Performance Analysis of Network Coding based P2P Live Video Streaming Systems

Bassel Saleh

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at
Concordia University
Montréal, Québec, Canada

September 2013

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By : **Bassel Saleh**

Entitled : **Performance Analysis of Network Coding based P2P Live Video Streaming Systems**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science - Electrical and Computer Engineering

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee :

Dr. M. Zahangir Kabir	Chair
Dr. Mustafa K. Mehmet Ali	Examiner
Dr. Amr M. Youssef	Examiner
Dr. Dongyu Qiu	Thesis Supervisor

Approved by ______________________________

Chair of Department or Graduate Program Director

____

Dean of Faculty

Date August 29, 2013

ABSTRACT

Performance Analysis of Network Coding based P2P Live Video Streaming Systems

Bassel Saleh

Peer-to-peer (P2P) video streaming is a scalable and cost-effective technology to stream video content to a large population of users and has attracted a lot of research for over a decade now. Recently, network coding has been introduced to improve the efficiency of these systems and to simplify the protocol design. There are already some successful commercial applications that utilize network coding. However, previous analytical studies of network-coding based P2P streaming systems mainly focused on fundamental properties of the system and ignored the influence of the protocol details. In this study, a unique stochastic model is developed to reveal how segments of the video stream evolve over their lifetime in the buffer before they go into playback. Different strategies for segment selection have been studied with the model and their performance has been compared. A new approximation of the probability of linear independency of coded blocks has been proposed to study the redundancy of network coding. Finally, extensive numerical results and simulations have been provided to validate our model. From these results, in-depth insights into how system parameters and segment selection strategies affect the performance of the system have been obtained.

Acknowledgements

I would like to thank my supervisor Dr. Dongyu Qiu for all the help he offered throughout the thesis process. His input and feedback have made significant impact on my work. I also really appreciate his patience, guidance, and encouragement whenever I faced a new challenge.

I am also very grateful to my family who was nothing but supportive throughout the last year in spite of all the difficulties they went through.

Contents

List of Figures

List of Tables

Chapter 1

Introduction & Literature Review

In recent years, the internet has become one of the most popular platforms for content distribution to end users with content types ranging from music, photos, books and software programs to online gaming and video and audio streaming. This was facilitated first by the increase of broadband deployment over the past decade. However, the recent years have witnessed a rapid growth of internet traffic beyond the expectations. According to Cisco's Visual Networking Index [1] released in 2012, global IP traffic has increased more than fourfold in the past 5 years, and will continue to increase threefold over the next 5 years at a compound annual growth rate (CAGR) of 23 percent from 2012 to 2017. The lion's share of the traffic type goes to video. Cisco reports that consumer Internet video traffic will be 69 percent of all consumer Internet traffic in 2017, up from 57 percent in 2012. They also add that the sum of all forms of video (TV, video on demand [VoD], Internet, and P2P) will be in the range of 80 to 90 percent of global consumer traffic by 2017.

More efficient architectures have been and are still being developed to meet the current and future bandwidth demand. Content Delivery Networks (CDN) and Peer-to-Peer (P2P) networks have emerged as possible solutions with totally different design philosophies. CDNs takes the load off the core of the network by placing the servers on the edge of the network in multiple geographic locations closer to the users. Load balancing and request routing techniques such as DNS-based routing could then be used to direct requests from users to the server that is best able to satisfy the request. CDNs provide a centralized architecture that builds on the traditional client-server model to offer high availability and performance for distributing static and streaming content to the users. However, CDNs do not fully address the challenge of explosive content growth in a cost-effective manner especially for large-scale user populations.

In contrast, P2P networks employ different design principles and promise high scalability at low cost. The scalability is achieved by utilizing user resources such as the computing power, storage space, or network bandwidth to help other users on the network. This essentially alleviates the load on the server and turns the users into contributors and not just receivers. As more users join the network in this case, more resources will become available without additional cost for the dedicated server(s). In the next section, we describe the major milestones in the evolution of P2P systems from file sharing to video streaming.

1.1 P2P Networks

P2P networks are a type of distributed systems that are decentralized, self-organized, scalable and symmetric. Users in P2P networks, called peers, organize themselves in application-layer topologies called overlays so that each peer connects to and maintains a limited number of other peers called its neighbors. Based on the type of the overlay topology constructed such as a mesh or tree, P2P networks are classified into structured and unstructured networks. We discuss this more later in this chapter. All peers are symmetric in the sense that all perform the same functions and there is no special peers such as dedicated servers in the overlay. This, in addition to removing the single points of failure, allows peers to depart gracefully or due to failure without taking the whole network down. Moreover, since peers contribute their resources to help each other achieve a common objective, more resources will be available as more peers join the overlay which enables the network to scale to millions of peers without generally incurring additional cost.

1.1.1 P2P file sharing

Napster was the first successful attempt at realizing the P2P concept that triggered the academic interest in P2P networks. Napster [2] was released in 1999 aiming to facilitate sharing of mp3 music files among users. Napster was not a pure P2P network and could be instead described as a hybrid network. Files are stored on the peers' local disks instead of central server(s). However, Napster used a central directory to keep track of where each file is hosted in the network. Peers would connect to the computer hosting the central directory to search for available files and obtain the address of their hosting peer. Once that is done, the file could be transferred directly between the peers. Napster was marred by copyrights issues [3] and it was argued that maintaining a central directory could enable the operators to detect distribution of pirated materials and take them off the network. Due to failure of Napster to deal with copyright infringement issues, it was shut down in 2001 by a court order [2].
To address the single point of failure in Napster, Gnutella [4] was developed in the 2000. Gnutella is basically a search and discovery protocol that allows peers to organize in an unstructured overlay topology [5]. There is no central directory in Gnutella, instead peers issue search requests that are flooded to their neighbors. The neighbors would continue to forward the requests to their neighbors in a way similar to expanded ring. The radius of the ring would usually be between 4 to 7 levels. Neighbors that have the content matching the search parameters would reply with a request hit message. The gnutella network has survived till today and gone through some changes to

its architecture such as the the introduction of ultrapeers. In addition, many studies were conducted to enhance the efficiency of its search mechanism, and to understand how its topology and peering algorithm affect its performance.

The massive popular success of Napster and Gnutella have sparked the interest of the research community in P2P networks. The early focus was on distributed search algorithm for content lookup. This led to the introduction of the structured overlays and the concept of distributed index. In this concept, the index is not stored in a central machine but rather distributed on the peers using a technique called Distributed Hash Tables (DHTs).

DHTs provide a mechanism for distributed index storage and efficient content lookup. They allow peers to search for the content and obtain deterministic results such that a search query would certainly return a result if the content exists in the overlay which is not guaranteed by the flooding search in Gnutella.

Structured Overlays

The overlay topology is structured when the geometry of the topology, .i.e the way the peers are connected to each other, is governed by some rules. In other words, the neighbors each peer maintains are selected and maintained according to rules and procedures dictated by the design of the DHT scheme. This is in contrast to unstructured topologies where the neighbors of a peer are chosen randomly from the set of all peers in the overlay.

The basic idea of DHTs is to distribute the responsibility of storing the data index among all the peers in the overlay so that each peer stores a partition of the index. The data index usually consists of entries in the form of $< key, value >$ pairs where the key could be a file name or meta data and the value could be the IP address and port of the peer hosting the file. A hash function would then take a key k as an input to generate a hash that corresponds to a unique element in a space that could be a coordinate virtual space as in CAN [6] or a number in a bit space as in Chord [7] depending on the design. The DHT design defines what subspace each peer is responsible for storing the keys hashed in it. Neighbors to a peer would be other peers that are responsible for the *adjacent* or *close* subspaces with some notion of closeness defined in the design. Furthermore, DHT designs perform a basic set of operations such as insertion, deletion, or lookup of a $< key, value >$ pair with procedures to construct and maintain the overlay as peers join and depart all without any sort of central control. The main operation though is the lookup where a lookup request is routed through neighbors so that it gets closer and closer to the target node. There are many different designs of DHTs such as CAN [6], which was the first proposed design of a DHT that defines the key space as a d-dimensional virtual cartesian coordinate space with the notion of closeness defined as the cartesian distance. Other well-known designs are Chord [7] which organizes the key space as a ring and Kademlia [8] in which the key space is organized as a binary tree with the distance defined by the XOR operation.

1.1.2 P2P video streaming

Peer-to-peer video streaming technology was introduced almost a decode ago following the popular success of p2p file sharing systems. Contrary to traditional server-client streaming systems, it takes the load off the server(s) by allowing clients to contribute their upload bandwidth to assist each other. By doing so, the system can scale to a large number of users at a low cost for the server bandwidth.

Since its inception, P2P video streaming has attracted a lot of attention from both the research community and the industry. However, research efforts before the introduction of P2P streaming formulated the problem of disseminating the video stream from the server to the clients as a multicast problem with a source located at the server and multiple receivers represented by the clients. Earlier architectures proposed to optimize the data dissemination by building an IP multicast tree on the network layer [9]. These kind of architectures require changes in the internet routers and switches which limited its deployment. Moreover, IP multicast did not scale well and lacked support for higher layers functionality.

To address these problems, it has been suggested to implement the multicast functionality in the application layer of the end hosts. This would allow more flexibility in the design because no changes to the internet infrastructure are required. Application-layer multicast was first proposed in [10] and reported to offer acceptable delay with limited bandwidth penalty compared to IP multicast. Currently a whole family of streaming protocols exist that employ a tree topology so that they could be classified into their own category. Subsequent designs of the P2P streaming protocols opted for a random mesh topology because of its resilience against peer dynamics and ease of implementation. Next we are going to discuss the main concepts and advantages and disadvantages of each topology which led to the creation of hybrid topologies and then the introduction of network coding to video streaming.

1.1.3 Tree-based Approach

In this approach, peers would organize themselves in an application-layer multicast tree with the source of the stream placed at the root of the tree. Each peer would then receive the stream from its parent and forward it (copy it) to its children. One of the first designs of an overlay multicast system was End System Multicast (EMS) [10]. The protocol of EMS called Narada functions in the application layer of end hosts to form small-scale multicast groups with the ability of having multiple sources geared towards video-conferencing applications. Later Overcast [11] was introduced to support large-scale single-source multicast groups formed using an overlay network. Overcast builds a bandwidth-efficient application layer mutlicast tree by letting peers join near the root, then moving them down the tree subject to the bandwidth available.

First designs like Overcast used a single tree for each stream which has a major drawback. Peers that have no children i.e. are leaves in the tree would not contribute their upload bandwidth which is nor efficient neither fair to other internal peers in the overlay. One other drawback of single trees is the sensitivity to peer churn. Peer churn occurs when peers fail or depart the overlay, which would cause the whole subtree rooted at the departing peer to be cut off from rest of the tree. Consequently, every time a peer departs, the tree structure has to be repaired which results in a higher maintenance overhead and longer delays. To utilize the bandwidth of leaf peers and improve the resiliency to peer churn, multiple trees for a single stream were introduced in 2003 [12]. In [12] a ”forest” is created out of multiple interior-node-disjoint multicast trees. The multiple trees are formed such that most of the leaf peers in one tree are interior peers in another tree to solve the problem of bandwidth efficiency of leaf peers. Furthermore, the video stream is divided into multiple substreams or stripes such that every substream would be distributed over a different tree. Therefore, a peer would be part of multiple trees and receive a different substream of the video from its parent in each tree. This design is depicted in Figure 1.1

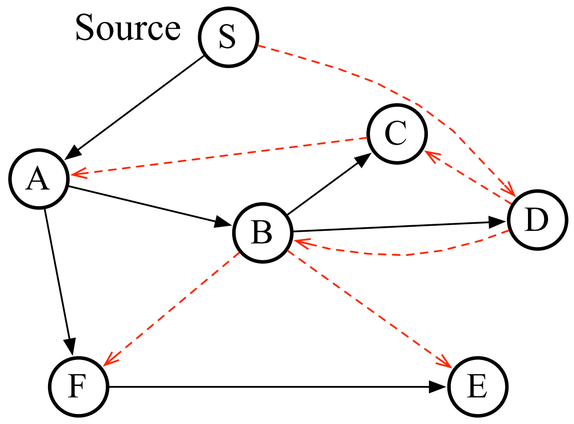


Figure 1.1: Application-layer multiple multicast trees [13]

Since peers in the multicast tree can push the video stream to their children without worrying about children receiving redundant data or having to receive requests from the children, the delay experienced by the peers might be low generally. However, the lack of robustness to peer dynamics and the resulting overhead to maintain the tree structure may significantly hurt the performance. This made subsequent designs favor an easier-to-implement random mesh topology that we will discuss next.

1.1.4 Random mesh based streaming

Peers in this approach organize themselves in an unstructured topology such that each peer connects to a number of neighbors randomly selected from the peers in the overlay, hence the random mesh topology. This makes the topology more robust and resilient to peer churn. If a few neighbors depart or fail, the peer could still communicate to other functioning neighbors. Different approaches could be used to construct the mesh. One example could be using a gossip-like protocol by which peers exchange neighbor information with each other and then a peer would select some of the peers it has received and connect to them. Another approach is to use a service called the tracker which maintains a list of all the peers in the overlay. When a peer first joins the overlay, it registers with the tracker and obtains a subset of the peers in the overlay that it uses as neighbors. The mechanism of a tracker is borrowed from the BitTorrent file sharing protocol that gained massive popularity after a short period of its release. In fact, the success and simplicity of BitTorrent led to studies that investigated if this success could be replicated for P2P video streaming. Shortly after that, new P2P live video streaming designs emerged such as Chainsaw [14] and CoolStreaming [15] with design clues inspired by BitTorrent. CoolStreaming was the first experiment that demonstrated the practicality of implementing P2P live video streaming in a real-world setting.

Although the general design principles of these protocols bear a lot of similarities to BitTorrent, they also cater for the rigorous timing requirements of live streaming. The delay from the source to a peer has to be kept at minimum such that video data has to be received in time before their playback deadline. If some video data arrived late at a peer, they will be discarded and the user experience will suffer. Just as in BitTorrent where a file is divided into pieces, the video stream is divided into chunks of data of the same size called segments as shown in Figure 1.2. The segment duration

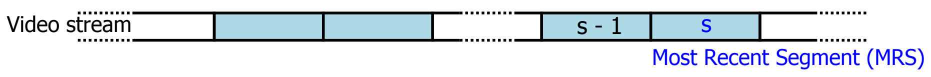


Figure 1.2: The division of the video stream into segments

or playback time could be one to few seconds e.g. one second in CoolStreaming. Peers would usually have buffers to store the segments before playback and are also used as caches to serve other peers. The buffer length in CoolStreaming is 120 segments which amounts to 2 minutes. Each peer exchanges with its neighbors periodically buffer maps messages which contain information about what segments are currently available in its buffer. This enables the peer to determine what neighbors currently have the segments that are missing from its buffer. The peer can then send requests to its neighbors to retrieve its missing segments. The details of which missing segments to download from which peers are dictated by what is called the *scheduling algorithm*. The peer also receive requests from its neighbors and works on satisfying them in a way that saturates its upload bandwidth.

If a segment is still missing at the time of its playback, a peer in a live streaming session has two options to deal with this situation: *i*) either skips the segment and causes an interruption or *discontinuity* in the video playback until the next available segment is up for playback, or *ii*) give the segment some time and wait for it to get downloaded. This would cause the video playback to pause for a little time but if the segment gets downloaded the user would not miss the equivalent part of the video. However, peers have to maintain a maximum delay between their current playback time and the time a live event occurs on the server.

The mechanism of sending requests to download missing segments is known as the *Pull-based* approach as opposed to the *Push-based* approach used on the tree topologies. The pull-based method has been found to cause delays in retrieving the segments which is not desirable in streaming applications. This has led to investigating whether the benefits of a mesh topology could be combined with the advantages of the Push mechanism to get the best of both worlds. From research on this matter, hybrid approaches has emerged.

In the hybrid approach, the neighbors of a peer would act as parents and push video segments to the peer. To avoid sending redundant data and be more robust against peer dynamics, the video stream would be divided into multiple substreams. For example, if the stream is divided into K substreams in total, then the i^{th} substream would include segments with IDs $i + nK$ for $n = 0, 1, 2, \ldots$. A peer would then subscribe to K of its neighbors to have each push a different substream. In addition, *Pull* may still be used to retrieve lost segments. The scheduling algorithm that controls which neighbors *Pull-based* to subscribe to for which substreams could be designed to minimize the delay.

Although hybrid designs may actually reduce the source-to-peer delay, they introduced more complexity in the scheduling algorithm. Moreover, they are still not quite resilient to peer dynamics. For instance, if a neighbor that is pushing a substream to a peer fails, the peer has to switch to a different neighbor or even re-run the scheduling algorithm again which may result in the loss of a few segments.

Fortunately, the introduction of network coding to P2P live video streaming around 2006 offered a much more simplified design while maximizing the positives of the push approach. In the next section, we introduce network coding and give an overview of how it works withing the context of live P2P video streaming.

1.2 Network Coding based P2P streaming

Network coding was originally proposed in information theory [16, 17] as an approach to achieve max-flow min-cut capacity in multicast sessions over certain network topologies. In normal networks, the nodes use the *store and forward* approach to forward received packets out their output links. In this approach, the messages would just be copied from the input links to any subset of the output links. In contrast, *network coding* allows intermediate notes in the network to perform operations on, or i.e. encode, the incoming data before forwarding it. Later it was shown that the linear coding scheme is sufficient to achieve the max-flow from the sender to the receiver [18]. With linear coding, a node can encode received packets by simply forming a linear combination of them to produce a coded packet. Later it was proposed in [19] that the coding coefficients used to form the linear combination of incoming packets could be chosen randomly from a finite field of size q. Moreover, the coding efficients could be carried with coded packet itself. This would enable a receiving node to decode the original packets after it receives enough number of coded packets with their coefficients.

Therefore, it has been established that network coding is resilient to random packet loss and delay and to variations in network capacity and topology, and, furthermore, it can utilize the bandwidth efficiently to achieve near-optimal performance.

This prompted research on the practicality of replicating the theoretical results of random linear network coding in P2P content distribution systems. The benefits of network coding in P2P BitTorrent-like content distribution systems was first investigated in [20]. The argument here is that if peers encode all the blocks they have received so far of a file and serve coded blocks to other peers, the download time of a large file could be reduced. It has been found in [20] through simulations that the download time could be improved by 2-3 times compared to traditional approaches without network coding. However, since network coding requires more computation complexity for the encoding and decoding processes than the simple store and forward approach, it was of practical importance to evaluate the performance of network coding with a real implementation. In [21], a prototype was in implemented in C# of a P2P content distribution system to measure the performance of network coding. It has been reported in [21] that network coding provides fast and smooth downloads with little overhead in terms of CPU cycles and I/O activity. In addition, since all coded pieces are equally useful, there is no need to distribute the rarest piece of the file first which would solve the ”end-game” problem.

Once again the history repeats itself, the promising results of network coding in P2P file sharing encouraged researchers to find out whether the potential benefits of network coding could be harnessed in the context of live P2P video streaming systems that has strict timing requirements.

The first systematic attempt to explore the practicality of network coding in live streaming setting was done in [22] where an experimental testbed called Lava was built to test the performance of network coding against traditional pull-based streaming. It has been found that network coding offers a better performance especially when the bandwidth supply barely exceeds the demand while incurring a low computation overhead.

With these reported performance gains, they believed in [23] that in order to take full advantage of the network coding, a complete redesign of the streaming protocol was needed. Therefore, R^2 was introduced as the first P2P network-coding based live streaming protocol [23].

After that, the first production deployment of a p2p video streaming system designed from the scratch with network coding in mind was launched by UUSee and reported to offer superior real-world perforamce during Summer Olympics in China in 2008 [24]. Next we will state the most important design principles of R^2.

1.2.1 Design principles with network coding

Each segment in the video stream in network coding is divided into a number of blocks, let it be m, of the same size. The segment duration of playback time is typically between one and a few seconds (e.g. 4 seconds), and the the number of blocks in live streaming is roughly between 50-200 blocks.
To reduce the computation complexity of network coding, chunked codes were introduced in [25]. The idea has partly inspired the design of R^2 which states that coding should occur only within the boundaries of a segment and independently for each segment as opposed to encoding the blocks through multiple segments. Random linear coding is used to form a linear combination of the blocks the segment has so far and generate a coded block with the coefficients chosen uniformly at random and independently from the Galois finite field of size q i.e. $\mathcal{GF}(q)$. The coefficients are embedded with the coded block and sent to the neighbors. Therefore, in a sense, a peer serves a linear equation to its neighbors with the unknowns being the original blocks of the segment.
Once a peer receives m linearly independent equations (i.e. coded blocks) for a segment, it can recover the original blocks of the segment by solving the system of the received linear equations. However, the decoding process could be performed progressively as each coded block is received using Gauss-Jordan elimination method and thus save time. If a received coded block is tested to be linearly dependent with the existing blocks of a segment, it would be discarded.

Contrary to the hybrid models we discussed earlier, network coding simplifies the design and allows for the Push approach to be used more easily and be much more robust to peer dynamics. In $R2$, peers push coded blocks to their neighbors without having to receive explicit requests. Moreover, since all the coded blocks of a segment are equally useful, the neighbors of a downstream peer can push coded blocks to the same segment on that peer without worrying about redundancy and without having

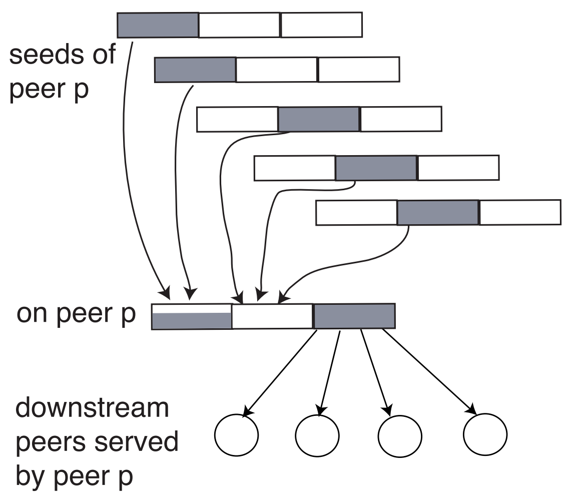


Figure 1.3: Multiple neighbors (seeds) collaborate to serve segments on a peer p without coordination. reprinted from [23]

to coordinate among each other. Consequently, neighbors can collaborate to serve a segment on a downstream peer without exchanging any protocol messages as shown in Figure 1.3. This makes the protocol much more robust to peer churn since loosing a few neighbors would not stop downloading a segment completely as other neighbors would continue to serve coded blocks to the segment.

