)^{d_i-2j-1}. \tag{3.3}$$

where the Hamming weight denoted by d_i is the number of non-zero elements in the ith column of the generator matrix G_c. The NFV codes considered in this work have a column weight of 1 and 2. For $d_i = 1$, the bit flipping probability is

$$\gamma_i = 2\delta \tag{3.4}$$

For $d_i = 2$, the bit flipping probability is

$$\gamma_i = 2\delta(1-\delta) + 2\delta^3(1-\delta)^{-1}. \tag{3.5}$$

As the value of d_i increases γ_i decreases. The crossover probability for each frame depends on the column weight of the NFV code used.

The frame unavailability probability (FUP) denoted by $P_u(t)$ is defined as the probability that the i^{th} server decodes a frame $\widetilde{Y_i}$ successfully if the server completes decoding the entire frame within time t and without any errors. The FUP is given as

$$P_u(t) = P_r[\hat{\underset{u}{}}(t) \neq u], \tag{3.6}$$

where $P_u(t) = 0$ if the number of servers that successfully complete decoding is larger than $N - d_{min} + 1$. The time taken by a server to complete decoding a frame is given by T_i. Since each server can decode multiple frames we assume the processing time T_i is mutually independent for each server and each frame. The master server is able to decode the message when $N - d_{min} + 1$ servers have successfully completed decoding. In the multiframe model, each server can decode upto $N - d_{min} + 1$ frames. The time taken by the i^{th} server to complete decoding the l^{th} frame is denoted by T_l^i, defined

as

$$T_l^i = \sum_{l=1}^{N-d_{\min}+1} (an + A_l), \tag{3.7}$$

where A_l is an exponential random variable with mean $\frac{n}{\lambda}$. The largest l which satisfies the inequality

$$\sum_{l=1}^{N-d_{min}+1} an < t \tag{3.8}$$

provides an upper bound to the maximum possible frames a server can decode by time t, which is represented by M. The time taken by a server to complete decoding M frames is the sum of exponential random variables for each frame. The summation of l i.i.d. exponential distributions with rate λ can be represented as an Erlang distribution with parameter $\frac{l}{\lambda}$ and shape l. The time for a server to complete decoding multiple frames T_M^i is given by an Erlang distribution with CDF

$$F_M(t) = \left\{ \begin{array}{ll} 1 - \sum_{l=0}^{M-1} \frac{e^{-\left(\frac{\lambda}{n}\right)^l (t-lna)^l}}{l!} \left(\frac{\lambda}{n}\right)^l (t-lna)^l, & t \geq lna \\ 0, \qquad \text{otherwise} & \end{array} \right\} \tag{3.9}$$

3.2 Error Probability

The probability that the ith server completes decoding the lth frame in time t is given by an Erlang distribution with PDF

$$p_l^i(t) = \left(\frac{\lambda}{n}\right)^l \frac{(t-lna)^{l-1} e^{-\left(\frac{\lambda}{n}\right)(t-lna)}}{(l-1)!} \tag{3.10}$$

for $t > lna$ and $p_l^i(t) = 0$, otherwise. The frame error rate (FER) measures the probability that a server fails to decode a frame correctly at time t. Considering $N - d_{min} + 1$ frames have completed decoding by time t, the FER is defined as

$$P_e = \lim_{t \to \infty} P_u(t) \tag{3.11}$$

The FER is dependent on the crossover probability of the channel. The probability that the ith server corrects the errors caused by the BSC and decodes a frame correctly is given by $1 - p_e^i$, with

$$p_e^i = 1\{\hat{u_i} \neq u_i\}, \tag{3.12}$$

To find the FUP at each time t we define a discrete indicator random variable $I_i\,(t) \in \{0, 1, 2, ..., N - d_{\min} + 1\}$, $i \in \{1, 2, 3..., N\}$ with a distribution given by $1 - \eta_i^L(t)$ with

$$\eta_i(t) = \sum_{l=0}^{M} p_l^i(t) p_e^i, \tag{3.13}$$

denoting the average error probability at time t that L frames have been processed at server i, $M = 0, 1, 2, ..., N - d_{min} + 1$. The FUP is then given as

$$P_u(t) = \Pr\left[\sum_{i=1}^{N} I_i(t) \leq N - d_{\min}\right]. \tag{3.14}$$

The decoding latency versus FUP is simulated in the next chapter to evaluate the system performance of the multiframe NFV model.