In order for the push approach to work properly, peers have to know what segments are still not complete in the buffers of their downstream neighbors. Otherwise, they would risk sending coded blocks to segments that had already received enough linearly independent blocks and gotten decoded. Therefore, each peer sends a buffer map message to all of its neighbors that contains whether each segment in the buffer is complete or not. In addition, buffer maps updates are sent immediately every time the state of the buffer changes, or in other words, every time a segment gets complete or gets played out of the buffer. Furthermore, buffer maps do not include the number of blocks each segment currently has which has many upsides. This would keep the design simpler and reduce the size of the messages as well as their sending frequency. It is also worth mentioning that network coding uses shorter buffer lengths with larger segments which would also contribute to reducing the size of buffer map messages.

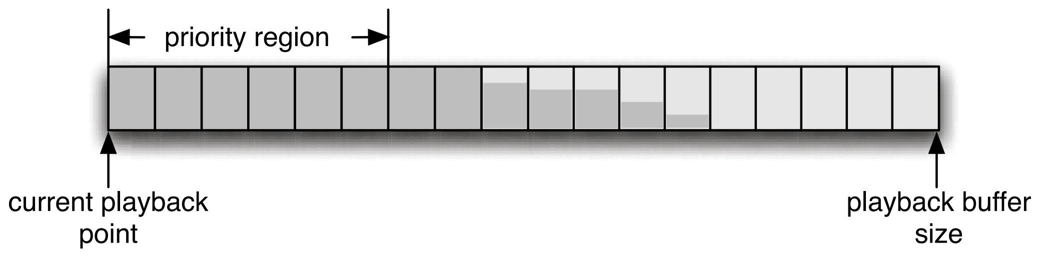


Figure 1.4: Playback buffer organization in R^2. reprinted from [23]

Therefore, the overhead of buffer maps would be very low compared to traditional Pull-based streaming.

Finally, peers decide which segments on which peers to serve coded blocks to on a random basis, hence the name R^2. Whenever a peer can push a coded block, it consults the buffer maps received from its neighbors and randomly chooses one of its neighbors that has some missing segments. Then it randomly chooses a missing segment on that neighbor. R^2 divides the buffer into two regions and calls the one that has segments closer to their playback deadlines the priority region as shown in Figure 1.4. Missing segments in the priority region take priority over segments in the other region and are chosen with uniform distribution. Missing segments in the other region have the chance to receive blocks only when the priority region is full and could be sampled with any distribution that favors segments closer to their playback deadline.

1.3 Related Work

Significant amount of research was done in the field of P2P live video streaming. After introducing network coding as an information theoretic approach that can achieve min-cut capacity in multicast sessions on certain network topologies [16] [17] [18] [19], studies were conducted to investigate the feasibility of using network coding to improve performance in p2p content distribution [20, 21] and p2p video streaming systems [22, 23]. Network coding was found to improve downloading time in p2p file sharing and also get rid of the "rarest piece" problem. In the context of live video streaming, it was shown that it is practical to implement network coding within P2P live streaming systems and it offers significant improvements of the performance compared to traditional streaming [22]. After that, the first production deployment of a p2p video streaming system designed from the scratch with network coding in mind was launched by UUSee and reported to offer superior real-world perforamce during Summer Olympics in China in 2008 [24].

In the early P2P live streaming systems, peers used the pull approach to send requests for missing segments to neighbors. While pull does not suffer from the redundancy problem, it led to a large delay in fetching the segments and was less resilient to neighbor departures or failures. Hybrid Pull/Push was introduced later to reduce the delay but led to more complex scheduling. In the hybrid design the video stream would be divided into several substreams. A peer then would subscribe to different neighbors that would act as parents and push different substreams to the peer. On the other hand, network coding allows push scheduling to be used in a much simpler way. This is due to the fact that coded blocks of the same segment from any peer are equally useful, which allows multiple neighbors to cooperate and serve coded blocks to the same segment on a peer without worrying about the redundancy and without exchanging any messages to coordinate among each other.

The push approach for network coding along with other design principles was introduced in R^2 in [23] which is a complete redesign of the streaming protocol to take full advantage of network coding. Following that, a mathematical study to model these design principles was presented in [26] to understand the fundamental properties of the system and identify sufficient condition for perfect playback. However, the influence of the protocol details or other design options was ignored. Other studies as in [27] that investigated the influence of the system parameters focused on a limited set of them such as the block size and aggressiveness only and did not provide much insights into their influence on the system. Therefore, we believe that there exists a need to reveal what effects some design options like the segment selection strategies may have on the performance. In addition, we think more in-depth insights into the effect of the system parameters are needed in order to understand the unique nature of network coding.

1.4 Thesis Objective

The aim of this thesis is to study the performance of push-based network coding P2P live video streaming systems and obtain important performance metrics such as the probability of continuity and efficiency. Our purpose is to investigate the influence of the system parameters and design options on the performance. To that end, we develop a unique stochastic model that reveals how the number of blocks in a segment evolve over the time the segment spends in the buffer. Through the model, we provide analysis for the most urgent and uniform segment selection strategies and compare their performance which has not been done before in the literature to the best of our knowledge.

We pick only the main design principles from $R2$ to analyze in our study such as the Push approach. We do not model the relatively complex segment selection strategy used by R^2 in which the buffer is divided into two regions. Instead, we opt for simpler design options for the segment selection strategy that treat the buffer as one region.

Unfortunately, the benefits of network coding comes with a price. A peer may receive coded blocks that may be linearly dependent with the existing blocks and thus would provide no useful information and would be discarded. Linearly dependent blocks lead to waste of bandwidth and thus it would be of great interest to study their effect on the efficiency of the protocol. Previous research [26, 28] provided only a relatively loose lower bound of the probability that a received coded block is linearly independent. We propose a much better approximation of this probability and investigate the toll it takes on the performance. Finally, we do simulations to verify the model results.

1.5 Thesis Organization

The rest of this thesis is organized as follows:

In **chapter 2**, we start by presenting a system overview that provides a high-level view of how the system works as well as the specifics of how network coding is performed. After that, we present a description of the system model along with system parameters and notations. Then, we start our analysis of the protocol. We first treat the case of homogeneous peer upload bandwidth and obtain the distribution of the number of blocks the segment accumulates throughout its stay in the buffer for both the most urgent and uniform segment selection strategies. Next, we relax the assumption of homogeneous bandwidth and investigate the case of heterogeneous peer upload bandwidth. We conclude the chapter by presenting the analysis for the efficiency under homogeneous and heterogeneous upload capacity.

In **chapter 3**, we present the numerical and simulation results. Next we investigate the effect of the bandwidth supply from the peers and the server on the performance of both strategies. We also provide a comparison of both the uniform and most urgent strategies and explain why one performs better than the other. Then we study the effect of most of the system parameters related both to video streaming and network coding. We also provide in-depth insights into the influence of each of the parameters on the performance of both strategies.

In **chapter 4**, we summarize our work and provide highlights of our findings. We then conclude by stating possible improvements and additional work the could be done to extend our model.

Chapter 2

System Model and Analysis

2.1 System overview

The peers form an application layer overlay network with mesh topology. Unlike [26] where they assume a fully connected mesh (graph), we study the case where each peer connects to only a subset of the overlay consisting of H randomly selected peers called the neighbors. This allows us to investigate the effect of a limited neighborhood on the performance. There is only one source of the video in the overlay and we call it the server. We elaborate on how the server works later in this section.

2.1.1 How Network Coding is Performed

Just as in traditional p2p streaming systems, the video stream is divided into small chunks of data of equal length called *segments*. The segment duration of playback time, denoted as S, could be about a few seconds, typically 1 to 4 seconds. However, in network coding, each segment is further subdivided into m blocks $[b_1, b_2, \ldots, b_m]$ of the same size B_s. Moreover, peers do not exchange raw segments but instead encode the segments before uploading them to their neighbors. The coding is performed on the blocks within the boundaries of a segment and separately for each segment to generate a coded block as opposed to coding for multiple segments. This is to reduce the computation complexity of network coding.

Encoding

From a sending perspective, to generate a coded block x for a segment, a peer calculates a linear combination of the original blocks of the segment $[b_1, b_2, \ldots, b_m]$ as follows:

$$x = c_1 b_1 + c_2 b_2 + \ldots + c_m b_m \tag{2.1}$$

where each of the coefficients c_i's is chosen independently and uniformly at random from the characteristic-two Galois field of size $q = 2^d$ i.e. $\mathcal{GF}(2^d)$. The exponent d is a positive integer that represents the number of bits required to store each coefficient. Furthermore, the addition and multiplication operations in equation (2.1) are all done over the characteristic-two finite field $\mathcal{GF}(q = 2^d)$ which makes the coded block x always have the same size as the original blocks. Moreover, to perform the operations over $\mathcal{GF}(2^d)$, the video blocks b_i's are treated as vectors over $\mathcal{GF}(2^d)$ such that each consecutive d bits in a block b_i is interpreted as a symbol $b_{i,j}$ over $\mathcal{GF}(2^d)$ [29]. Therefore, the summation has to occur for every symbol position to generate the corresponding coded symbol x_j i.e. $x_j = \sum_i c_i b_{i,j}$. The coded block x would then be given as:

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_g \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \cdots & c_m \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,g} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,g} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m,1} & b_{m,2} & \cdots & b_{m,g} \end{bmatrix} \tag{2.2}$$

where g is the number of symbols in a block.

To serve a segment to a neighbor, the peer sends a coded block x of the segment along with the coefficients vector $[c_1\, c_2\, \cdots\, c_m]$ to the neighbor. Consequently, in a sense, the peer is sending a linear equation to the neighbor with the unknowns being the original blocks of the segment. This makes each pushed coded block incur a **communication overhead** resulting from the carried coefficients. We denote by O the size of the coefficient overhead and study its effect later.

In network coding, coded blocks could be generated from other coded blocks. We mean by this that a peer can serve coded blocks from segments that are not decoded yet. This can be done be re-coding the existing blocks of a segment to generate a new coded blocks. We denote by (**a**) the minimum number of coded blocks a segment should have before a peer can start to serve coded blocks from it to neighbors. a is known as the **aggressiveness** parameter because it indicates how aggressive a peer is to start serving a segment. A trade-off associated with a exists between how likely a pushed block is linearly independent and the delay before starting to serve a segment. We discuss this trade-off in chapter 3. We call segments with at least a coded blocks as *Ready-to-be-Served (RtbS).*

When a segment has $a \leq f < m$ coded blocks $x^1, \ldots, x^f$ with corresponding m-dim. coefficients vectors $\boldsymbol{c^1}, \ldots, \boldsymbol{c^f}$, a coded block x' could be generated by randomly and independently choosing f coefficients $[k_1, \ldots, k_f]$ from $\mathcal{GF}(2^d)$ to calculate $x' = \sum_{i=1}^{f} k_i x^i$. However, the coefficient vector sent with x' is not $[k_1, \cdots, k_f]$, instead the equations of the coded blocks x^i are substituted into the linear combination to obtain a global coefficients vector $[c'_1, \cdots, c'_m]$ that multiplies the original blocks as follows:

$$\begin{aligned}
x' &= k_1 x^1 + \ldots + k_f x^f \\
&= k_1(c_1^1 b_1 + \ldots + c_m^1 b_m) + \ldots + k_f(c_1^f b_1 + \ldots + c_m^f b_m) \\
&= (k_1 c_1^1 + k_2 c_1^2 + \ldots + k_f c_1^f) b_1 + \ldots + (k_1 c_m^1 + k_2 c_m^2 + \ldots + k_f c_m^f) b_m \\
&= c'_1 b_1 + \ldots + c'_m b_m
\end{aligned}$$

where

$$[c'_1 \cdots c'_m] = [k_1 \cdots k_f] \begin{bmatrix} c^1_1 & \cdots & c^1_m \\ \vdots & \ddots & \vdots \\ c^f_1 & \ldots & c^f_m \end{bmatrix} \tag{2.3}$$

The resulting m-component vector $[c'_1 \cdots c'_m]$ is sent with x' to the neighbor. Therefore, coded blocks are always linear combinations of the original blocks whether they are formed from the original blocks of a complete segment or from the coded blocks of non-complete segment.

Decoding

From a receiving perspective, once a peer receives m linearly independent equations i.e. coded blocks for a segment, it can fully recover the segment by solving the following system of equations for $\mathbf{b}$:

$$\mathbf{x} = \mathbf{C}\mathbf{b}$$

$$\begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^m \end{bmatrix} = \begin{bmatrix} c^1_1 & c^1_2 & \cdots & c^1_m \\ c^2_1 & c^2_2 & \cdots & c^2_m \\ \vdots & \vdots & \ddots & \vdots \\ c^m_1 & c^m_2 & \ldots & c^m_m \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{2.4}$$

were each row of the coefficient matrix $\mathbf{C}$ is the coefficient vector received with each incoming coded block.

If a received coded block is tested to be linearly dependent with the existing blocks of a segment, it will not be innovative or i.e. will not provide any useful information and thus will be discarded.

Therefore, a segment is complete only when it has accumulated m Linearly independent coded blocks.

To solve the system of equation in (2.4), a peer can calculate the inverse of the matrix $\mathbf{C}$ using Gaussian Elimination over $\mathcal{GF}(2^d)$ and then obtain the original blocks from $\mathbf{b} = \mathbf{C^{-1}x}$. In this case, the peer has to wait until it receives all the m coded blocks to start the decoding process. A better approach to avoid the waiting is to use Gauss-Jordan method that reduces the coefficients matrix to Reduced Row Echelon Form (RREF). It enables the peer to start the decoding process progressively as each coded block is received, and it will produce a row of zeros if a received coded block is linearly dependent which removes the need for explicit dependency checks.

Implementation of Random Linear Network Coding

As we have previously mentioned, all the operations in the encoding and decoding processes are performed over the characteristic-two finite field $\mathcal{GF}(2^d)$. This field is known as a *binary extension field* and its elements are the integers from 0 to $2^d - 1$ that could be represented as a d-bit binary numbers. In addition, each element could also be represented as a polynomial with a degree of at most $d - 1$ and coefficients from the field $\mathcal{GF}(2)$. For example, the 4-bit field element $u = 0101$ is represented by the polynomial $u(x) = x^2 + 1$. The polynomial representation is used to define how the operations are performed. To add two elements, the corresponding polynomials are added with the addition of the coefficients is done modulo 2 since the coefficients belong to $\mathcal{GF}(2)$. This makes the addition equivalent to the inexpensive bitwise-XOR operation. On the other hand, the multiplication is also done as polynomial multiplication. However, since it may results in a polynomial of degree $2d - 2$ at most, the resulting polynomial is divided by an irreducible polynomial of degree d to reduce its degree to at most $d - 1$. Each field $\mathcal{GF}(2^d)$ is characterized by an irreducible polynomial of degree d with binary coefficients from $\mathcal{GF}(2)$. Irreducible means the polynomial can not be factorized into the product of two or more polynomials of degrees less than d. Such irreducible polynomial exists for each d and could be found efficiently. For example, $x^8 + x^4 + x^3 + x + 1$ is one of the irreducible polynomials for

$\mathcal{GF}(2^8)$.

Unlike the inexpensive XOR bitwise addition, the multiplication and division of polynomials is not efficient. Thus, many techniques and algorithms have been employed to speed up the multiplications such as using a pre-computed multiplication table. Since finite fields have applications in cryptography and reliable storage systems to name a few, a substantial amount of research has been conducted to increase the efficiency of its arithmetic and especially the multiplication and division operations. The research focused on developing efficient software implementations [30, 31] and hardware architectures [32] to speed up the multiplication and division operations.

As far as network coding is concerned, a parallel implementation has been introduced in [33] that utilized the SSE2 instruction set and the multiple cores in modern processors to perform the multiplication in parallel. The coding bandwidth in [33] has reached 43 MB/s for 64 blocks with 32 KB each. Later implementations of network coding [34, 35] have taken advantage of the massive parallel structures in the current video cards (GPUs). The combined CPU-GPU encoding scheme in [35] is able to achieve a coding bandwidth of up to 116 MB/s which is enough to saturate Gigabit network interfaces.

Finally, our model does not depend on how network coding is implemented. What matters in our analysis is that the addition and multiplication operations used in forming the linear combination and inverting the coefficients matrix are all done over the finite field $\mathcal{GF}(2^d)$. How these operations are implemented has no effect on our analysis.

2.1.2 Buffer organization

Each peer maintains a buffer to store video segments before they are played back. The buffer is used also as a cache from which peers serve segments to their neighbors. Buffer positions are numbered from 1 to L as shown in figure 2.1. The first position is for the most recent segment s. The second position holds segment $s-1$. The last position L holds the segment that is being currently played back $s-L+1$. Each

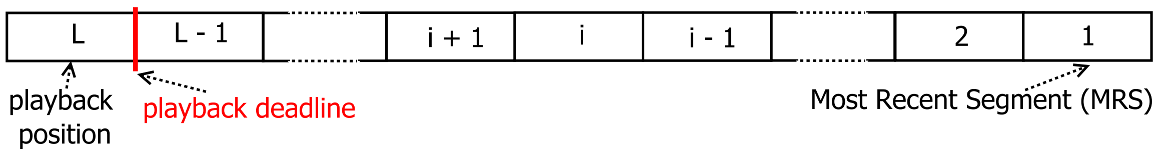


Figure 2.1: playback buffer

segment starts its life in the buffer in position 1, and then every S seconds moves to the next position. It has a chance to receive coded blocks until it reaches its playback deadline when it moves into the last position. At that point, the segment would cease to receive any blocks and would be played back only if it is complete (has accumulated m blocks). If a segment reaches the playback position and yet still not complete, it would not be given any more time and would be skipped. This would cause an interruption in the video playback for the user for the segment duration. All the peers are synchronized in the sense that all are playing the same video instant at approximately the same time. This assumption is in line with [23], the original design of the first network coding live streaming protocol R^2. The reason for this is to allow the buffers to overlap as much as possible making each segment have as many neighbors able to serve it as possible. Therefore, each peer maintains a delay of (LS) seconds from the server throughout the streaming session.

2.1.3 Server algorithm

The server could be designed in several ways. To simplify our analysis and without loss of generality we model the following design of the server. The server always distributes the most recent segment which is denoted herein as MRS. Whenever a new segment becomes available, the server chooses a subset of peers uniformly at random among all the peers in the overlay and sends at least a linearly independent coded blocks of the current MRS to each. The server decides on the number of peers so that it best utilizes its upload bandwidth and could get the list of all neighbors from the tracker that may be implemented on the server machine itself or on a separate computer.

The rationale for this design is to increase the number of peers that can serve a new segment while taking advantage of the buffering time available hoping the segment will reach more peers more quickly. However, the implication is that peers would not be able to assist each other for the first buffer position (that holds the MRS). And at the end of the current MRS duration, only peers with a blocks or more can start to serve coded blocks of the segment to their neighbors.

In fact, this kind of design is also used in [36]. The difference in our case is that we do not restrict the server to push to its neighbors only as they did in [36], instead the server in our case can push coded blocks to any peer in the network and does not maintain any neighbors.

2.1.4 Block Scheduling

As we mentioned earlier, we model the push approach. In this approach, each peer does scheduling to decide what segments on what peers it should push coded blocks to. The scheduling algorithm is run periodically and the output is either a list of specific segments on specific neighbors or nothing. The peer then sends a coded block of each selected segment to the selected neighbors or sits idle if there is no neighbor it can serve at the currently. There are many strategies that can be employed for i) neighbor selection, and ii) segment (or buffer position) selection. We choose to model the uniform strategy for neighbor selection whereby each peer selects one neighbor uniformly at random among the servable neighbors. The selection is repeated independently with replacement for each block the peer can send at the current time slot. As for segment selection, we model two design options: the Uniform selection strategy and the Most Urgent selection strategy and compare their performance.

2.2 System Model

For easier reference, we organize all the system parameters and notations in Table 2.1.

Parameter	**Description**
Z	Peer population size
U_s	Server upload bandwidth in bps
U_p	Peer upload bandwidth in bps
R_s	Streaming rate in bps
H	Number of neighbors each peer maintains
L	Buffer Length in segments
S	segment duration in *seconds*
m	Number of blocks in a segment
a	aggressiveness
q	Galois field size
T	Slot time in *seconds*
E	Number of slots per segment duration
B_s	block size in *bits* including the overhead
O	size of the coefficients overhead in *bits*

Table 2.1: system parameters and notations

The network bandwidth of a connected peer is the most important resource in P2P video streaming systems. The upload bandwidth of a peer U_p dictates the maximum data rate at which a peer can send out information to other peers and determines its contribution in the overlay. While the download bandwidth limits the data rate at which a peer can receive information from the network. In current home or enterprise grade internet connections, the download bandwidth is much higher than the upload bandwidth. Thus, we assume the download bandwidth is unlimited and the bottleneck is only the upload bandwidth.

We first treat the case of *homogeneous* peer upload bandwidth U_p where it is assumed to be constant and the same on all peers. We will relax this assumption later and treat the case of *heterogeneous* upload bandwidth where the upload bandwidth is a random variable and peers may have different upload speeds.

The time is slotted and the slot duration T is equal to the time required to upload one block. Therefore, peers can only upload one coded block per time slot in the homogeneous case. Thus T is then given by:

$$T = \frac{B_s}{U_p}, \quad \text{seconds} \tag{2.5}$$

where B_s is the the size of the coded block and could be obtained as follows. We know each coded block carries m coefficients in addition to the value of the block itself. We call this extra data the coefficients overhead and can be obtained as:

$$O = m \log_2 q, \quad \text{bits} \tag{2.6}$$

where each coefficient requires $\log_2 q$ bits. The block size B_s could then be obtained from:

$$B_s = \frac{SR_s}{m} + O = \frac{SR_s}{m} + m \log_2 q \tag{2.7}$$

By substituting eq. (2.7) into (2.5), we obtain the slot duration.

Definition *Segment Age*: The amount of time a segment has spent so far in the buffer which is measured from the time it has become available on the server to the current time instant. The *segment lifetime* then is the total age of the segment in the buffer before it enters the playback position.

Therefore, the *segment lifetime* is equal to $(L-1)S$. We distinguish between two parts of the segment lifetime as shown in Figure 2.2: a) the first S seconds where the segment would be in the first position on all peers and would be servable only by the server, and b) the rest of the lifetime $(L-2)S$ spent in positions 2 to $L-1$ where it would be servable only by the peers that can assist each other to complete the segment. We divide this part of the segment lifetime that is servable by the peers into N slots of time equal to our system's slot time T, and we call them *age slots*

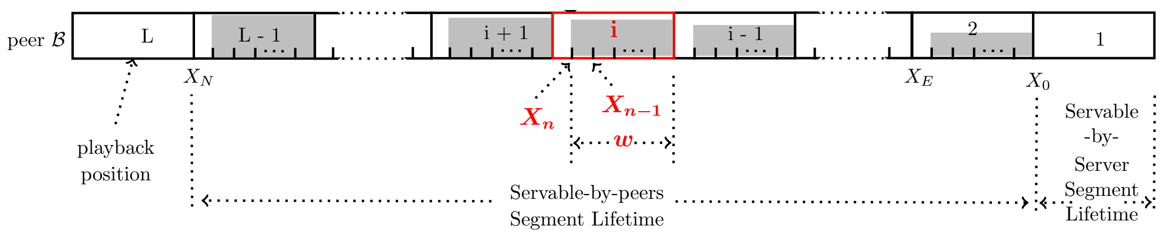


Figure 2.2: Segment lifetime and age slots

with numbers $n = 0, \ldots, N-1$ as shown in Figure 2.2. Thus, we have:

$$N = \frac{(L-2) \times S}{T} = (L-2) \times E \tag{2.8}$$

where we define E as the number of age slots in a segment duration, or i.e. in the period a segment spends in one buffer position. Thus:

$$E = \frac{S}{T} = \frac{SU_p}{B_s} \tag{2.9}$$

The segment age n indicates its current buffer position. For example, at age $n = 0$ the segment has just entered the second position and would be at the end of its stay in this position at $n = E-1$ where it has spent almost S seconds. At the next $n = E$ the segment would be at the beginning of the third position as shown in Figure 2.2. In general, the segment will be in buffer position i between age slots $n = (i-1)E$ and $iE - 1$ where $i = 1, \ldots, L$.

Let $s_n^{\mathcal{B}}$ denote the segment of age n on some peer $\mathcal{B}$, and let it be in buffer position i at time slot t. The position of $s_n^{\mathcal{B}}$ is then given by:

$$i = \left\lceil \frac{n+1}{E} \right\rceil + 1 \tag{2.10}$$

where the segment spends E age slots in each position and the $+1$ to account for the first position.

At any time slot t, there are L segments in the buffer of each peer where each segment has spent the same time in its current position as shown in Figure 2.2. We denote the number of age slots each segment has spent in its current position at time slot t as w. Thus $w \in \{0, \ldots, E-1\}$. w could be obtained from the age of the segment such that if we know that one of the segments in the buffer is at age n^i at time slot t corresponding to position i, then w could be given as:

$$w = n^i \bmod E \tag{2.11}$$

The ages of the segments in the other positions $k = 1, \ldots, i-1, i+1, \ldots, L$ could be obtained from w:

$$n^k = w + (k-1)E \tag{2.12}$$

where the difference between the ages of two consecutive segments is E age slots.

At the next time slot $t+1$, the segment in position i will move to age slot $n^i + 1$ and thus there would be no segment of age n^i in the buffer. This would remain the case until the segment that was in position $i-1$ at t moves to position i as time progresses and reaches age n^i after S seconds at $t + E$. The same thing happens to the other segments in the buffer.

This process repeats as the time advances such that every S seconds new segments would have spent w age slots in their current positions and one of them is at age slot n^i.

We assume the system reaches a steady state where, once that happens, every segment that reaches age n would exhibit the same behavior in terms of the number of blocks it has accumulated regardless of what time slot t it is at age slot n. Based on this, we denote by $X_n^{\mathcal{B}}$ the random variable of the number of blocks of $s_n^{\mathcal{B}}$; the segment of age n on some peer $\mathcal{B}$. Thus, $X_n^{\mathcal{B}}$ takes on values in the set $\{0, \ldots, m\}$.

We randomly select a peer $\mathcal{B}$ and study the block distribution of $X_n^{\mathcal{B}}$ of the segment $s_n^{\mathcal{B}}$ as it grows older starting from age slot $n = 0$ until it enters the playback position at $n = N$. We assume that all peers have the same behavior and protocol parameters, and thus we assume the distributions $pr[X_n^{\mathcal{B}} = k]$ for $n = 0, \ldots, N$ are the same across all peers and independent between peers.

Except for X_0, at any age slot $n = 1, \ldots, N$, X_n would depend only on the number of blocks at the previous age slot X_{n-1} and how many blocks, denoted by R_n, are received from neighbors during the current age slot. R_n determines how the transition from X_{n-1} to X_n would occur and depends on the buffer state plus the system parameters and the scheduling design choices. In the rest of this section, we obtain the distributions of X_n's and R_n's to form a series of equations that we solve starting from X_0 until X_N. The solutions X_n's would reveal the stochastic behavior of a segment over its lifetime in the buffer and enable us to obtain the continuity performance which is the most important metric for evaluating the user experience. The continuity, denoted by $P_{Cont.}$ could be defined as the average percentage of time the video playback is continuous or, in other words, non-interrupted. Thus, the continuity could be given by:

$$P_{Cont.} = pr[X_N = m] \tag{2.13}$$

where N is the age of the segment at the beginning of the playback position.

From now on, we will drop the peer $\mathcal{B}$ when referring to the number of blocks X_n at age n and only mention the peer where it is necessary to make clear what peer we are talking about.

2.2.1 Number of blocks at age 0, X_0

While the segment is still in the first position, it has a chance to receive blocks only from the server as it is still not ready-to-be-served on the peers due to our server design. After that, at age 0, where the segment has just entered the second position, it would have either received at least a linearly independent (L.I.) blocks from the server on the peers the server selected for this segment, or received nothing on other peers. Using this observation, we find the distribution of the number of blocks at age 0, X_0.

To that end, let E_s be the number of coded blocks the server can send during a segment duration. E_s could then be obtained from the equation:

$$E_s = \left\lfloor \frac{U_s S}{B_s} \right\rfloor \quad blocks/Seg. \tag{2.14}$$

where we take the floor of the right side to make sure E_s is integer.

we state a very important condition on the upload bandwidth of the server. On one hand, if $E_s < m$, the server capacity would not be sufficient to put all the m linearly independent blocks of the current segment in the network before the end of the segment duration. This would make peers unable to recover the segment whatever the conditions in the network are simply because some of its coded blocks may not exist. The system would fail in this case. Therefore, the following equation should be satisfied to have a viable system:

$$E_s \geq m \tag{2.15}$$

which implies the server capacity has to be at least slightly greater than the streaming rate. It is slightly greater and not equal to the streaming rate because of the extra bandwidth taken up by the coefficients overhead carried with each coded block.

On the other hand, if $\frac{E_s}{m} \geq Z$, then the server is able to serve complete segments to every peer in the overlay. In other words, the server is able to provide all the peers with the full streaming rate and the system turns into a server-client network. In this case the continuity would always be 1 under static network conditions and there is nothing to study in this case. Therefore, from now on we assume that the server bandwidth is not sufficient to provide all the peers with complete segments, and thus peers have to assist each other to make the segments complete before their playback deadline. Consequently, we study the system under the following condition:

$$\frac{E_s}{m} < Z \tag{2.16}$$

Now we can get back to obtaining the distribution of X_0. Let Z_s denote the number of peers the server can serve a linearly independent blocks to each in a segment duration, then:

$$Z_s = \frac{E_s}{a} \tag{2.17}$$

If $Z_s > Z$, the server is able more than a L.I. blocks to some or all peers. In this case, the server utilizes its extra bandwidth and sends $\lfloor \frac{Z_s}{Z} \rfloor a$ blocks to $Z - (Z_s \bmod Z)$ peers and $\lfloor \frac{Z_s}{Z} \rfloor a + a$ blocks to the rest of the peers $(Z_s \bmod Z)$ if Z_s is an integer. If Z_s is not an integer, there would be one more peer that would receive less than a blocks i.e. $E_s \bmod a$. The server chooses the peers to serve for each segment uniformly at random among all the peers in the overlay as we assumed earlier, the distribution of X_0 could be expressed as:

if Z_s is an integer:

$$pr[X_0 = k] = \begin{cases} \frac{Z-(Z_s \bmod Z)}{Z} & \text{if } k = \lfloor \frac{Z_s}{Z} \rfloor a \\ \frac{Z_s \bmod Z}{Z} & \text{if } k = \lfloor \frac{Z_s}{Z} \rfloor a + a \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

if Z_s is not an integer:

$$pr[X_0 = k] = \begin{cases} \frac{Z-1-(\lfloor Z_s \rfloor \bmod Z)}{Z} & \text{if } k = \left\lfloor \frac{\lfloor Z_s \rfloor}{Z} \right\rfloor a \\ \frac{\lfloor Z_s \rfloor \bmod Z}{Z} & \text{if } k = \left\lfloor \frac{\lfloor Z_s \rfloor}{Z} \right\rfloor a + a \\ \frac{1}{Z} & \text{if } k = E_s \bmod a \\ 0 & \text{otherwise} \end{cases} \qquad (2.19)$$

Equations (2.18) and (2.19) include also the case of $Z_s \leq Z$.

From the distribution of X_0, we can determine the average bandwidth contribution of the server $\overline{U_s}$ for each peer in the network as follows:

$$\overline{U_s} = \frac{B_s E[X_0]}{S}, \qquad \text{bps} \qquad (2.20)$$

where every S seconds, the server pushes an average of $E[X_0]$ L.I. coded blocks of the MRS to each peer with each block of size B_s. $\overline{U_s}$ is part of the bandwidth supply in the network while the remainder comes from the peers.

2.2.2 Number of blocks X_n for $n > 0$

In this section we obtain the distribution of the number of blocks X_n's throughout the rest of the segment lifetime servable by peers i.e. for $n = 1, \ldots, N$. To that end, let $R_n(B_j)$ be the Random Variable of the number of useful (linearly independent) coded blocks received to segment s_n^B from $\mathcal{B}$'s neighbors given $X_{n-1}^B = j$ blocks. R_n takes on values from the set $\{0 \ldots H\}$.
At any age slot n, X_n depends only on the number of blocks $s_n^{\mathcal{B}}$ had at the previous age slot X_{n-1} and how many blocks it would receive R_n at the current age slot n as

shown in Figure 2.3. The transition probability from X_{n-1} to X_n is then given by:

$$pr[X_n = k \,|\, X_{n-1} = j] = \begin{cases} 0 & \text{if } k < j, \\ pr[R_n(B_j) = k - j] & \text{if } j \leq k < m, \\ pr[R_n(B_j) \geq m - j] & \text{if } k = m, \end{cases} \tag{2.21}$$

where if the segment had $j < m$ blocks, the transition to m blocks would happen if it receives $m - j$ blocks or more. The extra blocks received over the $m - j$ are redundant and would be discarded.

Redundant blocks are caused by two reasons. First, the number of blocks in the

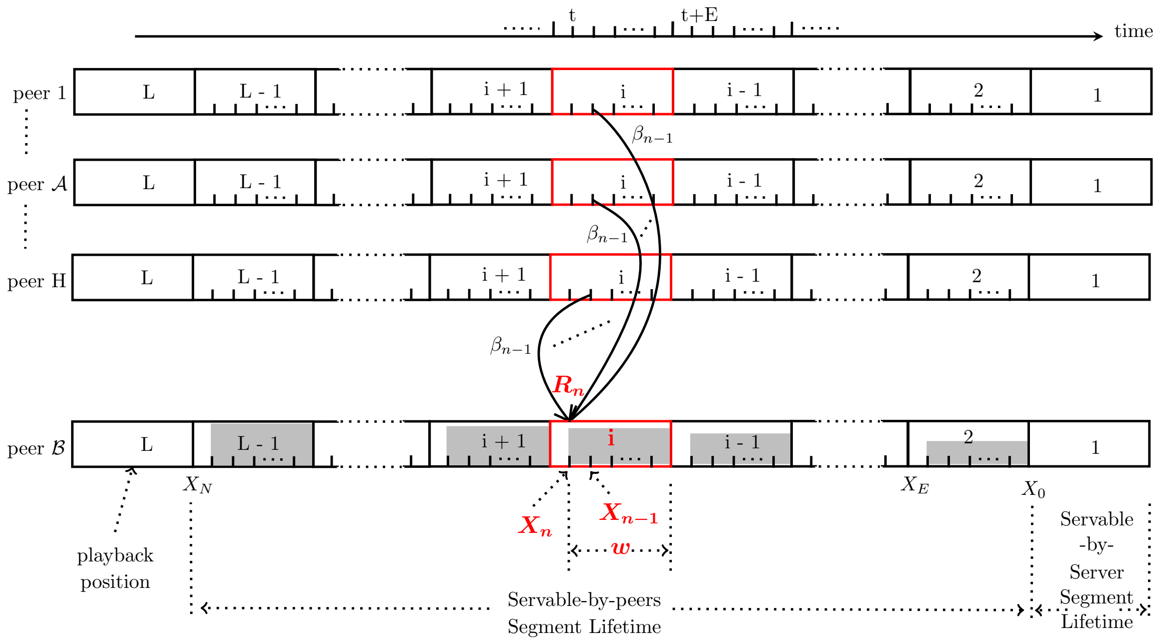


Figure 2.3: Transition probability from X_{n-1} to X_n depends on R_n

servable segments on downstream neighbors is not available to the peers since buffer maps received from neighbors indicate only whether a segment is complete or not. This is to reduce the size of the buffer map messages and make their sending frequency less. The second reason is that peers do not coordinate with each other when they do the scheduling.

Redundant blocks are much more likely to be received when the segment has most

of its blocks and is about to turn complete. Furthermore, with the randomization in the neighbor selection, and then the segment selection in the uniform strategy, the probability to receive even one redundant block is very low. Therefore, redundant blocks is not a major problem compared to the benefits gained. However, they may start to hurt the performance noticeably when large block sizes are used. We discuss how to mitigate their effect in chapter 3 by opting for smaller block sizes. Finally, it is worth mentioning that this problem could be dealt with by designing schemes to make peers, for example, start sending signals to gradually stop neighbors when the segment is *about* to get complete. Such schemes are out the scope of our research. The transition probability in eq. (2.21) is a conditional pmf with probabilities that sum up to 1 which we verify as follows:
for any $0 \leq j \leq m$, we have:

$$\begin{aligned}\sum_{k=0}^{m} pr[X_n = k \mid X_{n-1} = j] &= 0 + \sum_{k=j}^{m-1} pr[R_n(B_j) = k - j] + pr[R_n(B_j) \geq m - j] \\ &= \sum_{k=0}^{m-1-j} pr[R_n(B_j) = k] + pr[R_n(B_j) \geq m - j] \\ &= pr[R_n(B_j) < m - j] + pr[R_n(B_j) \geq m - j] \\ &= 1\end{aligned}$$

The distribution of X_n could be found by averaging the conditional pmf in (2.21) over $pr[X_{n-1} = j]$:

$$pr[X_n = k] = \sum_{j=0}^{m} pr[X_n = k | X_{n-1} = j] pr[X_{n-1} = j] \tag{2.22}$$

2.2.3 R_n the number of blocks received from neighbors to $s_n^{\mathcal{B}}$

As we mentioned previously, the number of coded blocks R_n received at age slot n controls how the transition from X_{n-1} to X_n happens. Therefore, before we can obtain the distributions of X_n's, the distribution of R_n has to be found for all $n = 0, \ldots, N$. To that end, let $\beta_{n-1}(\mathcal{B}_j)$ be the probability that a random neighbor of $\mathcal{B}$, denoted by $\mathcal{A}$, pushes a useful (Linearly independent) coded block to $s_n^{\mathcal{B}}$ given $X_{n-1}^{\mathcal{B}} = j$. We assumed earlier that all the peers are identical and independent, which allows us to view each one of the H neighbors of $\mathcal{B}$ as an independent Bernoulli trial with the success probability being sending a coded block to $s_n^{\mathcal{B}}$. Therefore, $R_n(\mathcal{B}_j)$ has a binomial distribution with parameters (H,β_{n-1}):

$$pr[R_n(\mathcal{B}_j) = k] = \binom{H}{k} (\beta_{n-1}(\mathcal{B}_j))^k (1 - \beta_{n-1}(\mathcal{B}_j))^{H-k} \quad \text{,for } k = 0, \ldots, H \qquad (2.23)$$

At any age slot n, whatever time slot t the system is at, we must find the success probability β_n in order to compute the distribution of R_n. Once that is done, we can solve the series of equations in (2.22) starting from X_0, that is given in equations (2.18) and (2.19), to obtain X_1 then X_2 and so on so forth all the way to X_N. The solutions X_n's reveal the stochastic behavior of the number of blocks in a segment as it grows older in the buffer until it reaches its playback deadline at age N at the beginning of the playback position. Then the segment would be played back if it is complete with probability $pr[X_N = m]$ which is the probability of continuity defined in eq. (2.13), or otherwise skipped if it is not complete.
The probability β_n depends on the buffer states at n in addition to the scheduling algorithm design choices (neighbor selection, segment selection) on the neighbors. We obtain β_n in the rest of this chapter.

2.2.4 β_n, probability a neighbor pushes a useful block to $s_n^{\mathcal{B}}$

As we mentioned previously, the age a segment has reached so far would imply its current buffer position and the time it has spent in it. Suppose the segment of age n on some peer $\mathcal{B}$ is in buffer position $i = 2, \ldots, L-1$ where it has spent w age slots. i and w are given in equations (2.10) and (2.11) respectively. The ages of the $L-3$ segments in the other servable-by-peers buffer positions could be obtained from equation (2.12). In this section, we investigate what chance $s_n^{\mathcal{B}}$ has, given its position and the current time it has spent in it, to receive a useful coded block from a neighbor.

At each age n, all H neighbors of $\mathcal{B}$ run the scheduling algorithm whereby each peer decides for each block it can send what neighbor and what segment on that neighbor to push the block to. Let $\mathcal{A}$ be a random neighbor of $\mathcal{B}$. For $s_n^{\mathcal{B}}$ to receive a useful coded block from $\mathcal{A}$, $\mathcal{A}$ has to i) select $\mathcal{B}$ among all the neighbors it can serve at age n, ii) select $s_n^{\mathcal{B}}$ among all the segments it can serve on $\mathcal{B}$. iii) if both $\mathcal{B}$ and $s_n^{\mathcal{B}}$ are selected, the coded block generated by $\mathcal{A}$ and pushed to $\mathcal{B}$ has to be useful i.e. L.I. with the current coded blocks of $s_n^{\mathcal{B}}$ to be added to the segment. The scheduling algorithm on $\mathcal{A}$ will use the buffer maps it has received from its neighbors including $\mathcal{B}$ in addition to the buffer state of $\mathcal{A}$ to decide what neighbors are servable at n. Therefore, given $s_n^{\mathcal{B}}$ has j blocks, $\beta_n(\mathcal{B}_j)$ could be expressed as:

$$\beta_n(\mathcal{B}_j) = pr\left[\mathcal{B}_j \text{ is selected, } s_n^{\mathcal{B}} \text{ is selected, useful block}\right]$$

When $X_n^{\mathcal{B}} = m$, β_n would be 0 since $s_n^{\mathcal{B}}$ is complete and would not receive any blocks. $s_n^{\mathcal{B}}$ has a chance to receive blocks only when it is not complete i.e. $X_n < m$. Thus we can write:

$$\beta_n(\mathcal{B}_j) = \begin{cases} 0 & \text{if } j = m, \\ \beta_n(\mathcal{B}_{j<m}) & \text{if } j < m, \end{cases} \tag{2.24}$$

In the rest of the section, we investigate the case when $X_{n-1} = j < m$.

If we define the following notations:

$$\alpha_n(\mathcal{B}_{j<m}) = pr\,[\mathcal{B}_{j<m} \text{ is selected}]$$
$$\gamma_n(\mathcal{B}_{j<m}) = pr\left[s_n^{\mathcal{B}} \text{ is selected} \mid B_{j<m} \text{ is selected}\right]$$
$$\lambda_n(\mathcal{B}_{j<m}) = pr[\text{useful code} \,|\, \mathcal{B}_{j<m} \text{ is selected, } s_n^{\mathcal{B}} \text{ is selected}]$$

then $\beta_n(\mathcal{B}_{j<m})$ could be written as:

$$\beta_n(\mathcal{B}_{j<m}) = \alpha_n(\mathcal{B}_{j<m})\,\gamma_n(\mathcal{B}_{j<m})\,\lambda_n(\mathcal{B}_{j<m}) \tag{2.25}$$

Next we find the probability components of β_n.

α_n, probability peer $\mathcal{B}$ is selected by neighbor $\mathcal{A}$ at age n

Here we model the neighbor selection part of the scheduling algorithm in which a peer would select uniformly at random one of its neighbors that it can server at the current age slot.