CHAPTER 4

EXAMPLE AND RESULTS

We observe the frame unavailability probability (FUP) as a function of the decoding latency. The multiframe NFV model is compared to a standard NFV model. In this implementation we consider the transmission of a single frame in a binary symmetric channel (BSC) for uplink communication. We consider a genie model that performs an allocation of frames without repetition for the multiframe decoding scheme. The FUP measures the probability that a frame might not be decoded correctly or is unavailable at time t. The standard NFV model with single frame decoding is compared to the proposed multiframe decoding scheme. The error probabilities are evaluated via Monte Carlo simulations. In Figure 4.1 we consider the transmission of a message

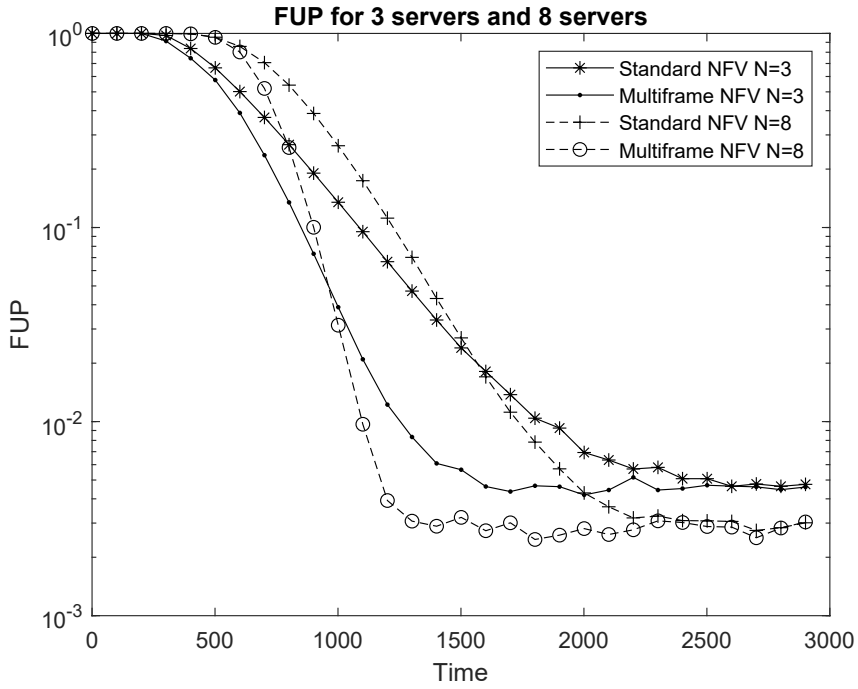

Figure 4.1 Decoding latency vs FUP for both a standard NFV and a multiframe NFV model. Parameters: $N_1 = 3$, $N_2 = 8$,$L_1 = 252$, $L_2 = 504$,$K_1 = 2$, $K_2 = 4$, $d_{min,1} = 2$, $d_{min,2} = 3$, $r = 0.5$, $n = 252$, $\delta = 0.07$, $\mu = 1$, $a = 1$.

with length $L_1 = 252$ and a number of servers of $N_1 = 3$. We use an LDPC user code with rate $r = 0.5$ and block length $n = 252$. The number of received frames is $K_1 = 2$. We use a (2,3) NFV code C_c with generator matrix G_c given as

$$G_c = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{4.1}$$

The minimum distance of the code is $d_{min} = 2$. The crossover probability of the BSC channel is selected to be $\delta = 0.07$, which leads to $\gamma_1 = 0.14$ and $\gamma_2 = 0.1309$. The minimum processing delay parameter is $a = 1$. As the decoding latency of the servers increase, the FUP decreases for $t > n$. Increasing the frame length increases the decoding latency. We consider a large variance in the processing times, and therefore select a value of $\mu = 1$. The master server finishes decoding as soon as $N - d_{min} + 1 = 2$ servers have finished decoding. As illustrated in Figure 4.1 the output shows that the decoding latency decreases with the multiframe decoding scheme. Using multiple servers in parallel for decoding and exploiting the processing time of each server yields significant gain in the trade-off between the decoding latency and the FUP.