Peer $\mathcal{A}$ selects peer $\mathcal{B}$, only if it is servable, uniformly at random from all the servable neighbors at age slot n. If $\mathcal{B}$ is not servable at n, $\mathcal{A}$ will not select it. Therefore, if we let

$\boldsymbol{Q_n}$ the number of neighbors servable by $\mathcal{A}$ at age slot n among the $H-1$ neighbors of $\mathcal{A}$ that does not include peer $\mathcal{B}$. $Q_n \in \{0, \ldots, H-1\}$

$\boldsymbol{Q_n(\mathcal{B}_j)}$ the indicator function of whether or not peer $\mathcal{B}_j$ is servable by $\mathcal{A}$ at age slot n given $X_{n-1}^{\mathcal{B}} = j$.

then we can express α_n for any $k = 0, \ldots, H-1$ as follows:

$$\alpha_n(Q_n(\mathcal{B}_{j<m}), Q_n{=}\,k) = \begin{cases} \dfrac{1}{k+1} & \text{if } Q_n(\mathcal{B}_{j<m}) = 1 \\ 0 & \text{if } Q_n(\mathcal{B}_{j<m}) = 0 \end{cases} \tag{2.26}$$

Next we find the distributions of Q_n and $Q_n(\mathcal{B}_{j<m})$. A segment of age n, $s_n^{\mathcal{C}}$, on a neighbor $\mathcal{C}$ of $\mathcal{A}$ is servable by $\mathcal{A}$ if it has less than m blocks at n, i.e. not complete, on $\mathcal{C}$ and has a or more blocks on $\mathcal{A}$ i.e. Ready-to-be-Served. Therefore, if we let V_n denote the probability that the segment $s_n^{\mathcal{C}}$ is servable by $\mathcal{A}$ at n, then:

$$V_n = pr[X_n^C < m]\, pr[X_n^A \geq a] \tag{2.27}$$

Let V_n' be the probability of the complement of V_n, then:

$$V_n' = 1 - V_n \tag{2.28}$$

A peer is servable by an upstream neighbor $\mathcal{A}$ at age slot n if at least one of its $L-2$ segments (2 for the first and last segments) is servable by $\mathcal{A}$. Let θ_n be the probability that neighbor $\mathcal{C}$ is servable by $\mathcal{A}$ at age slot n. The complement event that $\mathcal{C}$ is *not* servable by $\mathcal{A}$ at age n occurs when none of $\mathcal{C}$'s segments are servable by $\mathcal{A}$ at n. Assuming that segment servability is independent of whether or not other segments are servable, we have

$$\theta_n' = \prod_{i=1}^{L-2} V'_{(n \bmod E)\,+\,iE} \tag{2.29}$$

where the subscript $(n \bmod E) + iE$ generates the age of the segments in buffer positions $2, \ldots, L-1$ at age slot n where each segment has spent $(n \bmod E)$ age slots in its current position.
θ_n is then given by:

$$\theta_n = 1 - \theta_n' \tag{2.30}$$

The probability θ_n is the same for all the neighbors of $\mathcal{A}$ except $\mathcal{B}$. Therefore, since peers are independent, Q_n would have a binomial distribution with parameters

$(H-1, \theta_n)$:

$$pr[Q_n = k] = \binom{H-1}{k} (\theta_n)^k (1-\theta_n)^{H-1-k} \tag{2.31}$$

On the other hand, since $s_n^{\mathcal{B}}$ is known to have j blocks as indicated in the conditional probability in (2.21), peer $\mathcal{B}$ would have a slightly different probability of being servable by $\mathcal{A}$. Let $V_n(\mathcal{B}_{j<m})$ and $\theta_n(\mathcal{B}_{j<m})$ and their complement probabilities have the same definition as before except now they are conditioned on the fact that $s_n^{\mathcal{B}}$ has j blocks. Then:

$$V_n(\mathcal{B}_{j<m}) = pr[X_n^{\mathcal{A}} \geq a] \tag{2.32}$$

$$\theta'_n(\mathcal{B}_{j<m}) = pr[X_n^{\mathcal{A}} < a] \prod_{\substack{i=1 \\ i \neq \lceil \frac{n+1}{E} \rceil + 1}}^{L-2} V'_{(n \mod E) + iE} \tag{2.33}$$

where the subscript $i \neq \lceil \frac{n+1}{E} \rceil + 1$ is to make sure the position of $s_n^{\mathcal{B}}$ is skipped as we already know it is not complete $j < m$, and thus would not be servable only if it has less than a blocks on $\mathcal{A}$ i.e. with probability $pr[X_n^{\mathcal{A}} < a]$. Consequently, probability peer $\mathcal{B}_{j<m}$ is servable by $\mathcal{A}$ at n is given by:

$$pr[Q(\mathcal{B}_{j<m}) = 1] = \theta_n(\mathcal{B}_{j<m}) = 1 - \theta'_n(\mathcal{B}_{j<m}) \tag{2.34}$$

γ_n, probability the segment of age n on peer $\mathcal{B}$ is selected

Following the selection of a neighbor, the second part of the scheduling algorithm is to select what segment is to be served a coded block among the servable segments in the buffer of the selected neighbor. Different strategies could be employed for the segment selection. We choose to model the most intuitive strategy i.e. the most urgent by which the segment closest to the playback deadline that is still servable is given priority. To find out how the most urgent compare to other strategies, we model the uniform strategy by which segments that are still servable in the buffer are given equal chances. Next we investigate the probability that $s_n^{\mathcal{B}}$ is selected by $\mathcal{A}$

given $\mathcal{B}_{j<m}$ was selected by $\mathcal{A}$.

The fact that $\mathcal{B}_{j<m}$ was selected by $\mathcal{A}$ implies that $\mathcal{B}$ is servable by $\mathcal{A}$. Thus we are after:

$$\gamma_n(\mathcal{B}_{j<m}) = pr[s_n^{\mathcal{B}_{j<m}} \text{ is selected} \mid Q_n(\mathcal{B}_{j<m}) = 1] \tag{2.35}$$

To find γ_n, we denote by 1_i the state when the segment in buffer position i on peer $\mathcal{B}_{j<m}$ is servable by $\mathcal{A}$, and by 0_i the state when it is not servable. Then the buffer state of $\mathcal{B}_{j<m}$ corresponding to the event $Q_n(\mathcal{B}_{j<m}) = 1$ could be represented by the set:

$$\Omega^{A}_{B_{j<m}} = \left\{ Y_{L-1} \ldots Y_i \ldots Y_2 \mid \sum_{i=2}^{L-1} Y_i > 0, Y_i = 0, 1 \right\} \tag{2.36}$$

which means that at least one segment or more on $\mathcal{B}$ should be servable by $\mathcal{A}$ for peer $\mathcal{B}_{j<m}$ to be servable by $\mathcal{A}$.

Let Δ_n be the event $\{s_n^{\mathcal{B}_{j<m}}$ is selected by $\mathcal{A}\}$, then from equation 2.35, γ_n could be written:

$$\gamma_n(\mathcal{B}_{j<m}) = pr[\Delta_n \mid \Omega^{\mathcal{A}}_{\mathcal{B}_{j<m}}] = \frac{pr\left[\Delta_n \cap \Omega^{\mathcal{A}}_{\mathcal{B}_{j<m}}\right]}{pr\left[\Omega^{\mathcal{A}}_{\mathcal{B}_{j<m}}\right]} \tag{2.37}$$

We already found $pr[\Omega^{\mathcal{A}}_{\mathcal{B}_{j<m}}]$ to be equal to $\theta_n(\mathcal{B}_{j<m})$ in eq. (2.34). Thus we need to find only the probability of the numerator which depends on Δ_n. The event Δ_n depends on the segment selection strategy used. Next we find its probability for several segment selection strategies.

Using the Most Urgent Segment Selection Strategy

In this strategy peer $\mathcal{A}$ selects the closest segment to the playback deadline that it can serve at age slot n. Therefore, if $s_n^{\mathcal{B}_{j<m}}$ is in position i, Δ_n could be represented

by the following set for $i = 1, \dots, L-1$:

$$\Delta_n = \{0_{L-1} \dots 0_{i+1} 1_i Y_{i-1} \dots Y_2 \mid Y_i = 0, 1\} \tag{2.38}$$

which means that $s_n^{\mathcal{B}_{j<m}}$ is selected only if it is servable (1_i) and all the older segments $(0_{L-1} \dots 0_{i+1})$ are not servable, whatever the state may be for all the other younger segments.

Comparing this set with the set $\Omega^{\mathcal{A}}_{\mathcal{B}_{j<m}}$ in eq. (2.36), we can see right away that $\Delta_n \subset \Omega^{\mathcal{A}}_{\mathcal{B}_{j<m}}$. From this relation, the formula of γ_n in eq. (2.37) could be written:

$$\begin{aligned}\gamma_n(\mathcal{B}_{j<m}) &= \frac{pr\left[\Delta_n\right]}{pr\left[\Omega^{\mathcal{A}}_{\mathcal{B}_{j<m}}\right]} \\ &= \frac{pr[X_n^{\mathcal{A}} \geq a] \displaystyle\prod_{k=i+1}^{L-1} V'_{(n \mod m)\,+\,(k-1)m}}{\theta_n(\mathcal{B}_{j<m})}\end{aligned} \tag{2.39}$$

Using the Uniform Segment Selection Strategy

In this strategy peer $\mathcal{A}$ selects a segment on $\mathcal{B}$ uniformly at random among only the segments that are servable by it.

For the event $\Delta_n = \{\mathcal{A} \text{ selects } s_n^{\mathcal{B}_{j<m}}\}$ to take place, the segment $s_n^{\mathcal{B}_{j<m}}$ must be servable while the other $(L-3)$ segments may or may not be servable. In addition, $pr[\Delta_n]$ depends on how many of the other segments are servable at n. Therefore, let the set Δ_n^k represent the buffer state when $s_n^{\mathcal{B}_{j<m}}$ is servable and there are k other servable segments. Thus for any $k = 0, \dots, L-3$, Δ_n^k's could be expressed as:

$$\Delta_n^k = \left\{ Y_{L-1} \dots 1_i \dots Y_2 \mid Y_i \in \{0,1\}, \sum_{\substack{j=2 \\ j \neq i}}^{L-1} Y_j = k \right\} \tag{2.40}$$

we notice that Δ_n^k's are partitions of Ω in eq. (2.36), and therefore, we can find the

$pr\left[\Delta_n \cap \Omega^{\mathcal{A}}_{\mathcal{B}_{j<m}}\right]$ as follows:

$$\begin{aligned}
pr\left[\Delta_n \cap \Omega^{\mathcal{A}}_{\mathcal{B}_{j<m}}\right] &= pr\left[\Delta_n \cap \left(\bigcup_{k=0}^{L-3} \Delta_n^k\right)\right] \\
&= pr\left[\bigcup_{k=0}^{L-3} \left(\Delta_n \cap \Delta_n^k\right)\right] = \sum_{k=0}^{L-3} pr\left[\Delta_n \cap \Delta_n^k\right] \\
&= \sum_{k=0}^{L-3} pr\left[\Delta_n \mid \Delta_n^k\right] pr\left[\Delta_n^k\right] \\
&= \sum_{k=0}^{L-3} \frac{1}{k+1} pr[X_n^{\mathcal{A}} \geq a] pr[C_n^{\mathcal{B}} = k] \qquad (2.41)
\end{aligned}$$

where we define $C_n^{\mathcal{B}}$ to be the number of buffer positions on $\mathcal{B}$ excluding the position of $s_n^{\mathcal{B}_{j<m}}$ that contain segments servable by $\mathcal{A}$ at age slot n. Next we find its distribution. $C_n^{\mathcal{B}} \in \{0, \ldots, L-3\}$. We consider each buffer position as a Bernoulli trial with the success being having a servable segment. Although the servability of the segments is independent from each other as we assumed earlier, we can not use the binomial distribution for $C_n^{\mathcal{B}}$ because each position has a different success probability. Consequently, $C_n^{\mathcal{B}}$ is the sum of independent Bernoulli variables with not-all-equal success probabilities given in eq. (2.27). The distribution of $C_n^{\mathcal{B}}$ is known as the Poisson-Binomial Distribution in the literature and could be computed in our case from the following formula:

$$pr[C_n^B = k] = \sum_{\substack{i_1=2 \\ i_1 \neq i}}^{L-k} V_{i_1} \sum_{\substack{i_2=i_1+1 \\ i_2 \neq i}}^{L-k+1} V_{i_2} \ldots \sum_{\substack{i_k=i_{k-1}+1 \\ i_k \neq i}}^{L-1} V_{i_k} \prod_{\substack{j=2 \\ j \neq i_1 \\ \vdots \\ j \neq i_k \\ j \neq i}}^{L-1} V'_j \qquad (2.42)$$

where the position of s_n^B, i, is skipped as shown in the subscripts because C_n^B does not include it. In addition, the subscripts of V and V' represent the buffer position of the segment.

We earlier, in equation 2.27, defined the subscript to be the age of the segment, and here we have changed the notation just to have less clutter in the formula. Furthermore, from the position, say d, of any segment, we can find its current age at age slot n as: $n \mod E + (d-1)E$.

This formula accounts for all the cases of having k servable segments in any buffer position (except i) and $L-3-k$ non-servable segments in the remaining positions for any $k = 0, \ldots, L-3$. However, the problem with this formula is that the number of cases of having k successes, which is given by $\binom{L-3}{k}$, becomes very large even with relatively small values of L making the calculation inefficient. Fortunately, there are much more efficient methods to compute it. In [37], they compare the different methods used in the literature to compute it including a recursive formula described in [38]. They refer to that formula as RF1 in their paper and we use this same formula here to compute the distribution of C_n. If we let C_n^j be the poisson binomial random variable with total number of trials j with success probabilities $V_1, \ldots, V_j$, and also let $\xi_{k,j} = pr[C_n^j = k]$, then the recursive formula could be expressed as follows in our case:

$$\xi_{k,j} = (1 - V_j)\xi_{k,j-1} + V_j\,\xi_{k-1,j-1}, \qquad 0 \le k \le L-3, 0 \le j \le L-3 \tag{2.43}$$

with the boundary conditions are $\xi_{-1,j} = \xi_{j+1,j} = 0, j = 0, \ldots, L-3$ and $\xi_{0,0} = 1$.

Now the probability $\gamma_n(\mathcal{B}_{j<m})$ could be obtained by substituting eq. (2.41) into (2.37):

$$\gamma_n(\mathcal{B}_{j<m}) = \frac{pr[X_n^A \ge a]}{\theta_n(B_{j<m})} \sum_{k=0}^{L-3} \frac{1}{k+1}\, pr[C_n^B = k] \tag{2.44}$$

λ_n, probability a received coded block is useful

Even after peer $\mathcal{A}$ has decided to send a coded block to the segment of age n on peer $\mathcal{B}$, the block may not be useful to $\mathcal{B}$. By "Not Useful" we mean that the received block is linearly dependent on the existing blocks of $s_n^{\mathcal{B}}$, in which case it will be discarded. Therefore, not every received block to $s_n^{\mathcal{B}}$ is useful and we need to find the probability that a received block is linearly independent. In [26], they estimate the probability of a useful block based on the following lemma:

Lemma 1. *(Lemma 2.1, [28]) Let $S_{\mathcal{B}}$ denote the space spanned by the coded blocks on peer $\mathcal{B}$ and $S_{\mathcal{A}}$ denote the space spanned on one of peer $\mathcal{B}$'s upstream peers, namely peer $\mathcal{A}$. Consider a coded block x sent from peer $\mathcal{A}$ to peer $\mathcal{B}$. Then,*

$$pr[\textit{coded block } x \textit{ is useful} \mid S_{\mathcal{A}} \not\subseteq S_{\mathcal{B}}] \geq 1 - \frac{1}{q},$$

where q is the size of the Galois field.

Let p denote the probability of the event $\{S_{\mathcal{A}} \subseteq S_{\mathcal{B}}\}$, then the probability the coded block is useful, denoted by $I_{\mathcal{AB}}$ could be approximated by:

$$I_{\mathcal{AB}} = pr[\text{coded block x is useful}] \geq (1 - \frac{1}{q})(1 - p) \tag{2.45}$$

The problem with this approach in [26], in addition to only providing a lower bound for $I_{\mathcal{AB}}$, is that it also gives an arbitrary constant value to $p = pr[S_{\mathcal{A}} \subseteq S_{\mathcal{B}}]$. This does not take care of how the number of blocks on peers $\mathcal{A}$ and $\mathcal{B}$ and the parameters a and m affect p, and how p changes with the age of the segment. For example, our intuition tells us that larger values of $X_n^{\mathcal{B}} = j$ leads to smaller p as the space spanned by vectors on peer $\mathcal{B}$ becomes larger. Next, we work on this problem.

Let $s_n^{\mathcal{A}}$ (the segment of age n) on the upstream neighbor $\mathcal{A}$ have $X_n^{\mathcal{A}} = k$ blocks denoted by $\vec{V_{\mathcal{A}_k}} = \{v_1, v_2, \dots, v_k\}$. These blocks are linearly independent, and each of them has m components in the m-dimensional vector space over the Galois Field of size q. Since they are linearly independent they span a k-dimensional subspace $S_{\mathcal{A}_k}^n$. We also know that $s_n^{\mathcal{B}}$ on the downstream peer $\mathcal{B}$ has $X_n^{\mathcal{B}} = j$ linearly independent

blocks and it is not complete i.e. $j < m$. Let $\vec{\boldsymbol{U_{\mathcal{B}_j}}} = \{u_1, \ldots, u_j\}$ and $S^n_{\mathcal{B}_j}$ denote the set of the vectors on $\mathcal{B}$ and their spanned j-dimensional subspace respectively.
Let $x^{\mathcal{A}_k\mathcal{B}_j}$ be the coded block sent from $\mathcal{A}$ to $\mathcal{B}$. $x^{\mathcal{A}_k\mathcal{B}_j}$ is generated as a linear combination of the vectors on $\mathcal{A}$ which makes it be in their subspace $S^n_{\mathcal{A}_k}$. If $x^{\mathcal{A}_k\mathcal{B}_j}$ happens to fall in the subspace of the vectors on $\mathcal{B}$, $S^n_{\mathcal{B}_j}$, it will make it linearly dependent with $\vec{\boldsymbol{U_{\mathcal{B}_j}}}$ and will provide no useful information. In other words, $x^{\mathcal{A}_k\mathcal{B}_j}$ will not be useful if it falls in $S^n_{\mathcal{A}_k} \cap S^n_{\mathcal{B}_j}$. Thus for $x^{\mathcal{A}_k\mathcal{B}_j}$ to be useful, it should fall outside $S^n_{\mathcal{B}_j}$.
Let $I_n^{\mathcal{A}_k\mathcal{B}_j}$ be the indicator function of the event $x^{\mathcal{A}_k\mathcal{B}_j}$ falls outside $S^n_{\mathcal{B}_j}$. To find $pr[I_n^{\mathcal{A}_k\mathcal{B}_j} = 1]$, we need to find the probability of the relationship between $S^n_{\mathcal{A}_k}$ and $S^n_{\mathcal{B}_j}$. To that end, let K' denote the number of coded blocks (vectors) in $\vec{\boldsymbol{V_{\mathcal{A}_k}}}$ on peer $\mathcal{A}$ that are linearly dependent with $\vec{\boldsymbol{U_{\mathcal{B}_j}}}$, and also let $P_n(k'; k, j)$ be its pmf. K' could assume any value in $\{0, \ldots, k\}$. For example, on one hand, if $K' = 0$, then the intersection of $S^n_{\mathcal{A}_k}$ and $S^n_{\mathcal{B}_j}$ is empty, and any generated coded block falls outside $S^n_{\mathcal{B}_j}$ and is linearly independent. On the other hand, if $K' = k$, this means $S^n_{\mathcal{A}_k} \subseteq S^n_{\mathcal{B}_j}$, and every generated coded block will be linearly dependent. Thus we could write:

$$pr[I_n^{\mathcal{A}_k\mathcal{B}_j} = 1 | K' = k'] = \begin{cases} 1 & \text{if } k' = 0, \\ 1 - \dfrac{1}{q^{k-k'}} & \text{if } k' > 0, \end{cases} \tag{2.46}$$

where for $0 < K' < k$, the generated coded vector is in $S^n_{\mathcal{B}_j}$ if and only if all the $k - K'$ independent of $\vec{\boldsymbol{U_{\mathcal{B}_j}}}$ vectors are multiplied with zero coefficients when forming the linear combination, and only the K' dependent vectors are multiplied with not-all-zero coefficients. The probability to select $k - K'$ zero coefficients is $1/q^{k-k'}$ since they are chosen independently and uniformly at random from $GF(q)$. Its complement gives the desired probability.
Next we find the distribution of K'. The actual range of values K' assumes depends on k, j and m. The lower limit is dictated by the inequality $k - k' + j \leq m$ where

the number of linearly independent vectors in any set cannot exceed m. Then k' is given by:

$$k' \geq max(k + j - m, 0)$$

where if $k + j < m$, k' starts from 0.

For the upper limit, we distinguish between two cases. First, when $k \leq j$, all of the k vectors could be linearly dependent with the j vectors on peer $\mathcal{B}$. Thus, in this case, we have $k' \leq k$ which implies $k - k' \geq 0$. In the second case, when $k > j$, only a maximum of j vectors in $\vec{\boldsymbol{V}}_{\boldsymbol{\mathcal{A}_k}}$ could be linearly dependent with $\vec{\boldsymbol{U}}_{\boldsymbol{\mathcal{B}_j}}$, and the remaining $k-j$ must be linearly independent with $\vec{\boldsymbol{U}}_{\boldsymbol{\mathcal{B}_j}}$. That is because the subspace $S^n_{\mathcal{B}_j}$, spanned by the set $\vec{\boldsymbol{U}}_{\boldsymbol{\mathcal{B}_j}}$, cannot have a basis that contains more vectors than its dimension j. Thus, in this case, we have $k' \leq j$ and $k - k' \geq k - j$. Combining the two cases together, we obtain the following upper limit of k':

$$k' \leq min(k, j)$$

and also the following lower limit for the number of linearly independent vectors (which we will use later):

$$k - k' \geq max(k - j, 0) \tag{2.47}$$

from the last two inequalities for k', we obtain the range of values k' can assume:

$$max(k + j - m, 0) \leq k' \leq min(k, j) \tag{2.48}$$

Next, we find the distribution of k' given k and j at age slot n, $P_n(k'; k, j)$. To do that, we employ a urn model as in [39]. We imagine an urn containing all the possible non-zero vectors in the whole m-dimensional vector space. Let's consider one of the vectors v_{i_1} in $\vec{\boldsymbol{V}}_{\boldsymbol{\mathcal{A}_k}}$. v_{i_1} may have come from any neighbor of peer $\mathcal{A}$. And the subspace, from which v_{i_1} was selected, could be any subspace that has a dimension higher than a in the m-dim. space, because it was, in turn, formed by vectors that could have come again from any subspace on any neighbor. Therefore, we assume that v_{i_1} is equally likely to be any vector in the m-dim. space.

The event that v_{i_1} is linearly dependent with $\vec{\boldsymbol{U}}_{\boldsymbol{\mathcal{B}_j}}$ is equivalent to the event that v_{i_1} is in the subspace $S^n_{\mathcal{B}_j}$ spanned by $\vec{\boldsymbol{U}}_{\boldsymbol{\mathcal{B}_j}}$. Thus, based on our earlier assumption, we could write:

$$pr[v_{i_1} \in S^n_{\mathcal{B}_j}] = \frac{q^j - 1}{q^m - 1}$$

where the numerator and the denominator are the total number of non-zero vectors in $\vec{\boldsymbol{U}}_{\boldsymbol{\mathcal{B}_j}}$ and the whole m-dim. space respectively. Suppose v_{i_2} is another vector in $\vec{\boldsymbol{U}}_{\boldsymbol{\mathcal{B}_j}}$, then:

$$\begin{aligned} pr[v_{i_1} \in S^n_{\mathcal{B}_j}, v_{i_2} \in S^n_{\mathcal{B}_j}] &= pr[v_{i_2} \in S^n_{\mathcal{B}_j} | v_{i_1} \in S^n_{\mathcal{B}_j}] pr[v_{i_1} \in S^n_{\mathcal{B}_j}] \\ &= \frac{q^j - 1}{q^m - 1} \frac{q^j - 1 - (q - 1)}{q^m - 1} \\ &= \frac{q^j - 1}{q^m - 1} \frac{q^j - q}{q^m - 1} \end{aligned}$$

where since v_{i_1} and v_{i_2} are linearly independent, there are $q-1$ non-zero vectors, that are multiples of v_{i_1}, that v_{i_2} cannot be selected from. Following the same logic, we have:

$$\begin{aligned} pr[v_{i_1} \in S^n_{\mathcal{B}_j}, v_{i_2} \in S^n_{\mathcal{B}_j}, \ldots, v_{i_d} \in S^n_{\mathcal{B}_j}] &= \frac{q^j - 1}{q^m - 1} \frac{q^j - q}{q^m - 1} \cdots \frac{q^j - q^{d-1}}{q^m - 1} \\ &= \prod_{i=0}^{d-1} \frac{q^j - q^i}{q^m - 1} \end{aligned} \tag{2.49}$$

Now let v_{l_1} be another vector that is linearly independent with the set $\{v_{i_1}, \ldots, v_{i_d}\}$. Let us consider the probability of the event v_{l_1} is outside $S^n_{\mathcal{B}_j}$ given that $\{v_{i_1}, \ldots, v_{i_d}\}$

is in $S^n_{\mathcal{B}_j}$:

$$\begin{aligned} pr[v_{l_1} \notin S^n_{\mathcal{B}_j} | \{v_{i_1}, \dots, v_{i_d}\} \subseteq S^n_{\mathcal{B}_j}] &= \frac{q^m - 1 - (q^j - 1)}{q^m - 1} \\ &= \frac{q^m - q^j}{q^m - 1} \\ &= pr[v_{l_1} \notin S^n_{\mathcal{B}_j}] \end{aligned}$$

where the numerator in the second equation is the number of non-zero vectors outside $S^n_{\mathcal{B}_j}$. The third equation comes from the observation that the event $v_{l_1} \notin S^n_{\mathcal{B}_j}$ is independent of $\{v_{i_1}, \dots, v_{i_d}\} \subseteq S^n_{\mathcal{B}_j}$ since being outside $S^n_{\mathcal{B}_j}$ makes v_{l_1} automatically linearly independent of any vectors in $S^n_{\mathcal{B}_j}$.

Extending the argument for a set of linearly independent vectors $\{v_{l_1}, \dots, v_{l_y}\}$ that is linearly independent with with $\{v_{i_1}, \dots, v_{i_d}\}$, we have the following probability:

$$pr[\{v_{l_1}, \dots, v_{l_y}\} \not\subseteq S^n_{\mathcal{B}_j} | \{v_{i_1}, \dots, v_{i_d}\} \subseteq S^n_{\mathcal{B}_j}] = pr[\{v_{l_1}, \dots, v_{l_y}\} \not\subseteq S^n_{\mathcal{B}_j}]$$

$$\begin{aligned} pr[\{v_{l_1}, \dots, v_{l_y}\} \not\subseteq S^n_{\mathcal{B}_j}] &= \frac{q^m - q^j}{q^m - 1} \frac{q^m - q^j - (q-1)}{q^m - 1} \dots \frac{q^m - q^j - (q^{y-1} - 1)}{q^m - 1} \\ &= \prod_{l=0}^{y-1} \frac{q^m - q^j - q^l + 1}{q^m - 1} \end{aligned} \tag{2.50}$$

Now we are ready to find the distribution $P_n(k'; k, j)$. First, we rewrite inequality 2.48 as follows:

$$0 \leq k' - max(k + j - m, 0) \leq min(k, j) - max(k + j - m, 0) \tag{2.51}$$

if we let $k'' = k' - max(k + j - m, 0)$, then:

$$P_n(k' = c; k, j) \equiv pr[k'' = c - max(k + j - m, 0)]$$

from inequalities 2.48 and 2.47, we note the probability $pr[k'' = c - max(k+j-m, 0)]$ is conditioned on having $max(k+j-m, 0)$ vectors in $S^n_{\mathcal{B}_j}$ and $max(k-j, 0)$ outside $S^n_{\mathcal{B}_j}$ and , based on equation 2.49 and 2.50, is given by:

$$pr[k'' = y] = \binom{min(k,j) - max(k+j-m,0)}{y} \prod_{i_1=0}^{y-1} \frac{q^j - q^{max(k+j-m,0)+i_1}}{q^m - 1} \prod_{i_2=0}^{min(k,j)-max(k+j-m,0)-y-1} \frac{q^m - q^j - q^{max(k-j,0)+i_2} + 1}{q^m - 1} \quad (2.52)$$

and the distribution of k' is then given by:

$$\begin{aligned} P_n(k';k,j) &= pr[k'' = k' - max(k+j-m,0)] \\ &= \binom{min(k,j) - max(k+j-m,0)}{k' - max(k+j-m,0)} \prod_{i_1=max(k+j-m,0)}^{k'-1} \frac{q^j - q^{i_1}}{q^m - 1} \\ &\quad \prod_{i_2=max(k-j,0)}^{min(k,j)+max(k-j,0)-k'-1} \frac{q^m - q^j - q^{i_2} + 1}{q^m - 1} \end{aligned} \quad (2.53)$$

To check if the sum of the probabilities for all the values of k' given k and j would add up to 1, we compute the distribution $P_n(k';k,j)$ when $m = 50$, $q = 2^8$, and $a = 1$ for the all the possible values of k and j. We found out that the sum is 1 in most cases except when $k = j = 49$ where we got the worst value of 0.999985. This confirms that $P_n(k';k,j)$ is actually a conditional probability distribution.

The probability of the event $pr[I_n^{\mathcal{A}_k\mathcal{B}_j} = 1]$ can be obtained by averaging the conditional probability in equation 2.46 over k' using its distribution in equation 2.53:

$$\begin{aligned} pr[I_n^{\mathcal{A}_k\mathcal{B}_j} = 1] &= \sum_{k'=max(k+j-m,0)}^{min(k,j)} pr[I_n^{\mathcal{A}_k\mathcal{B}_j} = 1 | K' = k'] P_n(k';k,j) \\ &= P_n(0;k,j) + \sum_{k'=max(k+j-m,1)}^{min(k,j)} (1 - \frac{1}{q^{k-k'}}) P_n(k';k,j) \end{aligned} \quad (2.54)$$

The push of the coded block $x^{\mathcal{A}_k\mathcal{B}_j}$ happens only after peer $\mathcal{A}$ has selected the segment of age n on peer $\mathcal{B}$, $s_n^{\mathcal{B}}$, which implies the segment is servable by $\mathcal{A}$ and thus has more than a blocks on $\mathcal{A}$ at n and $j < m$ blocks on $\mathcal{B}$. Therefore, if we remove the conditioning on k in equation (2.54) given the fact $X_n^{\mathcal{A}} \geq a$, we obtain $\lambda_n(\mathcal{B}_{j<m})$ the probability the sent coded block to $s_n^{\mathcal{B}_{j<m}}$ is useful :

$$\begin{aligned}
\lambda_n(\mathcal{B}_{j<m}) &= \sum_{k=0}^{m} pr[I_n^{\mathcal{A}_k\mathcal{B}_j} = 1 | X_n^{\mathcal{A}} \geq a]\, pr[X_n^{\mathcal{A}} = k] \\
&= \sum_{k=a}^{m} \frac{pr[I_n^{\mathcal{A}_k\mathcal{B}_j} = 1]}{pr[X_n^{\mathcal{A}} \geq a]}\, pr[X_n^{\mathcal{A}} = k] \\
&= \frac{1}{pr[X_n^{\mathcal{A}} \geq a]} \sum_{k=a}^{m} \Bigg[P_n(0; k, j) + \\
&\qquad \sum_{k'=max(k+j-m,1)}^{min(k,j)} (1 - \frac{1}{q^{k-k'}}) P_n(k'; k, j) \Bigg] pr[X_n^{\mathcal{A}} = k] \qquad (2.55)
\end{aligned}$$

Now that we have obtained the formulas for the probability components of β_n we can go back and find $\beta_n(\mathcal{B}_{j<m})$.
β_n depends on whether peer $\mathcal{B}_{j<m}$ is servable or not and on Q_n through α_n as shown in equation 2.26. If peer $\mathcal{B}$ is not servable, it will not be selected by $\mathcal{A}$, which makes β_n equal 0, whatever the value of Q_n is, because α_n is 0:

$$\beta_n(Q_n(\mathcal{B}_{j<m}) = 0, Q_n = k) = 0 \qquad (2.56)$$

When $\mathcal{B}$ is servable by $\mathcal{A}$ at age slot n, then α is given by equation 2.26 that we substitute into the general equation of $\beta_n(\mathcal{B}_{j<m})$, 2.25, to obtain:

$$\begin{aligned}
\beta_n(Q_n(\mathcal{B}_{j<m}) = 1, Q_n = k) &= \gamma_n(\mathcal{B}_{j<m})\, \alpha_n(Q_n(\mathcal{B}_{j<m}) = 1, Q_n = k)\, \lambda_n(\mathcal{B}_{j<m}) \\
&= \gamma_n(\mathcal{B}_{j<m})\, \frac{1}{k+1}\, \lambda_n(\mathcal{B}_{j<m}) \qquad (2.57)
\end{aligned}$$

From equations 2.56 and 2.57, the unconditional probability, $\beta_n(\mathcal{B}_{j<m})$ is then given

by:

$$\begin{aligned}
\beta_n(\mathcal{B}_{j<m}) &= \sum_{i=0}^{1}\sum_{k=0}^{H-1} \beta_n(Q_n(\mathcal{B}_{j<m}) = i, Q_n = k) pr[Q_n(\mathcal{B}_{j<m}) = i, Q_n = k] \\
&= \sum_{i=0}^{1}\sum_{k=0}^{H-1} \beta_n(Q_n(\mathcal{B}_{j<m}) = i, Q_n = k) pr[Q_n(\mathcal{B}_{j<m}) = i] pr[Q_n = k] \\
&= \sum_{k=0}^{H-1} \beta_n(Q_n(\mathcal{B}_{j<m}) = 1, Q_n = k)\theta_n(\mathcal{B}_{j<m}) pr[Q_n = k] + \\
&\qquad \beta_n(Q_n(\mathcal{B}_{j<m}) = 0, Q_n = k) pr[Q_n(\mathcal{B}_{j<m}) = 0] pr[Q_n = k] \\
&= \sum_{k=0}^{H-1} \beta_n(Q_n(\mathcal{B}_{j<m}) = 1, Q_n = k)\theta_n(\mathcal{B}_{j<m}) pr[Q_n = k] \\
&= \sum_{k=0}^{H-1} \gamma_n(\mathcal{B}_{j<m}) \frac{1}{k+1} \lambda_n(\mathcal{B}_{j<m})\, \theta_n(\mathcal{B}_{j<m})\, pr[Q_n = k] \\
&= \gamma_n(\mathcal{B}_{j<m})\, \lambda_n(\mathcal{B}_{j<m})\, \theta_n(\mathcal{B}_{j<m}) \sum_{k=0}^{H-1} \frac{1}{k+1} pr[Q_n = k] \qquad (2.58)
\end{aligned}$$

where in the second equation the variables Q_n and $Q_n(\mathcal{B}_j)$ are assumed to be independent. Then we used equation 2.34 to substitute the value of $pr[Q_n(\mathcal{B}_{j<m}) = 1]$. We notice in the equation of $\beta_n(\mathcal{B}_{j<m})$ that the probability $\theta_n(\mathcal{B}_{j<m})$ cancels with itself when multiplying with $\gamma_n(\mathcal{B}_{j<m})$ as it appears in the denominator of the equations 2.39 and 2.44 of γ_n for the different segment selection strategies. Moreover, the probability $pr[X_n^{\mathcal{A}} \geq a]$ also cancels when multiplying $\lambda_n(\mathcal{B}_{j<m})$ and $\gamma_n(\mathcal{B}_{j<m})$.

2.2.5 Efficiency η

The efficiency is an important performance metric of both p2p file sharing and p2p video streaming systems because it quantifies how good the protocol is at utilizing the most important resource in the system, the upload bandwidth of the peers. Therefore, the efficiency we are referring to here is the upload efficiency, and we will study it in the same sense it was studied in [40]. In [40], they studied the upload efficiency of a BitTorrent-Like system as the probability that a peer has file pieces that are of interest to at least one of its neighbors. Similarly, in our case, we look at the event a peer has ready-to-be-served segments (segments with more than a block(s)) that are still not complete on at least one of its neighbors. In other words, the efficiency at an age slot n, η_n, is the probability a peer has at least one servable neighbor at n. However, although in that case the peer will push a block to one of its servable neighbors, the pushed block may be linearly dependent with the existing blocks of the destination segment on that neighbor, i.e. not useful to that neighbor, which means the time slot was not utilized to send useful information. Therefore, we also need to find the probability the pushed block at age slot n is useful. Consequently, if we slightly change the definition of Q_n in 2.2.4 to include all the servable neighbors a peer has at age slot n, the efficiency at age slot n could expressed as:

$$\eta_n = pr[\text{at least one neighbor is servable at n, the pushed block is useful}] \tag{2.59}$$

$$= pr[Q_n \geq 1]\, pr[\text{the pushed block is useful} \,|\, Q_n \geq 1] \tag{2.60}$$

The first probability could be easily found since the number of servable neighbors a peer has at n has a binomial distribution with parameters $(H,\, \theta_n)$ where θ_n is given in equation 2.30. Thus:

$$\begin{aligned} pr[Q_n \geq 1] &= 1 - pr[Q_n = 0] \\ &= 1 - (1-\theta_n)^H \end{aligned} \tag{2.61}$$

The usefulness of the pushed coded block does not depend on what neighbor is selected because all the neighbors have identical distributions of the number of blocks at n. However, it does depend on what segment is selected or, in other words, what buffer position is selected on that neighbor at the current age slot. Furthermore, the probability distribution to select a segment to serve depends on the segment selection strategy used.

Therefore, if we Let:

$\widehat{\lambda_n}$ $\equiv$ pr[the pushed block is useful | $Q_n \geq 1$].

$I(s_n)$ be the indicator function that the pushed block to the segment of age n, s_n, is useful.

$\gamma(s_n)$ $\equiv$ pr[s_n is selected | $Q_n \geq 1$]

we then can obtain the probability of usefulness as follows:

$$\widehat{\lambda_n} = \sum_{i=0}^{L-2} pr[I(s_{(n \mod E)\,+\,iE}) = 1 \,|\, s_{(n \mod E)\,+\,iE} \text{ is selected}]\gamma(s_{(n \mod E)\,+\,iE}) \quad (2.62)$$

where i is the buffer position of the segment and $(n \bmod E)$ is the time each segment has spent in its current buffer position.

The probability the pushed block to $s_{(n \bmod E)\,+\,iE}$ is useful given it is selected can be obtained from equation 2.55 after averaging over j as follows:

$$\begin{aligned} pr[I(s_n) = 1 \,|\, s_n \text{ is selected}] &= \sum_{j=0}^{m-1} pr[I_n^{\mathcal{AB}_j} = 1 | X_n^{\mathcal{B}} < m, X_n^{\mathcal{B}} = j]\, pr[X_n^{\mathcal{B}} = j] \\ &= \frac{1}{pr[X_n < m]} \sum_{j=0}^{m-1} \lambda_n(\mathcal{B}_{j<m}) pr[X_n = j] \qquad (2.63) \end{aligned}$$

where when $j = m$, we have $pr[I_n^{\mathcal{AB}_j} = 1] = 0$.

For the segment selection probability $\gamma(s_{(n \bmod E)\,+\,iE})$, we already have its formula given in equations 2.44 and 2.39 for both the uniform and most urgent selection strategies respectively. However, the formulas in these equations are conditioned on a) the segment of age n not being complete, i.e. $X_n = j < m$ and b) the downstream neighbor is servable at n. Thus we need to uncondition only over the number of blocks and keep the condition of the neighbor servability since we already know $Q_n \geq 1$ which means a servable neighbor has been selected. As we mentioned earlier, the selected neighbor does not affect the probability of usefulness since all the peers have identical distributions of the number of blocks at any age slot n. Therefore, we can write:

$$\gamma(s_{(n \mod E)\,+\,iE}) = \gamma_{(n \mod E)\,+\,iE}(\mathcal{B}_{j<m})pr[X_{(n \mod E)\,+\,iE} < m] \tag{2.64}$$

where $\gamma_n = 0$ when $X_n = m$. In addition, the probability a peer is servable in the denominator of the equations 2.44 and 2.39 should be replaced with the unconditional version in equation 2.30.

To obtain $\widehat{\lambda_n}$, we substitute equations 2.63 and 2.64 into equation 2.62:

$$\widehat{\lambda_n} = \sum_{i=0}^{L-2} \left\{ \gamma_{(n \bmod E)+iE}(B_{j<m}) \sum_{j=0}^{m-1} \lambda_{(n \bmod E)+iE}(\mathcal{B}_{j<m}) pr[X_{(n \bmod E)+iE} = j] \right\} \tag{2.65}$$

Now we are ready to find the efficiency by substituting equations 2.61, 2.65 into 2.60:

$$\eta_n = \left\{1 - (1-\theta_n)^H\right\} \widehat{\lambda_n} \tag{2.66}$$

The efficiency is a periodic function of n with a period of E age slots (S seconds). This is because, once the system is in steady state, the behavior of the segments in the buffer is the same over any period of segment duration of S seconds (E age slots) and depends only on their time w each has spent in their current positions. Therefore, we can define the average efficiency η over a period of E age slots as:

$$\eta = \frac{1}{E} \sum_{n=0}^{E-1} \eta_n \tag{2.67}$$

where E, as defined earlier, is the number of age slots in S seconds.

Next we find many important performance measures that will help us check the stability of the system. We start with the average effective upload rate peer per $\overline{U_p}$.

During an age slot n, the effective upload rate is $\eta_n U_p$ that we can average over a period of S seconds to get the average rate:

$$\overline{U_p} = \sum_{n=0}^{E-1} \frac{\eta_n U_p}{E} = \eta U_p \tag{2.68}$$

where η is the average efficiency as defined in equation (2.67).
$\overline{U_p}$ represents the average effective bandwidth contribution of a peer in the network. It constitutes along with the average server contribution in equation (2.20) the total effective bandwidth supply $\overline{U}$ in the network:

$$\overline{U} = \overline{U_p} + \overline{U_s} = \eta U_p + \frac{B_s E[X_0]}{S}, \qquad \text{bps} \tag{2.69}$$

Another performance measure of interest is the average number of useful coded blocks sent by a peer in a segment duration. Let it be denoted by $\overline{B}_{\text{sent}}$. It could be found from $\overline{U_p}$ as follows:

$$\begin{aligned}\overline{B}_{\text{sent}} &= \frac{S\overline{U_p}}{B_s} = \frac{S\eta U_p}{B_s} \\ &= \eta E, \qquad \text{blocks/segment}\end{aligned} \tag{2.70}$$

where $E = \frac{S}{T} = \frac{SU_p}{B_s}$.

In order to show the system is stable, we need to obtain the the average effective download rate per peer $\overline{D}$ which could be calculated through the average number of blocks $\overline{B}_{received}$ a peer receives per segment from its neighbors. First, for $\overline{B}_{received}$, the average number of useful blocks received at an age slot n is $H\beta_{n-1}$ since the number of received blocks at n, R_n, has a binomial distribution with parameters (H, β_{n-1}) as shown in equation 2.23. Therefore, $\overline{B}_{received}$ is given by:

$$\overline{B}_{received} = \sum_{n=1}^{N} H\beta_{n-1} \quad \text{blocks/segment} \tag{2.71}$$

where β_n is given in equation 2.58 but conditioned on having j blocks in the s_n^B. Thus we have first to average over j as follows:

$$\beta_n = \sum_{j=0}^{m-1} \beta_n(B_{j<m}) pr[X_n = j] \tag{2.72}$$

Now, the download rate, $\overline{D}$, could be obtained as follows:

$$\overline{D} = \frac{B_s \times \overline{B}_{received}}{S} \quad \text{bps} \tag{2.73}$$

For the system to be stable the following equation must be satisfied:

$$\overline{D} = \overline{U_p} \tag{2.74}$$

Extra bandwidth required for coefficients overhead and redundant blocks

Since the coefficients vector required to generate a coded block is embedded with the coded block and sent to neighbors, each received coded block carries an overhead. It would be interesting to find out how this coefficients overhead impacts the performance. We can obtain the average overhead $\overline{O}$ received from neighbors per segment from $\overline{B}_{received}$ as follows:

$$\overline{O} = \frac{O \times \overline{B}_{\text{received}}}{S} \quad \text{bps} \tag{2.75}$$

where O is the overhead size carried with each coded block and is given in eq. (2.6). We will discuss the influence of the overhead on the performance when we present the numerical results in chapter 3.

Another type of extra data the system suffers from is redundant blocks that we explained in eq. (2.21). The average number of redundant blocks, denoted by $\overline{B}_{Red.}$, could be obtained from the difference between the average number of block received over the segment lifetime and the actual average number of blocks the segment has when it enters the playback position:

$$\overline{B}_{Red.} = \overline{B}_{received} - E[X_N], \qquad \text{blocks/segment} \tag{2.76}$$

The average extra bandwidth wasted on redundant blocks $\overline{U}_{Red.}$ is then given by:

$$\overline{U}_{Red.} = \frac{(B_s - O)\overline{B}_{Red.}}{S}, \qquad \text{bps} \tag{2.77}$$

where we did not include the overhead carried with the redundant blocks because it is already calculated as part of the average total overhead from the peers $\overline{O}$. Therefore, the total average extra bandwidth $\overline{U}_{Red}$ wasted for both the coefficients overhead and redundant blocks is given by:

$$\overline{U}_{Red} = \overline{O} + \overline{U}_{Red.} \tag{2.78}$$

Therefore, in order for the video playback to be continuous on a peer without any interruptions, the total average bandwidth supply coming from neighbors and the server $U_p + \overline{U_s}$ should be greater than the streaming rate at least by $\overline{U}_{Red}$. If the bandwidth supply per peer is not high enough to provide the peer with a full streaming rate and also accommodate the extra wasted bandwidth, the video playback would not be smooth all the time and the user will experience interruptions. We discuss in chapter 3 how to tune up the system parameters to mitigate the effect of the coefficients overhead and redundant blocks.

2.2.6 Releasing the upload bandwidth of the peers: the case of heterogeneous upload bandwidth

Up until this point, we assumed all peers have the same constant upload bandwidth. In this subsection we relax this assumption and treat the case where the upload bandwidth of a peer, U_p, is random, and therefore peers may have different upload speeds.