In Figure 4.1 we also consider the transmission of a message with length $L_2 = 504$ and a number of servers of $N_2 = 8$. For the same LDPC user code as selected above for $N = 3$. The number of received frames is $K_2 = 4$. We use a (4,8) NFV code C_c with generator matrix G_c given as

$$G_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{4.2}$$

The minimum distance of the code is $d_{min} = 3$. The crossover probability of the BSC channel is again $\delta = 0.07$, which leads to $\gamma_1 = 0.14$ and $\gamma_2 = 0.1309$. The delay parameters are chosen again as $a = 1$ and $\mu = 1$. The master server finishes decoding

as soon as $N - d_{min} + 1 = 6$ servers have finished decoding. As already observed for $N = 3$, the multiframe NFV model shows a much lower decoding latency compared to the standard NFV model for $N = 8$ servers as well. We can also observe from Figure 4.1, that increasing the number of servers, results in a lower error floor. In addition, the reduction in decoding latency compared to the standard NFV model is much larger for $N = 8$ servers.

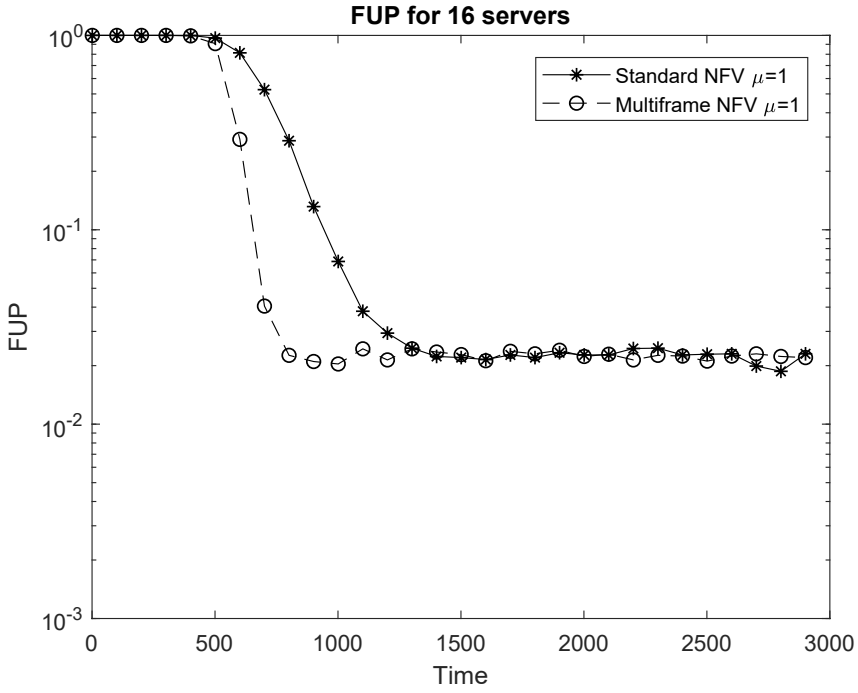

Figure 4.2 Decoding latency vs FUP for both a standard NFV and a multiframe NFV model. Parameters: $N = 16$, $L = 1008$, $K = 8$ $r = 0.5$, $n = 252$, $\delta = 0.07$, $\mu = 1$, $d_{min} = 3$, $a = 1$.

In Figure 4.2 we consider the transmission of a single frame with length $L = 1008$ and a number of servers of $N = 16$. The number of received frames if $K = 8$. The crossover probability of the BSC channel is again $\delta = 0.07$, which leads to $\gamma_1 = 0.14$ and $\gamma_2 = 0.1309$. All other parameters are kept constant as in Figure 4.1. We use an (8,16) NFV code C_c with generator matrix G_c given in equation (2.10). The master server finishes decoding as soon as $N - d_{min} + 1 = 14$ servers have finished decoding. Figure 4.2 shows that the decoding latency decreases with the

multiframe decoding scheme, where the improvement in delay performance is much more significant compared to the results for $N = 3$ and $N = 8$, respectively.

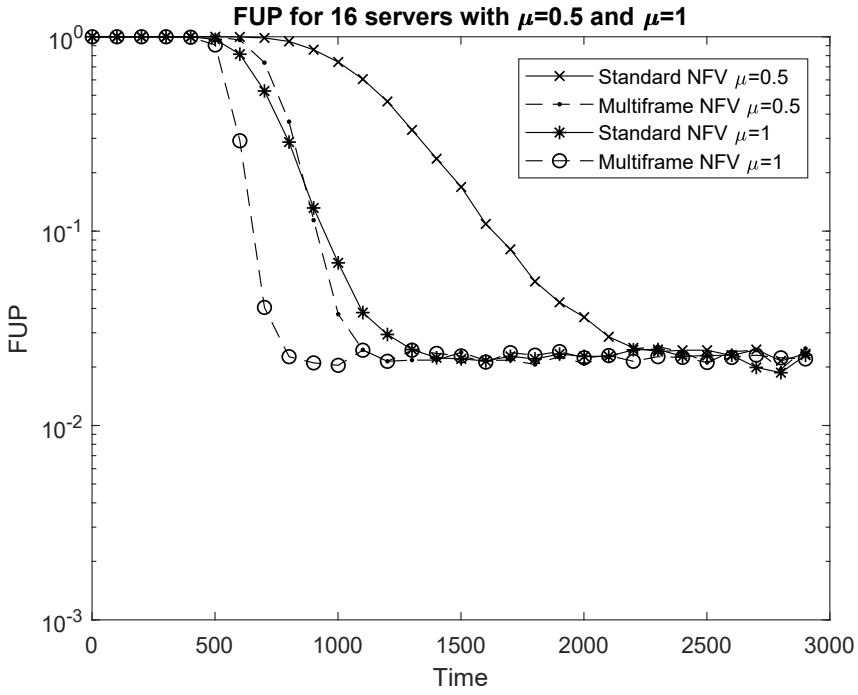

Figure 4.3 Decoding latency vs FUP for both a standard NFV and a multiframe NFV model comparing variance. Parameters: $N = 16$, $L = 1008$, $K = 8$ $r = 0.5$, $n = 252$, $\delta = 0.07$, $\mu = 1$, $d_{min} = 3$, $a = 1$.

Finally, Figure 4.3 shows the comparison between the standard NFV model and the multiframe model for $N = 16$ servers for both $\mu_1 = 0.5$ and $\mu_2 = 1$. All other parameters are kept the same as in Figure 4.2. We can see that as the variance increases the decoding latency of the multiframe model decreases. Thus, there exists a notable gain for the multiframe model in the system performance for systems with a large variance in the processing time of the servers.

CHAPTER 5

DISCUSSION

5.1 Benefits

Leveraging multiple server processing times in parallel for decoding yields significant gain in terms of the trade off between latency and FUP. Comparing the performance against the standard NFV coded scheme shows an improvement in the decoding latency of the system. The multiframe decoding scheme mitigates the redundant work done by stragglers, making the system much more efficient. This is done by allowing servers with lesser processing time to process multiple frames, collectively reducing the overall decoding time. One of the main advantage is that the fault tolerance of the network increases, making it more flexible to use COTS devices. This approach is based on a linear combination of the received frames prior to their distribution to the servers and the exploitation of the algebraic properties of linear channel codes. The simulation results demonstrate that linear coding of received frames yield trade offs between decoding latency, reliability and FUP. A specific decoding latency can be obtained by selecting a code that has a specific minimum distance and chromatic number. We also explained that the chromatic number is lower bounded by the minimum distance. Finding the chromatic number of a graph is NP-hard. However the chromatic number plays an important role in the analysis of the system. The larger the chromatic number, the higher is the probability of error. Thus the chromatic number should be small while the minimum distance of a code should by large. Increasing the minimum distance of a code increases the chromatic number. Thus for optimal results, the minimum distance is equal to the chromatic number. The multiframe NFV model works well for systems with a high variance in server processing time. We see from the simulations results that increasing the

number of servers improves the error flow of the system. The multiframe NFV model improves the system efficiency, reduces the decoding latency and increases the fault tolerance of the system.