The heterogeneous upload bandwidth requires changing the definition of the time slot duration T of the system. We defined T previously as the time to upload one block and had all peers run the scheduling algorithm at the same time at the beginning of each time slot. Using the same definition in the heterogeneous case would mean peers may run the scheduling algorithm at different times which complicates the analysis of finding the distribution of the number of blocks at each age slot for a randomly selected peer. Therefore, instead of basing T on the upload bandwidth of the peers, we base it on the block playback time. This means that T is equal to the time required to playback one block of the segment, and thus is given by:

$$T = \frac{S}{m} \; seconds \tag{2.79}$$

It is not necessary to define T in this way and could be defined in any way that makes it the same across all peers. Nonetheless, we should keep in mind that if T is set large enough so that a peer can send many blocks during it, the buffer state of its neighbors may change during T which may result in sending redundant blocks to some segments. In a real-world implementation, each peer chooses the frequency to run the scheduling algorithm separately and can minimize the side effect of sending redundant blocks by having the algorithm schedule one block only. This means the algorithm would run at the fastest frequency possible which would increase the time the peer sits idle, but because the scheduling algorithm in network coding is simple, it would be executed very fast so that the idle time would be very small and would not be a problem compared to the time that may be wasted in sending redundant blocks.

With this new definition of T, the number of age slots in the segment duration,

E, would become equal to m:

$$E = m \tag{2.80}$$

The equations for the rest of the variables in the model such as $N, \ldots$ stay the same.

A problem that results from the new definition of T is that the upload bandwidth U_p in *bps* may not translate to an integer number of blocks per time slot. For example, for the following set of parameters' configuration ($U_p = 775\,kb/s, R_s = 640\,kb/s, m = 50, S = 4\,secs, q = 256$), U_p would be equivalent to 1.2016 $blocks/slot$. If the peer decides to round down that number and send 1 block per slot, it then wastes some time within each time slot.

We can work around this problem by defining the upload bandwidth in $blocks/slot$ like making it take on values in this set $\{1, 2, 3\}$ for example, but this would restrict the bandwidth to certain values like $\{645, 1290, 1935\}\,kb/s$ for the above-mentioned configuration, and remove the ability to test the performance for more fined-grained values of U_p.

To deal with this problem, we suggest that a peer, with the previous configuration of parameters for example, sends either 1 block per slot with probability p or 2 blocks per slot with probability $1 - p$ such that the mean of the sent blocks per slot would equal 1.2016. This means with probability $p \approx 0.8$ the peer sends 1 block per slot and with probability of approximately 0.2 sends 2 blocks per slot.

Next we formalize this idea. Let U_p, defined in terms of *bps*, take on values in this set $\{u_1, u_2, \ldots, u_k\}$ with any given probability distribution $pr[U_p = u_i]$ for $i = 1, \ldots, k$. We can think of u_i's as classes of bandwidth. For example, if we have only u_1 and u_2, they could be though of as peers with home-grade and enterprise-grade internet speeds respectively.

Let $\widehat{B}$ and $\widehat{U_p}$ be the equivalent in real and integer $blocks/slot$ of U_p respectively.

Then they are given by:

$$\widehat{B} = \frac{U_p \times T}{B_s} \; \textit{blocks/slot} \tag{2.81}$$

$$\widehat{U_p} = \begin{cases} \lfloor \widehat{B} \rfloor & \text{with pr. } p, \\ \lceil \widehat{B} \rceil & \text{with pr. } 1-p, \end{cases} \tag{2.82}$$

where T and B_s are given in equations 2.79 and 2.7 respectively, plus the following condition must be satisfied:

$$p\lfloor \widehat{B} \rfloor + (1-p)\lceil \widehat{B} \rceil = \widehat{B} \tag{2.83}$$

from which we can solve for p:

$$p = \frac{\lceil \widehat{B} \rceil - \widehat{B}}{\lceil \widehat{B} \rceil - \lfloor \widehat{B} \rfloor} \tag{2.84}$$

$\widehat{U_p}$ is an non-negative integer random variable and a function of U_p with a conditional pmf given in equation 2.82. It takes on values in this set $\{0, 1, 2, \ldots, \widehat{u}_{max}\}$ where $\widehat{u}_{max} = \lceil max(\widehat{B}) \rceil$, and its unconditional distribution can be obtained by averaging over U_p:

$$pr[\widehat{U_p} = \widehat{u}] = \sum_{i=1}^{k} pr[\widehat{U_p} = \widehat{u} \,|\, U_p = u_i] \, pr[U_p = u_i] \tag{2.85}$$

for $\widehat{u} = 0, 1, 2, \ldots, \widehat{u}_{max}$.

Equation 2.85 represents the distribution of the number of blocks a peer can send per time slot, and this distribution is identical for all peers.

Let $\widehat{R_n}$ be the number of blocks that the segment of age n, $s_n^{B_j}$, on our our randomly selected peer, B_j, receives at age slot n. Next step is to find the distribution of $\widehat{R_n}$. Once that is done, we can plug it in the transition probability matrix in equation 2.21 instead of R_n, and then obtain the distributions of the X_n's.

When a peer has more than one block to send in a time slot, we assume, to make the analysis easier, that the scheduling algorithm is repeated for each block independently, that is, the peer selects uniformly at random a servable neighbor for each block with replacement, then selects randomly, according to the strategy used, a servable segment on the selected neighbor with replacement too for each block. The downside of this kind of scheduling is that it may result in sending multiple blocks to the same segment that could be redundant especially if other peers send blocks to the same segment too. Nonetheless, this assumption allows us to extend the homogeneous model easily to include the heterogeneous upload bandwidth as we will show next.
With this assumption, β_n the probability that $s_n^{B_j}$ receives one block from a neighbor and which was obtained for the homogeneous model in equation 2.58, remains valid in the heterogeneous case. This enables us to define the process of scheduling each block as a Bernoulli trial with success probability of β_n. Therefore, if we denote by $\widehat{R_n^i}$ the number of blocks pushed to $s_n^{B_j}$ by neighbor i with upload bandwidth $\widehat{U_p}$, then $\widehat{R_n^i}$ has a binomial distribution with following parameters:

$$\widehat{R_n^i} \sim B(\widehat{U_p}, \beta_{n-1}) \tag{2.86}$$

Therefore, $\widehat{R_n}$ the number of blocks received to $s_n^{B_j}$ from all neighbor is the sum of $\widehat{R_n^i}$ for i running from 1 to H:

$$\widehat{R_n} = \widehat{R_n^1} + \widehat{R_n^2} + \ldots + \widehat{R_n^H} \tag{2.87}$$

$\widehat{R_n}$ is the sum of iid binomial random variables and thus has a binomial distribution with parameters $(\widehat{H}, \beta_{n-1})$:

$$\begin{aligned} \widehat{R_n} &\sim B(\widehat{H}, \beta_{n-1}) & (2.88)\\ \widehat{H} &= \widehat{U_{p_1}} + \widehat{U_{p_2}} + \ldots + \widehat{U_{p_H}} & (2.89) \end{aligned}$$

where $\widehat{H}$ is the summation of the total number of blocks that can be sent from all the H neighbors of a peer during a time slot.

$\widehat{H}$ has an interesting interpretation. Since neighbors in the homogeneous model has only one block each to schedule per time slot, each block in the heterogeneous model could be thought of as an independent neighbor in the homogeneous model. Therefore, we can describe the effect of releasing the upload bandwidth as only changing the number of neighbors from the constant H to the random $\widehat{H}$ while the rest of the model remains the same. Hence $\widehat{H}$ could be interpreted as the new random number of neighbors that we need to obtain its distribution before we can find the distribution of $\widehat{R_n}$.

we can see that $\widehat{H}$ is the sum of iid random variables with a non-standard distribution given in equation 2.85. Hence its PGF, denoted by $\widehat{H}(z)$, is given by:

$$\widehat{H}(z) = \left[\widehat{U_p}(z)\right]^H \tag{2.90}$$

where $\widehat{U_p}(z)$ is the PGF of $\widehat{U_p}$ that is given by:

$$\begin{aligned}
\widehat{U_p}(z) &= E\left[z^{\widehat{U_p}}\right] \\
&= \sum_{\widehat{u}=0}^{\widehat{u}_{max}} z^{\widehat{u}}\, pr[\widehat{U_p} = \widehat{u}] \\
&= \widehat{p_0} + \widehat{p_1} z^1 + \ldots + \widehat{p}_{\widehat{u}_{max}} z^{\widehat{u}_{max}}
\end{aligned} \tag{2.91}$$

where $\widehat{p_0}, \widehat{p_1}, \ldots, \widehat{p}_{\widehat{u}_{max}}$ are the probabilities of $\widehat{u} = 0, 1, 2, \ldots, \widehat{u}_{max}$ respectively.

Now we find the PGF of $\widehat{R_n}$, denoted by $\widehat{R_n}(z)$, based on the conditional expectation:

$$\begin{aligned}
\widehat{R_n}(z) &= E\left[z^{\widehat{R_n}}\right] \\
&= E\left[E\left[z^{\widehat{R_n}} \,|\, \widehat{H}\right]\right] \\
&= E\left[(1-\beta_{n-1}+\beta_{n-1}z)^{\widehat{H}}\right] \text{ , } \widehat{R_n} \text{ has bino. dist. in 2.89} \\
&= \widehat{H}(z)\Big|_{z=1-\beta_{n-1}+\beta_{n-1}z} \text{ , substitute from eq. 2.90 to get:} \\
&= \left[\widehat{U_p}(1-\beta_{n-1}+\beta_{n-1}z)\right]^H \qquad (2.92) \\
&= \Big[\widehat{p_0}+\widehat{p_1}(\beta'_{n-1}+\beta_{n-1}z)+\widehat{p_2}(\beta'_{n-1}+\beta_{n-1}z)^2+\ldots \\
&\qquad \ldots+\widehat{p}_{\widehat{u}_{max}}(\beta'_{n-1}+\beta_{n-1}z)^{\widehat{u}_{max}}\Big]^H \\
&= \left[\widetilde{p_0}+\widetilde{p_1}z+\widetilde{p_2}z^2+\ldots+\widetilde{p}_{\widehat{u}_{max}}z^{\widehat{u}_{max}}\right]^H \qquad (2.93)
\end{aligned}$$

where $\beta'_{n-1}=1-\beta_{n-1}$ and after expanding and rearranging the sixth equation we have:

$$\begin{aligned}
\widetilde{p_0} &= \sum_{i=0}^{\widehat{u}_{max}} \widehat{p_i}\beta'^{i}_{n-1} \\
\widetilde{p_j} &= \widehat{p_j}\beta^{j}_{n-1}+\sum_{k=j+1}^{\widehat{u}_{max}} k\,\widehat{p_k}\,\beta'^{k-j}_{n-1}\,\beta^{j}_{n-1} \quad \text{for } j=1,2,\ldots,\widehat{u}_{max}
\end{aligned}$$

Equation 2.93 could be expanded using the multinomial theorem:

$$\begin{aligned}
\widehat{R_n}(z) &= \sum_{k_1+k_2+\ldots+k_{\widehat{u}_{max}}=H} \binom{H}{k_1,k_2,\ldots,k_{\widehat{u}_{max}}} \widetilde{p_0}^{k_1}(\widetilde{p_1}z)^{k_2}(\widetilde{p_2}z^2)^{k_3}\ldots \\
&\qquad \ldots(\widetilde{p}_{\widehat{u}_{max}}z^{\widehat{u}_{max}})^{k_{\widehat{u}_{max}}} \\
&= \sum_{k_1+k_2+\ldots+k_{\widehat{u}_{max}}=H} \binom{H}{k_1,k_2,\ldots,k_{\widehat{u}_{max}}} \widetilde{p_0}^{k_1}\widetilde{p_1}^{k_2}\widetilde{p_2}^{k_3}\ldots \\
&\qquad \ldots\widetilde{p}_{\widehat{u}_{max}}^{k_{\widehat{u}_{max}}} z^{k_2+2k_3+\ldots+\widehat{u}_{max}k_{\widehat{u}_{max}}} \\
&= p_{r_0}+p_{r_1}z+p_{r_2}z^2+\ldots+p_{r_{H\widehat{u}_{max}}}z^{H\widehat{u}_{max}} \qquad (2.94)
\end{aligned}$$

where after simplifying the second equation we arrive at equation 2.94 which is the PGF of $\widehat{R_n}$ in standard form, and thus $p_{r_0}, p_{r_1}, p_{r_2}, \ldots, p_{r_{H\widehat{u}_{max}}}$ are the probabilities of $\widehat{R_n}$ assuming values $0, 1, \ldots, H\widehat{u}_{max}$ respectively, and are given by:

$$\begin{aligned}
p_{r_i} &= \sum_{\substack{k_1+k_2+\ldots+k_{\widehat{u}_{max}}=H \\ k_2+2k_3+\ldots+\widehat{u}_{max}k_{\widehat{u}_{max}}=i}} \binom{H}{k_1, k_2, \ldots, k_{\widehat{u}_{max}}} \widetilde{p_0}^{k_1}\widetilde{p_1}^{k_2}\widetilde{p_2}^{k_3}\ldots\widetilde{p}_{\widehat{u}_{max}}^{k_{\widehat{u}_{max}}} \\
&= \sum_{-k_1+k_3+2k_4+\ldots+[\widehat{u}_{max}-1]k_{\widehat{u}_{max}}=i-H} \binom{H}{k_1, k_2, \ldots, k_{\widehat{u}_{max}}} \widetilde{p_0}^{k_1}\widetilde{p_1}^{k_2}\widetilde{p_2}^{k_3}\ldots\widetilde{p}_{\widehat{u}_{max}}^{k_{\widehat{u}_{max}}} \\
&\quad , \text{for } i = 0, 1, 2, \ldots, H\widehat{u}_{max} \qquad (2.95)
\end{aligned}$$

where the subscript of the sum in the second equation results from subtracting the first equality from the second in the subscript of the sum in the first equation.
Equation 2.95 could be solved numerically to obtain the exact pmf of $\widehat{R_n}$.

Another method to obtain the exact pmf of $\widehat{R_n}$ numerically is by obtaining the distribution of $\widehat{H}$ in equation 2.89 using the inverse Discrete Fourier Transform (DFT) method. Since $\widehat{H}$ is a sum of iid random variables, the DFT method could be used to find its exact pmf by inverting its discrete characteristic function. Fortunately, the DFT is implemented efficiently by the Fast Fourier Transform (FFT) algorithms that can carry it out in less operations. After obtaining the pmf of $\widehat{H}$, the distribution of $\widehat{R_n}$ could be found by averaging over $\widehat{H}$:

$$\begin{aligned}
pr[\widehat{R_n} = r] &= \sum_{k=0}^{H\widehat{u}_{max}} pr[\widehat{R_n} = r \mid \widehat{H} = k]\, pr[\widehat{H} = k] \\
&= \sum_{k=0}^{H\widehat{u}_{max}} \binom{k}{r} {\beta_{n-1}}^r (1-\beta_{n-1})^{k-r}\, pr[\widehat{H} = k] \\
&\quad \text{for } r = 0, 1, \ldots, H\widehat{u}_{max} \qquad (2.96)
\end{aligned}$$

where $\widehat{R_n}$ has a binomial distribution with parameters $(\widehat{H}, \beta_{n-1})$.
Now that we have the unconditional distribution of $\widehat{R_n}$ we can, as we mentioned

earlier, plug it in transition probability matrix in equation (2.21) in place of R_n, and then obtain the distributions of the number of blocks at each age slot n, X_n's.

2.2.7 Efficiency for the case of heterogeneous upload bandwidth

Since peers in this case may be able to push multiple coded blocks during a time slot, the definition of the efficiency introduced in equation 2.59 needs to be changed with regard to how many of the pushed blocks are useful.

Let B_n^{useful} be the number of useful blocks out of the $\widehat{U_p}$ coded blocks a peer pushes at age slot n, thus $B_n^{useful} \in \{0, \dots, \widehat{U_p}\}$. The efficiency at age slot n is a function of $\widehat{U_p}$, B_n^{useful}, and Q_n and could be defined as:

$$\eta_n(\widehat{U_p}, B_n^{useful}, Q_n) = \begin{cases} 0 & \text{if } Q_n = 0, \\ \dfrac{B_n^{useful}}{\widehat{U_p}} & \text{if } Q_n \geq 1, \end{cases} \tag{2.97}$$

where when there is at least one servable neighbor ($Q_n \geq 1$), the efficiency would increase when more useful blocks are sent and vice versa.

Next we find the distribution of B_n^{useful} given the peer to which each block is pushed is servable. In fact, in section 2.2.5, we already obtained in equation 2.65 the probability a pushed block is useful given the destination peer is servable $\widehat{\lambda_n}$. Moreover, $\widehat{\lambda_n}$ is the same for each of the $\widehat{U_p}$ blocks sent at n. In addition, these blocks are sent independently as we assumed previously and all are sent at the beginning of the time slot and received at the end of the time slot, thus the usefulness of one does not affect the usefulness of the others. Therefore, B_n^{useful} has a binomial distribution with parameters $(\widehat{U_p}, \widehat{\lambda_n})$.

Now the unconditional efficiency could be found by averaging eq. (2.97) as follows:

$$\begin{aligned}\eta_n &= \sum_{\hat{u}=0}^{\hat{u}_{max}} \sum_{k=0}^{\hat{u}} \frac{k}{\hat{u}} pr[B_n^{useful} = k | \widehat{U_p} = \hat{u}] pr[\widehat{U_p} = \hat{u}] pr[Q_n \geq 1] \\ &= \left(1 - (1-\theta_n)^H\right) \sum_{\hat{u}=0}^{\hat{u}_{max}} \frac{E[B_n^{useful} | \widehat{U_p} = \hat{u}]}{\hat{u}} pr[\widehat{U_p} = \hat{u}] \\ &= \left(1 - (1-\theta_n)^H\right) \sum_{\hat{u}=0}^{\hat{u}_{max}} \frac{\hat{u}\widehat{\lambda_n}}{\hat{u}} pr[\widehat{U_p} = \hat{u}] \\ &= \left(1 - (1-\theta_n)^H\right) \widehat{\lambda_n} \end{aligned} \tag{2.98}$$

where in the first equation Q_n and $\widehat{U_p}$ are independent and the condition $Q_n \geq 1$ is already taken care of when finding the distribution of B_n^{useful}. In addition, the conditional expectation is equal to $\hat{u}\widehat{\lambda_n}$ since B_n^{useful} has a binomial distribution.

We notice that the efficiency is identical to the efficiency for the homogeneous upload bandwidth in eq. (2.66) due to binomial distribution of B_n^{useful}. As we mentioned in the homogeneous case, the efficiency here is also periodic with a period equal to a segment duration of S seconds. The average efficiency η over any period of S seconds (or m age slots) is given by:

$$\eta = \sum_{n=0}^{m-1} \frac{\eta_n}{m} \tag{2.99}$$

Next we calculate the measures that help us investigate the stability of the system. First, we start with the average effective peer upload rate, $\overline{U_p}$.

Since η_n is the ratio of useful blocks sent during age slot n, then the average amount of useful information that is sent during time slot T is $E[U_p]T\eta_n$ kb/s. Therefore, $\overline{U_p}$ could be calculated as the sum of useful information sent over all age slots in a segment duration divided by the segment duration:

$$\begin{aligned}\overline{U_p} &= \frac{\displaystyle\sum_{n=0}^{m-1} E[U_p]T\eta_n}{S} \\ &= E[U_p]\frac{\sum_{n=0}^{m-1}\eta_n}{m} \\ &= E[U_p]\eta \qquad \text{b/s} \end{aligned} \tag{2.100}$$

where η is the average efficiency and $\frac{T}{S} = \frac{1}{m}$.

To verify the stability of the system, we find the average download rate from peers $\overline{D}$. First, we calculate the average number of useful blocks $\overline{B}_{received}$ a peer receives for a segment from its neighbors. $\overline{B}_{received}$ could be calculated by finding $E[\widehat{R_n}]$, the average number of useful blocks received at age slot n, then summing it over all the age slots of the servable-by-peers segment lifetime:

$$\begin{aligned}\overline{B}_{received} &= \sum_{n=1}^{N} E[\widehat{R_n}] \\ &= \sum_{n=1}^{N} E[E[\widehat{R_n}|\widehat{H}]] \\ &= \sum_{n=1}^{N} E[\widehat{H}\beta_{n-1}] \quad , \widehat{R_n} \text{ has binomial dist. w/ parameters in 2.89} \\ &= \sum_{n=1}^{N} \beta_{n-1}E[\widehat{H}] \\ &= \sum_{n=1}^{N} \beta_{n-1}HE[\widehat{U_p}] \quad , \widehat{H} \text{ is sum of } H \text{ iid } U_p\text{'s in 2.89} \\ &= HE[\widehat{U_p}]\sum_{n=1}^{N}\beta_{n-1} \qquad \text{Blocks/Segment}\end{aligned} \tag{2.101}$$

where β_n is unconditioned in equation (2.72).

Now we can calculate the average download rate from neighbors, $\overline{D}$, as follows:

$$\begin{aligned}\overline{D} &= \frac{\overline{B}_{received} B_s}{S} \\ &= \frac{B_s H E[\widehat{U_p}] \sum_{n=1}^{N} \beta_{n-1}}{S} \qquad b/s \end{aligned} \tag{2.102}$$

For the system to be stable, the average download rate should be equal to the average upload rate, thus the following equation should be satisfied:

$$\overline{D} = \overline{U_p} \tag{2.103}$$

The overhead rate $\overline{O}$ and average number of redundant blocks $\overline{B}_{Red.}$ received per segment could be obtained in the same fashion as in the homogeneous case.

The total average bandwidth supply consists of: *i*) the average peers contribution of $E[U_p]$, and *ii*) the average server contribution $\overline{U_s}$. For a smooth playback, the bandwidth supply should be higher enough to accommodate the streaming rate and the extra bandwidth wasted on the coefficients overhead and useless and redundant blocks.

Chapter 3

Numerical Results and Analysis of the Protocol Performance

In the previous chapter, we developed a stochastic analytical model to reveal the probability distribution of the number of blocks a segment accumulates at each age throughout its lifetime in the buffer for both the Uniform and Most-Urgent segment selection strategies. There is no closed-form expression for the model, and therefore we solve the model numerically to obtain the performance metrics and present the results in this chapter. Specifically, we study the effects of the main system parameters of network coding and video streaming on the continuity and efficiency performance. This would give us insights into what parameters are most critical to the performance and what parameters are not that critical. Furthermore, we would also understand the main trade-offs that should be made to obtain the best performance under a specific configuration of the parameters. We will also compare the continuity performance of the two segment selection strategies and find out which one performs better. Moreover, we will compare the theoretical results to the simulation results to validate the model. As we will see later, both the model and simulation exhibit the same behavior for both strategies for different streaming scenarios.