5.2 Limitations

The multiframe decoding scheme reduces the decoding latency only if the variance in the processing times of the servers is large. If the variance is small the system performance is the same as a standard NFV model with single frame decoding as seen in the results. To evaluate the multiframe scheme in this work we assume a ginie model that performs perfect allocation of frames without repetition. The performance we see in the simulation results is the upper bound to the system performance using a multiframe scheme due to the assumption of perfect frame allocation. The multiframe decoding model requires additional administrative function by the master server to store the frames and reallocate the frames to the servers once they have completed decoding a frame. This requires additional memory and processing at the master server. The processing time of a server for each frames in the multiframe NFV model is dependent on the processing time of the previous frame at that server this makes the system complex. Since the indicator variable in the multiframe model is non binary, it is not possible to obtain tight bounds on the system performance. The probability of allocating repeated frames introduces redundant computation by the servers. The multiframe NFV model can be extended to achieve more practical results by considering the probability of repeated frames.

CHAPTER 6

CONCLUSION

In this report we propose a multiframe decoding scheme for distributed uplink channel decoding. The proposed scheme was simulated to evaluate the performance of the system based on the frame unavailability probability. The scheme was compared to a standard NFV scheme with single frame decoding and showed significant improvement in the overall performance of the system. The simulation results illustrate considerable improvement in decoding latency. The proposed model also increases the fault tolerance of the system and improves the straggler tolerance. An upper bound can be derived for the proposed scheme. However, since the indicator variable is non binary and dependent on the processing time of the previous frame, a tight bound might not be possible for this scheme. The multiframe decoding model can also be extended to other channels. In this work the model applies to additive noise channels in which the user code is an additive group. An open issue is finding the optimal trade off between the variance in processing time and the decoding latency. The multiframe scheme performs better for a larger variance in the processing time.

REFERENCES

[1] Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Taieb Znati. Network function virtualization in 5G. *IEEE Communications Magazine*, 54(4):84–91, 2016.

[2] Ali Al-Shuwaili, Osvaldo Simeone, Joerg Kliewer, and Petar Popovski. Coded network function virtualization: Fault tolerance via in-network coding. *IEEE Wireless Communications Letters*, 5(6):644–647, 2016.

[3] Malihe Aliasgari, Jörg Kliewer, and Osvaldo Simeone. Coded computation against processing delays for virtualized cloud-based channel decoding. *IEEE Transactions on Communications*, 67(1):28–38, 2019.

[4] Islam Alyafawi, Eryk Schiller, Torsten Braun, Desislava Dimitrova, Andre Gomes, and Navid Nikaein. Critical issues of centralized and cloudified LTE-FDD radio access networks. In *2015 IEEE International Conference on Communications (ICC)*, pages 5523–5528. IEEE, 2015.

[5] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[6] I Chih-Lin, Jinri Huang, Ran Duan, Chunfeng Cui, Jesse Xiaogen Jiang, and Lei Li. Recent progress on C-RAN centralization and cloudification. *IEEE Access*, 2:1030–1039, 2014.

[7] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[8] Anindya B Das and Aditya Ramamoorthy. C3LES: Codes for coded computation that leverage stragglers. *arXiv*, 2018.

[9] Marios D Dikaiakos, Dimitrios Katsaros, Pankaj Mehra, George Pallis, and Athena Vakali. Cloud computing: Distributed internet computing for IT and scientific research. *IEEE Internet computing*, 13(5):10–13, 2009.

[10] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: issues and challenges. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 27–33. Ieee, 2010.

[11] Nuwan S Ferdinand and Stark C Draper. Anytime coding for distributed computation. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 954–960. IEEE, 2016.

[12] Christina Fragouli, Emina Soljanin, et al. Network coding fundamentals. *Foundations and Trends® in Networking*, 2(1):1–133, 2007.

[13] Ken Gray and Thomas D Nadeau. *Network function virtualization*. Morgan Kaufmann, 2016.

[14] Mesud Hadzialic, Branko Dosenovic, Merim Dzaferagic, and Jasmin Musovic. Cloud-RAN: Innovative radio access network architecture. In *Proceedings ELMAR-2013*, pages 115–120. IEEE, 2013.

[15] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[16] Yashpalsinh Jadeja and Kirit Modi. Cloud computing-concepts, architecture and challenges. In *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, pages 877–880. IEEE, 2012.

[17] Svante Janson. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24(3):234–248, 2004.

[18] Anthony Josep, Randy Katz, Andy Konwinski, Lee Gunho, David Patterson, and Ariel Rabkin. A view of cloud computing. *Communications of the ACM*, 53(4), 2010.

[19] Frank R Kschischang, Brendan J Frey, Hans-Andrea Loeliger, et al. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[20] Songze Li, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. A unified coding framework for distributed computing with straggling servers. In *2016 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2016.

[21] Songze Li, Mohammad Ali Maddah-Ali, Qian Yu, and A Salman Avestimehr. A fundamental tradeoff between computation and communication in distributed computing. *IEEE Transactions on Information Theory*, 64(1):109–128, 2018.

[22] Gianluigi Liva, Shumei Song, Lan Lan, Yifei Zhang, Shu Lin, and William E Ryan. Design of LDPC codes: A survey and new results. 2006.

[23] Ankur Mallick, Malhar Chaudhari, and Gauri Joshi. Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication. *arXiv preprint arXiv:1804.10331*, 2018.

[24] John McCarthy. *Programs with common sense*. RLE and MIT computation center, 1960.

[25] Robert McEliece. *The theory of information and coding*, volume 3. Cambridge University Press, 2002.

[26] Peter Mell, Tim Grance, et al. The NIST definition of cloud computing. 2011.

[27] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016.

[28] Seok-Hwan Park, Osvaldo Simeone, Onur Sahin, and Shlomo Shamai. Robust and efficient distributed compression for cloud radio access networks. *IEEE Transactions on Vehicular Technology*, 62(2):692–703, 2013.

[29] Mugen Peng, Chonggang Wang, Vincent Lau, and H Vincent Poor. Fronthaul-constrained cloud radio access networks: Insights and challenges. *arXiv preprint arXiv:1503.01187*, 2015.

[30] Tony QS Quek and Wei Yu. *Cloud radio access networks: Principles, technologies, and applications*. Cambridge University Press, 2017.

[31] Tara Salman. Cloud RAN: Basics, advances and challenges. *A Survey of C-RAN Basics, Virtualization, Resource Allocation, and Challenges*, 2016.

[32] Puripong Suthisopapan, Mongkol Kupimai, Rangsan Tongta, and Virasit Imtawil. Design of high-rate LDGM codes. In *2009 Fourth International Conference on Communications and Networking in China*, pages 1–4. IEEE, 2009.

[33] Jun Wu, Zhifeng Zhang, Yu Hong, and Yonggang Wen. Cloud radio access network (C-RAN): A primer. *IEEE Network*, 29(1):35–41, 2015.

[34] Raymond W Yeung. *Information theory and network coding*. Springer Science & Business Media, 2008.

[35] Qian Yu, Mohammad Ali Maddah-Ali, and A Salman Avestimehr. Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2022–2026. IEEE, 2018.

[36] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[37] Liang Zhao, Ming Li, Yasir Zaki, Andreas Timm-Giel, and Carmelita Görg. LTE virtualization: From theoretical gain to practical solution. In *Proceedings of the 23rd International Teletraffic Congress*, pages 71–78. International Teletraffic Congress, 2011.

[38] Yuhan Zhou and Wei Yu. Optimized backhaul compression for uplink cloud radio access network. *IEEE Journal on Selected Areas in Communications*, 32(6):1295–1307, 2014.

[39] ZhenBo Zhu, Parul Gupta, Qing Wang, Shivkumar Kalyanaraman, Yonghua Lin, Hubertus Franke, and Smruti Sarangi. Virtual base station pool: towards a wireless network cloud for radio access networks. In *Proceedings of the 8th ACM international conference on computing frontiers*, page 34. ACM, 2011.