First of all, we define a standard configuration of the system parameters as shown in table 3.1 with two scenarios for $E[U_p]$; the mean peer upload bandwidth: *i)* $E[U_p] =$ 653 kb/s which is equivalent to 1.0203 times higher than the streaming rate i.e. $\frac{E[U_p]}{R_s} = 1.0203$, *ii)* $E[U_p]$ is equal to the streaming rate. After that, we present the performance graph for this configuration in figure 3.1 and explain how to read the graph. In the rest of the chapter, we investigate the effect of most of the parameters under both the uniform (Uni.) and most urgent (M.U.) segment selection strategies with the scenario of $E[U_p]$ for which the effect is most noticeable. We vary only the value of the studied parameter while the rest of the standard configuration remains the same.

Parameter	**Value**
Z	100 *peers*
U_s	1 Mb/s
$i)E[U_p]$	653 kb/s $\equiv$ 1.0124 blocks/slot
$ii)E[U_p]$	640 Kb/s $\equiv$ 0.9922 blocks/slot
R_s	640 Kb/s
H	10 *neighbors*
L	4 *positions*
S	4 *seconds*
m	50 *blocks*
a	1 *block*
q	$2^8 = 256$

Table 3.1: Standard parameter configuration

The lines in Figure 3.1 represent the probability of a complete segment $pr[X_n = m]$ as a function of the age slot n for both strategies. The age range in the graph covers only the part of the segment lifetime during which it is servable by peers, or, in other words, the range does not include the time spent in the first or last buffer positions. The reason is that the segment, during its time spent in the first position, will not have any blocks on all peers until it reaches the end of its stay in the first position where it will have received a linearly independent blocks from the server but only on the peers the server selected. Thus, $pr[X_n = m]$ would be equal to 0 throughout the

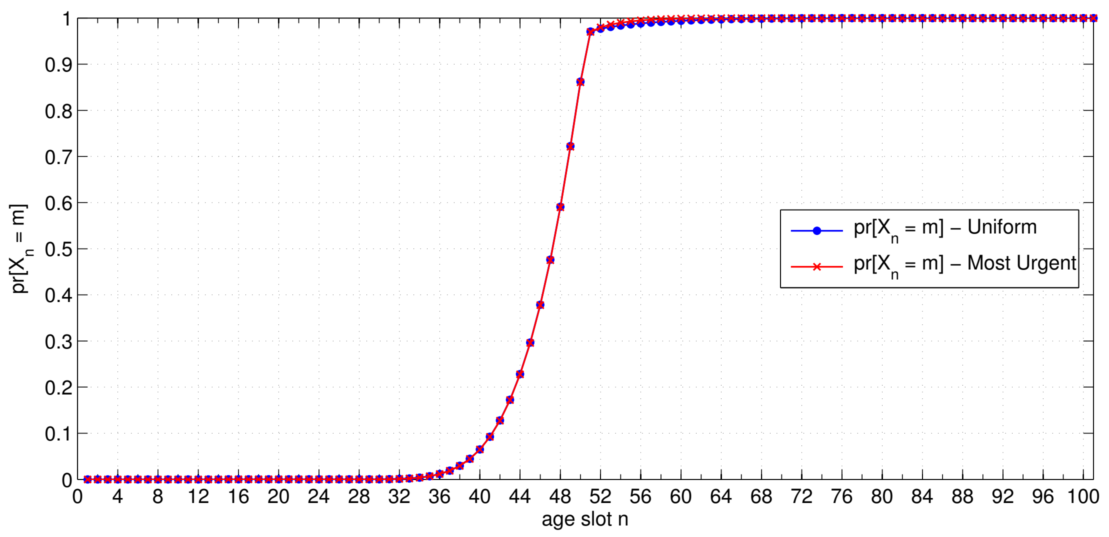


Figure 3.1: Performance of both strategies for the standard configuration when $\frac{E[U_p]}{R_s} = 1.0203$

first position, and it suffices to only show $pr[X_n = m]$ at the beginning of the stay in the second position shown as age slot $n = 0$ in the graph. For the last position, the segment will not receive any blocks either once it enters this position and will be either played back or skipped depending on the number of blocks it had at the end of its stay in the previous position.

Therefore, $pr[X_n = m]$ at the end of the second to last position i.e. at age slot $n = N$, or $n = 100$ in figure 3.1, is the probability of continuity $P_{cont.}$. Moreover, figure 3.1 reveals how the segment develops, in terms of how likely it will be complete, over the time it spends in each buffer position. For example, since $E = m = 50$ in the standard configuration, the segment is highly unlikely to be complete while in the second position that corresponds to the range $n = 0, \ldots, 49$ in the graph, whereas it is certainly complete during most of its time in the third position that corresponds to the range $n = 50, \ldots, 99$ and enters the playback position with probability of continuity $pr[X_{100} = m] \approx 1$ for both strategies.

Other quantities of interest could also be plotted against the segment age such as the efficiency and the average number of blocks the segment has at each age slot, as we will show later.

Next we investigate the effects of the system parameters and start with the most important parameter; the upload bandwidth U_p.

3.1 Video streaming parameters

3.1.1 Peer upload bandwidth U_p

The expected value $E[U_p]$ of U_p is the mean upload bandwidth available to a peer through its neighbors. $E[U_p]$ plus the average server contribution per peer $\frac{B_s E[X_0]}{S}$ constitutes the bandwidth supply per peer. The server contribution per peer is 9.933 kb/s for the standard configuration which is equivalent to only 0.0155 of R_s. This would make the bandwidth supply per peer be 1.0155 and 1.0358 of the streaming rate or 649.933 kb/s and 662.933 kb/s for the two scenarios $E[U_p] = R_s$ and $\frac{E[U_p]}{R_s} = 1.0203$ respectively.

We have observed that the performance depends only on the mean of U_p such that different sets of values and distributions of U_p that have the same mean $E[U_p]$ would lead to almost the same performance. We state the reason for this later. Therefore, we will not show the figures with different distributions of U_p and will only use $E[U_p]$ for the following figures.

We plot in Figure 3.2 the probability of continuity $P_{Cont.}$ as a function of the difference of the mean upload bandwidth and the streaming rate $E[U_p] - R_s$ to show the lowest $E[U_p]$ from which upward perfect continuity is achieved.

As we see in Figure 3.2, with an average server contribution of only about 9.9 kb/s per peer, the rest of the bandwidth supply coming from the neighbors i.e. $E[U_p]$ is sufficient to be as low as the streaming rate i.e. $E[U_p] = R_s$ for the uniform strategy to start achieving almost perfect continuity for both the model and the simulation. This is because the bandwidth supply per peer, which is 649.9 kb/s, is sufficient to

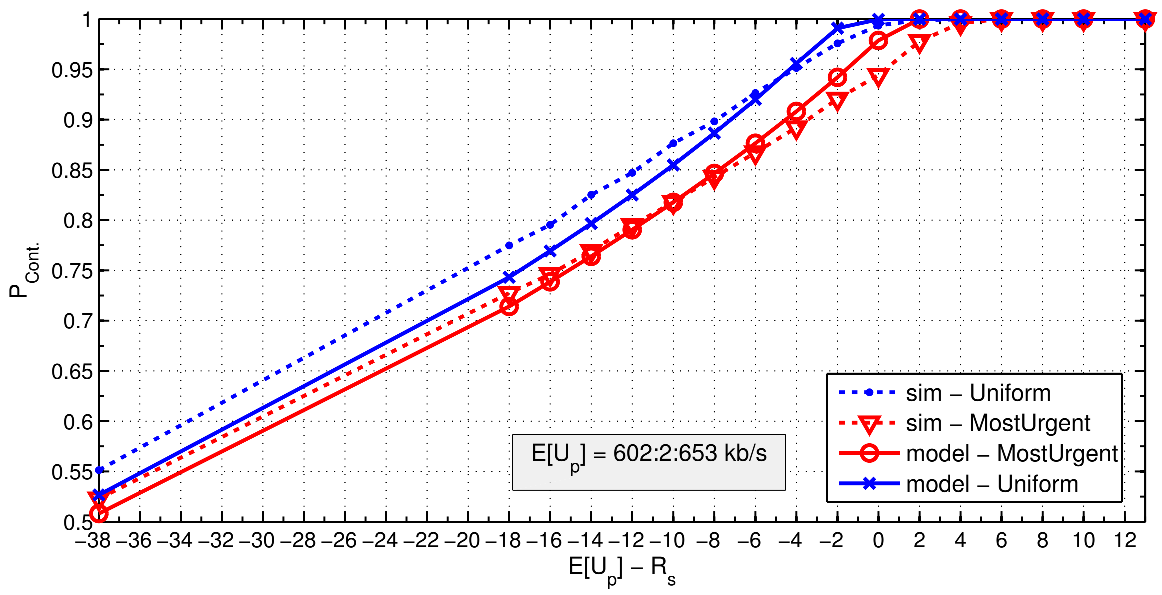


Figure 3.2: Continuity performance as a function of $E[U_p] - R_s$ for the standard configuration for both strategies.

accommodate the extra bandwidth wasted by the coefficients overhead and redundant blocks, that we calculated to be 9.87 kb/s, and provide a streaming rate of 640 kb/s to the peer.

On the other hand, the most urgent strategy requires $E[U_p]$ to be 2 and 4 kb/s above R_s for the model and the simulation respectively to start achieving continuity close to 1. This is caused by the extra bandwidth wasted per peer which is 10.67 kb/s that makes a streaming rate of 640 kb/s not possible when $E[U_p] = R_s$.

Figure 3.2 also shows that, for the uniform strategy the simulation matches the model nicely when $E[U_p]$ is greater than R_s while it starts to deviate just a little bit from the theoretical results when $E[U_p]$ is lower than R_s which is anyway not a case of interest. As for the most urgent strategy, the deviation is most noticeable when $E[U_p]$ is within 2 kb/s around R_s but still less than 0.05.

While this small difference between the simulation and the model could be in general attributed to the independence assumptions we made in the model, the most noticeable difference around the streaming rate, especially for the most urgent strategy, could be further explained.

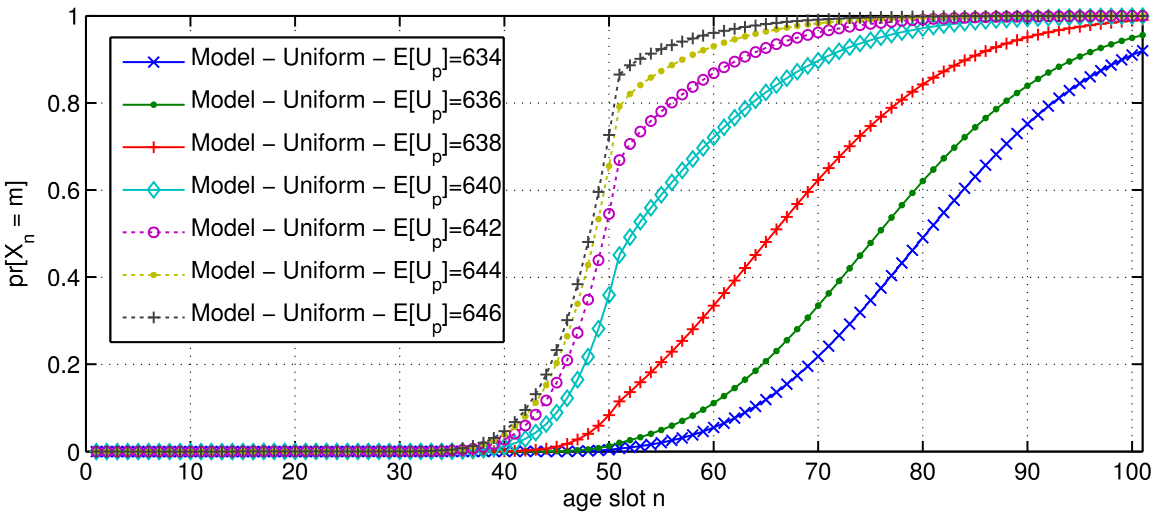
We have observed that the performance is very sensitive to small changes in the mean upload bandwidth when the average bandwidth supply is close to the demand while it becomes much less sensitive when the supply is greater than the demand by an enough margin. Furthermore, the uniform strategy is less sensitive than the most urgent strategy to these kind of changes. This sensitivity may make any errors that might have been made in the model or the simulation have greater effect when the demand is close to the supply than what they would have otherwise which may explain the small difference in their performance when the bandwidth supply is close to the demand.

Figure 3.3 shows this behavior for both the most urgent and uniform strategy respectively. We see in this figure 3.3b that increasing the mean upload bandwidth when it is equal to $R_s = 640\,kb/s$ by two successive increments of only 1 kb/s leads to relatively big jumps in the performance of the most urgent strategy compared to much smaller improvements when $E[U_p]$ is increased from 644 kb/s to 650 kb/s in increments of 2 kb/s.

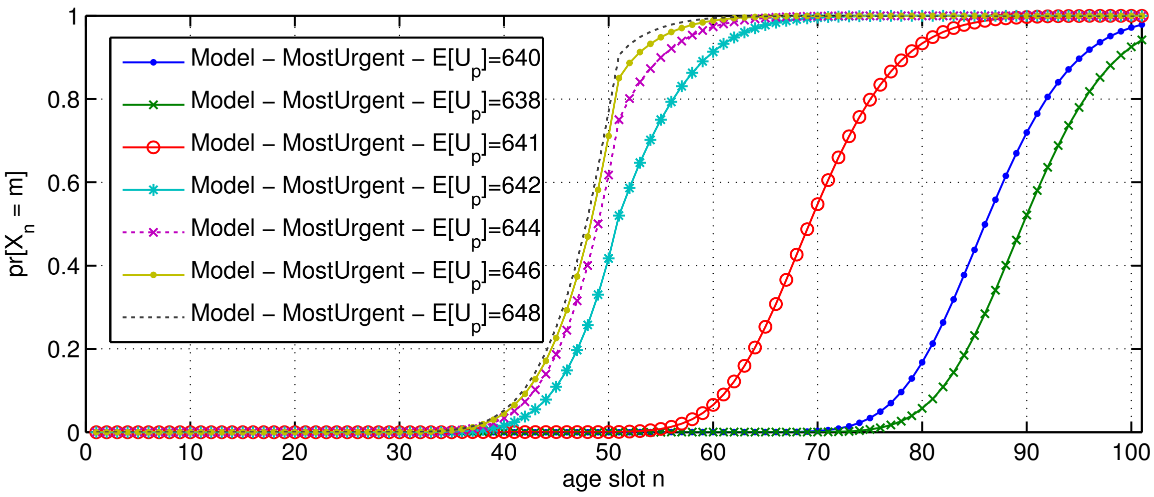
Compared to the most urgent strategy, the uniform strategy is less sensitive to small changes in $E[U_p]$ around R_s as shown in figure 3.3a. Moreover, the significant changes in the performance happen when $E[U_p]$ is less than $R_s = 640\,kb/s$ by a small margin. As we see in the figure 3.3a, performance drops more significantly when $E[U_p]$ is lower than R_s by 2 and 4 kb/s whereas much less performance drop happen when $E[U_p]$ drops from 646 to 640 kb/s in decrements of 2 kb/s.

3.1.2 Comparison of the Uniform and Most Urgent strategies

The most interesting observation in Figure 3.2 is that both the model and simulation show that the uniform strategy (Uni.) performs better than the most urgent (M.U.) strategy when $E[U_p]$ is less or slightly higher than R_s while both strategies are able



(a) Uniform



(b) Most Urgent

Figure 3.3: Sensitivity of the performance of both strategies when the bandwidth supply is close to the demand

to achieve a continuity of 1 when $E[U_p]$ is even slightly greater than R_s. Having said that, the difference in the continuity performance shown in figure 3.2 is less than 0.05 which seems to be not that significant. To reveal more about how the segment develops over its lifetime in the buffer, we present Figure 3.4 that shows the probability that the segment is complete as a function of its age when $E[U_p] = R_s$. Both the model and simulation in this figure clearly show that a segment in the Uni. strategy is much more likely to be complete at an earlier age compared to the M.U.

strategy. Furthermore, the Uni. strategy is able to achieve almost perfect continuity even when $E[U_p]$ is as low as R_s.

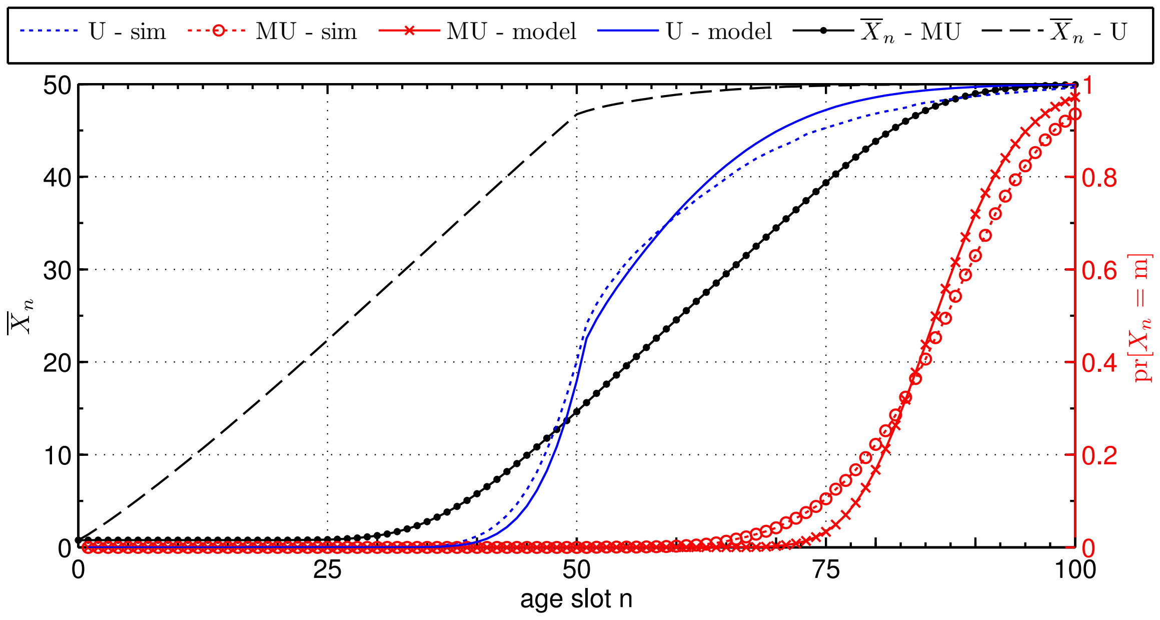


Figure 3.4: Comparison of performance (red & blue lines) on the right axis and average number of blocks (black lines) on the left axis of both strategies when $E[U_p] = R_s$

This may seem counter intuitive for the first glance as we would expect the M.U. strategy to perform better since the focus would be on the segment closest to the playback deadline. However, we can explain this behavior by realizing that in the Uni. strategy servable segments in the buffer have equal chances to receive coded blocks regardless of their positions. This makes the segment start accumulating blocks early in its lifetime and essentially gives it more time to become complete before it reaches its playback deadline.

By contrast, the segment in the M.U. strategy starts to receive blocks only when it is the closest to its playback deadline among the other servable segments which gives it less time to become complete. Moreover, if the most urgent segment has received a large number of its blocks, then blocks received from neighbors are more likely to be linearly dependent since the existing blocks of the segment would span a larger subspace. Thus focusing the efforts of all neighbors on this segment would make the

M.U. strategy waste a little bit more bandwidth than the Uni. strategy. In fact, we have found in our standard config. with $E[U_p] = R_s$ that the extra bandwidth to be around 9.87 and 10.67 kb/s for the Uni. and M.U. strategies respectively. Therefore, when the bandwidth supply is not high enough to make the most urgent segment complete before its playback deadline, the segment would start to receive blocks only when it enters the second to last buffer position by which time it is too late to get complete before the deadline. However, if the the bandwidth supply gets higher enough than the streaming rate, the segment starts to receive blocks earlier and the performance of the M.U. strategy will be similar to the Uni. strategy.

To help us see this behavior we plot the average number of blocks $\overline{X} = E[X_n]$ a segment has at each age slot for both strategies when $E[U_p] = R_s$ in the same Figure 3.4. We clearly notice that a segment in the Uni. strategy starts receiving blocks as soon as it becomes servable at the beginning of its stay in the first position and enters the second position with more than 90% of its blocks received. On the other hand, a segment in the M.U. strategy starts to accumulate blocks only at the end of the first position and enters the next position with only just under 30% of its blocks received.

Performance sensitivity to the mean upload bandwidth $E[U_p]$

As we mentioned earlier, the performance of both strategies depends only on the mean of the peer upload bandwidth U_p and not on the specific set of values and the corresponding distributions as long as these distributions have the same mean. Figure 3.5 shows that each of the strategies has exactly the same performance for the set of values $\{510, 620, 650, 700, 800\}$ and $\{600, 640, 650, 700\}$ of U_p with probability distributions of $\{.2, .3, .1, .2, .2\}$ and $\{.1, .2, .5, .2\}$ respectively with the mean of 653 kb/s for both distributions. Same thing happens when the mean is 640 kb/s.

This behavior is caused by the assumption we made when we studied the case of

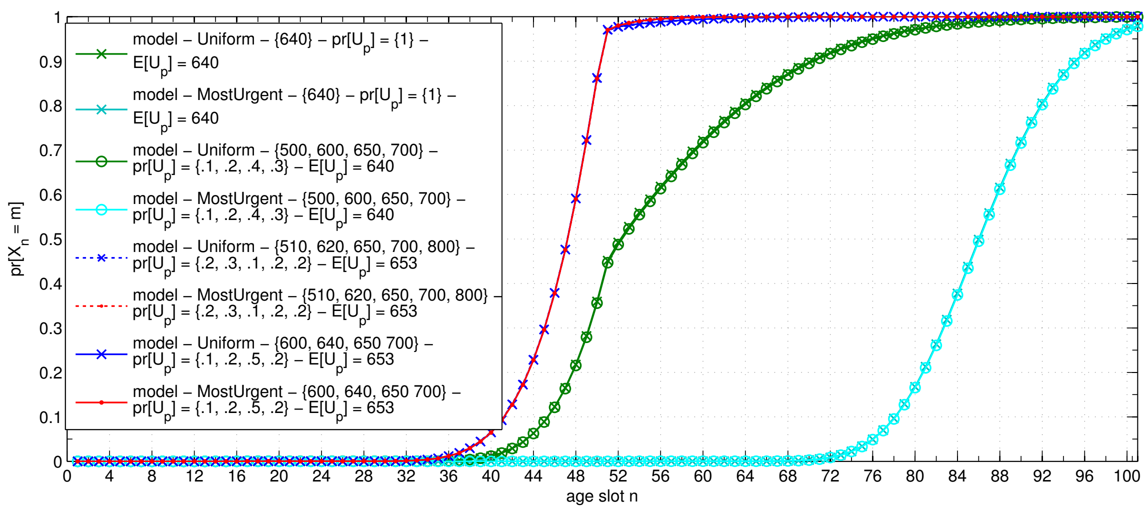
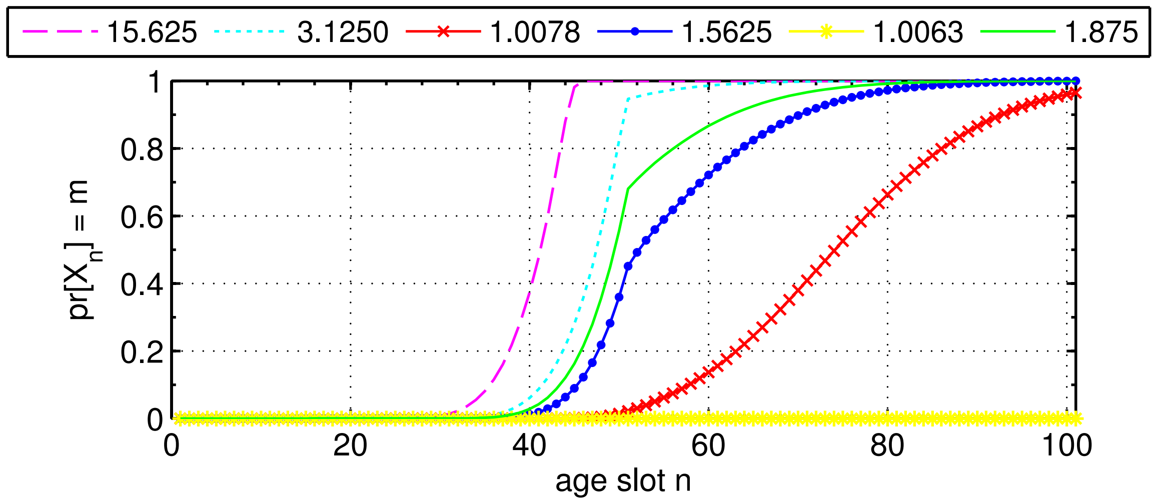


Figure 3.5: Performance of each strategy is identical for distributions of U_p that have the same mean $E[U_p]$ when the value of U_p may change at each time slot.

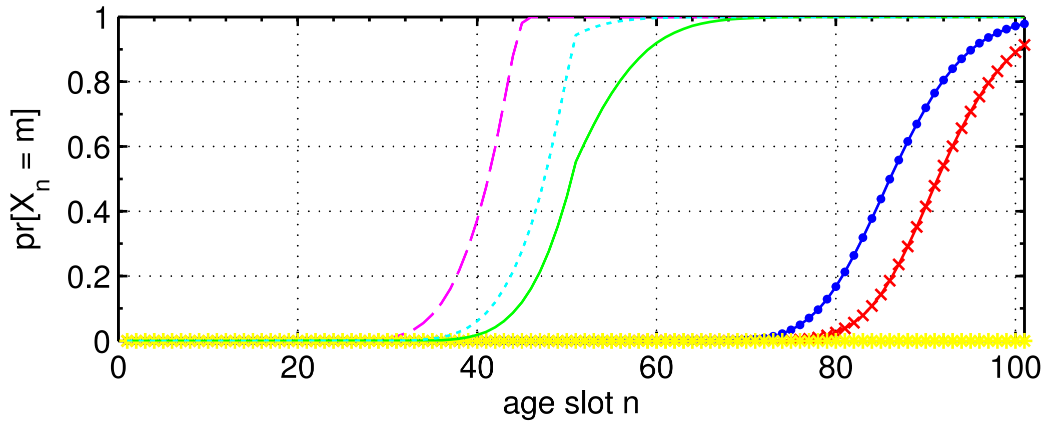
heterogeneous upload bandwidth. We assumed there that peers select the value of U_p randomly at each time slot according to its giving distribution. This is in contrast to what may happen in the real world where a peer would have a random upload speed when it first joins the overlay but its speed remains constant throughout the streaming session. We made this assumption to make modeling the system easier as peers would be identical in this case. Nonetheless, it could still have some realistic interpretation. We can argue that having the upload bandwidth remain constant throughout the streaming session may not be that realistic as the bandwidth may fluctuate because of the user behavior on his computer or the network congestion. In this sense changing the bandwidth at each age slot may reflect fluctuations in the available bandwidth.

3.1.3 Server Capacity U_s

For our standard configuration when $E[U_p] = R_s$, the average server contribution per peer has to be at least 6.45 kb/s, equivalent to a server capacity of 1.0078 times larger than the streaming rate (i.e. $U_s = 645$ kb/s), for the server to be able to push all the



(a) Uniform Strategy



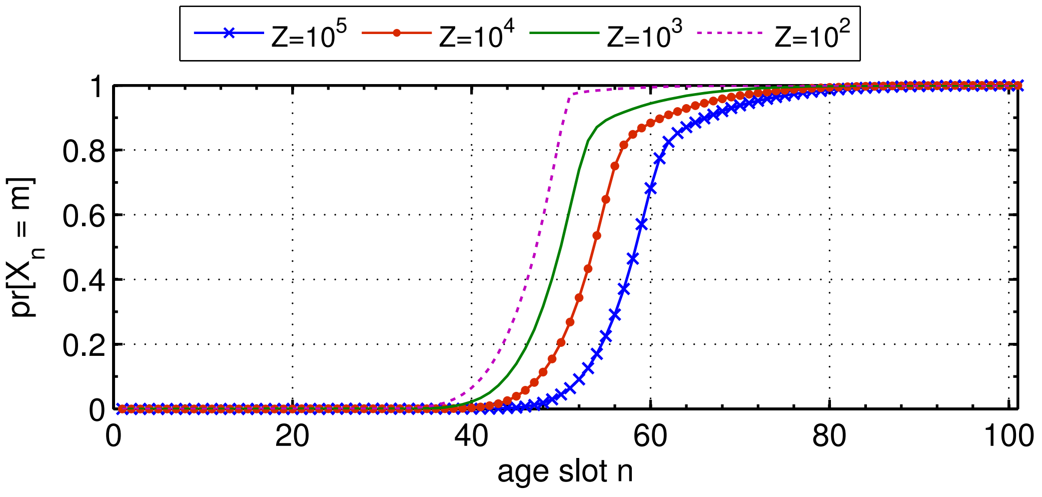
(b) Most Urgent Strategy

Figure 3.6: Performance w.r.t. the server capacity normalized by the streaming rate $\frac{U_s}{R_s}$ when $E[U_p] = R_s$

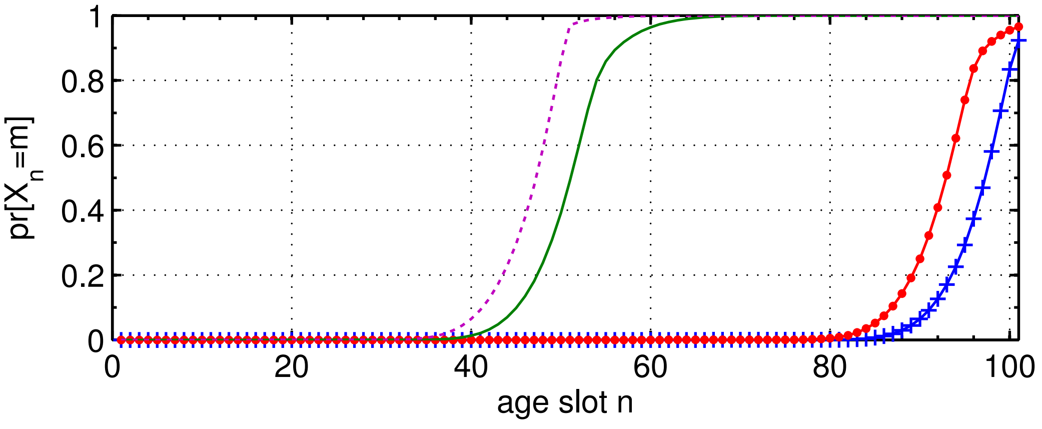
m blocks of the segment to the peers as shown in Figure 3.6. If the server capacity drops below that, the system would fail regardless of $E[U_p]$. Moreover, when the mean peer bandwidth supply is close to the streaming rate the uniform strategy would require less server bandwidth than the most urgent to achieve a continuity higher than 99.99%. This is again because the uniform strategy takes a better advantage of the buffering time. We have found that when $E[U_p] = R_s$ increasing U_s from 1 Mb/s to just 1.2 Mb/s (1.875 times of R_s) would be sufficient for the most urgent strategy to catch up with the uniform strategy and achieve a continuity higher than 99.99%. Finally, when the bandwidth supply becomes higher enough than the demand, both strategies would exhibit the same performance.

3.1.4 Overlay Size Z

As shown in Figure 3.7, the uniform strategy can scale up to 10^5 peers with a relatively small decline in the performance even though the bandwidth supply exceeds the demand by only about 23 kb/s ($E[U_p] - R_s = 13$ kb/s + 9.9 kb/s server contribution). On the other hand, the most urgent strategy would witness a much more dramatic decline in the performance starting from a smaller scale of 10^4 users with



(a) Uniform Strategy



(b) Most Urgent Strategy

Figure 3.7: Effect of the populations size Z on the performance of both strategies when $\frac{E[U_p]}{R_s} = 1.0203$

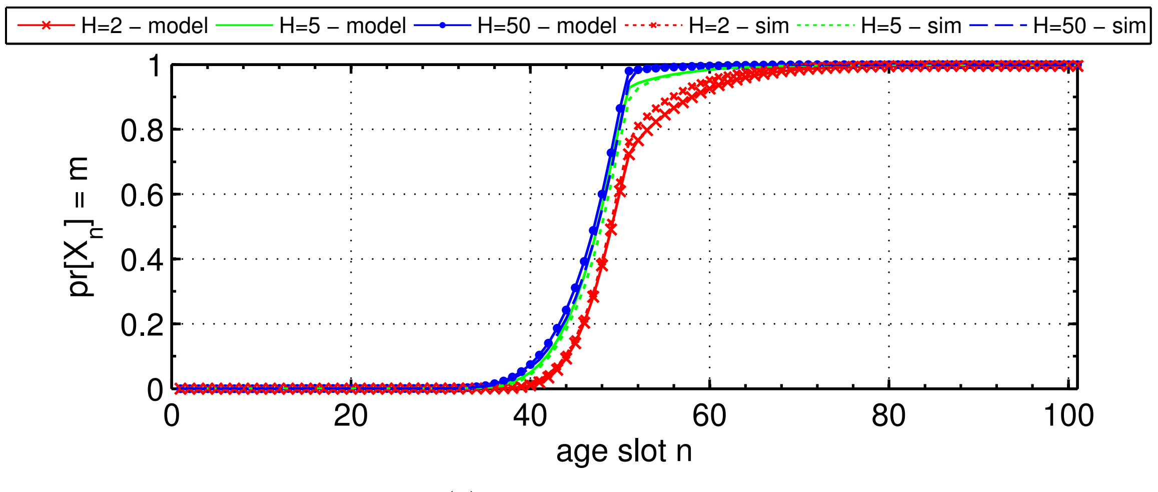
the continuity dropping below 1. This is because the server contribution per peer would drop as Z increases and there would be much fewer peers able to serve the segment at age slot 0 due to our server design. We have found that in order for the most urgent strategy to achieve a continuity higher than 99.99% when the scale is $Z = 10^5$, it would require either the server capacity to be increased 40 times from 1 Mb/s to around 40 Mb/s, or the mean peer upload speed to be increased by only 12 kb/s from 653 to 665 kb/s.

3.1.5 Number of Neighbors H

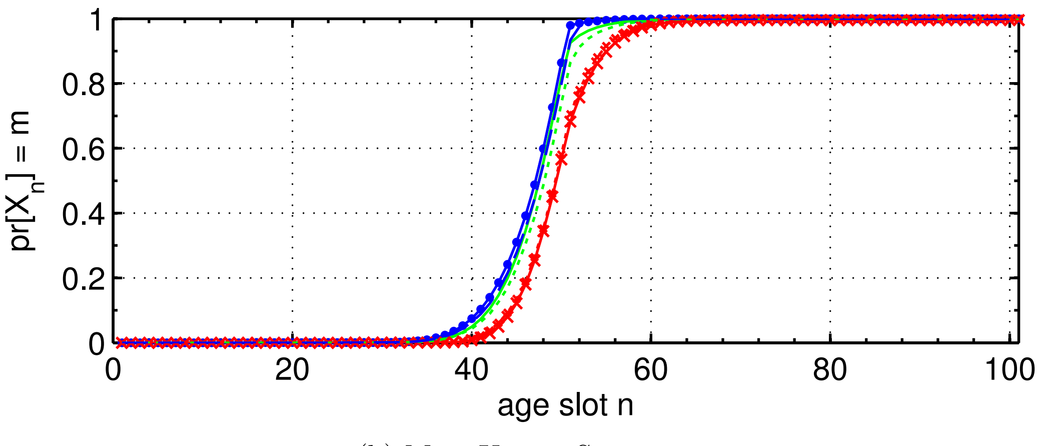
We have found that increasing the number of neighbors would lead to a better performance for both strategies up to a certain value after which having more neighbors would not yield any more gains in the performance as shown in Figure 3.8. In the standard configuration when the bandwidth supply exceeds the demand, we found that maintaining 10 neighbors is sufficient to achieve the best performance. While maintaining a very small number of neighbors would also give a good performance, it would not be resilient to peer churn. On the other hand, having a large number of neighbors such as 50 or more would increase the overhead required to maintain them.

3.1.6 Buffer Length L

Increasing the buffering time by increasing the buffer positions would not help the most urgent strategy improve its performance when the bandwidth supply is close to the demand as shown in Figure 3.9. As we explained earlier, the bandwidth supply is not high enough keeping the focus only on the segment in the second to last position where there would not be enough time for the segment to turn complete. On the other hand, the uniform strategy would benefit from increasing the buffering time as shown in Figure 3.9 when increasing L from 3, where there is only 1 servable-by-peers



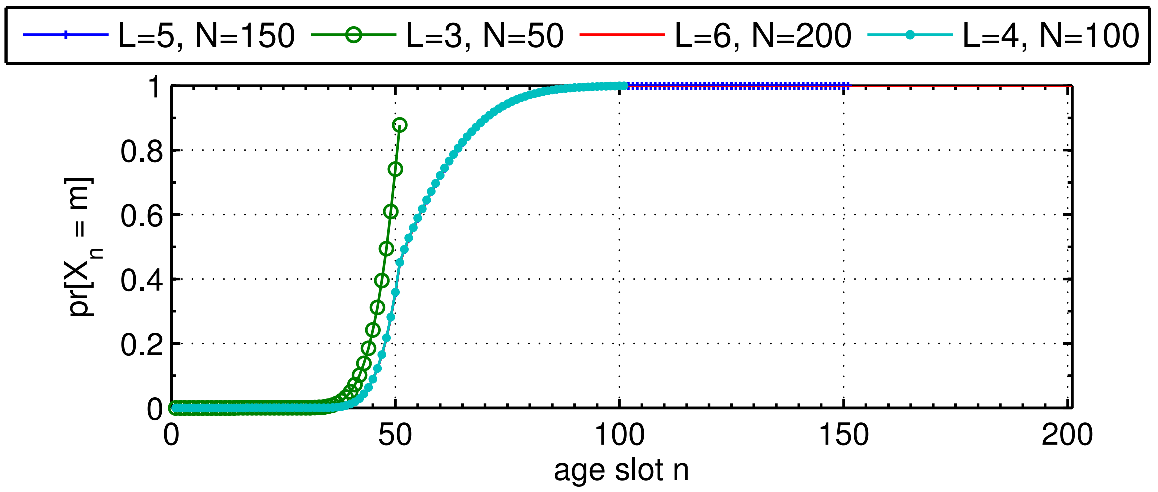
(a) Uniform Strategy



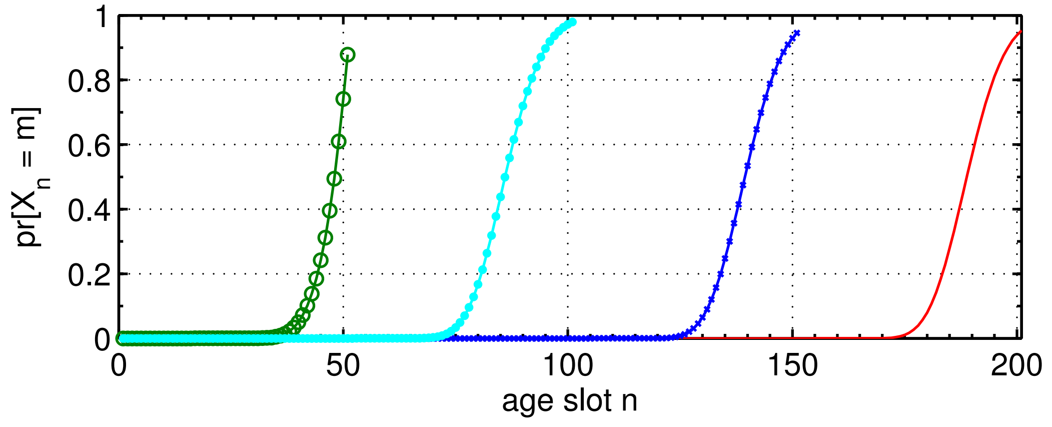
(b) Most Urgent Strategy

Figure 3.8: Effect of the number of neighbors H on the performance of both strategies when $\frac{E[U_p]}{R_s} = 1.0203$

position, to 4 where the segment gets complete around the middle of its stay in the third position. If the segment turns complete before its playback deadline, increasing L would not change the performance and would just increase the delay before the playback.



(a) Uniform Strategy



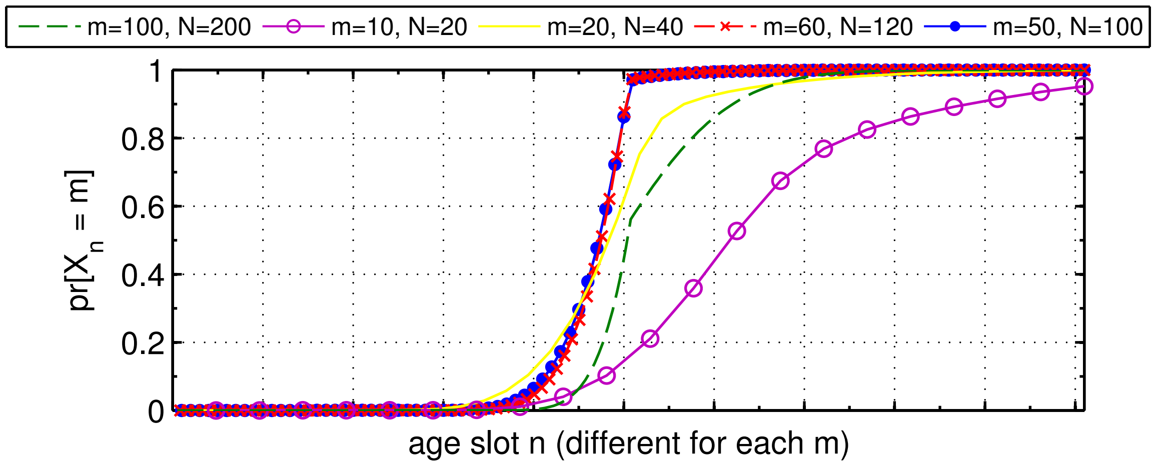
(b) Most Urgent Strategy

Figure 3.9: Buffer Length L effect on the performance of both strategies when $E[U_p] = R_s$

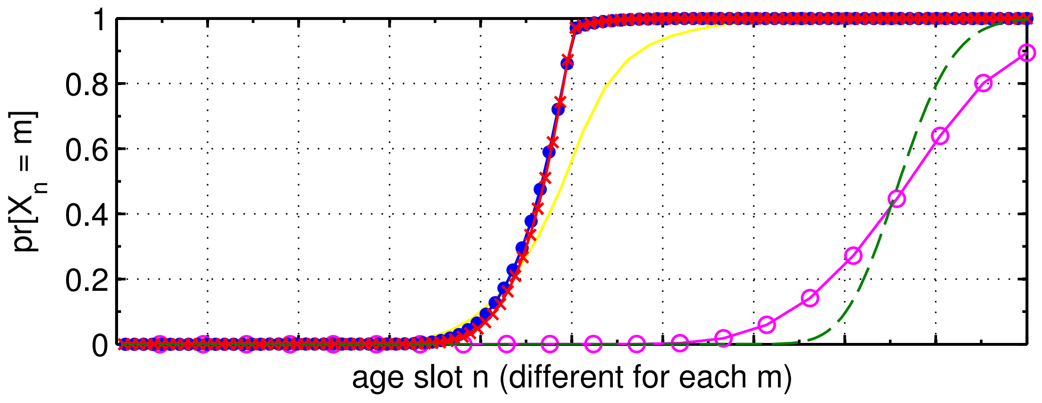
3.2 Network coding parameters

3.2.1 Effect of the number of blocks m

The number of blocks the segment is divided into is one of the most important parameters in the network coding p2p streaming protocols and plays a significant role in determining the performance of the protocol. It is directly related to the block size as shown in (2.7) such that increasing m leads to smaller block sizes and vice verse. Figure 3.10 shows that increasing m leads to a better performance for both strategies up until a certain value after which the performance starts to degrade. This can be explained as, on one hand, as m gets smaller for a given segment duration, the block size gets larger making the amount of bandwidth wasted due to linear dependency become much larger than what it would be for smaller block sizes. On the other hand, increasing m after a certain value would kick in the effect of the coefficients overhead that would get larger with larger m as shown in (2.6). Therefore, there is a sweet spot for m per S for which the extra bandwidth for redundant blocks and overhead is minimal. For our standard configuration, we have found the extra bandwidth required ranges from 35.7 kb/s for $m = 10$ to around 23 kb/s for $m = 100$ with the lowest 19.4 kb/s happening at $m = 50$ at which we got the best performance.

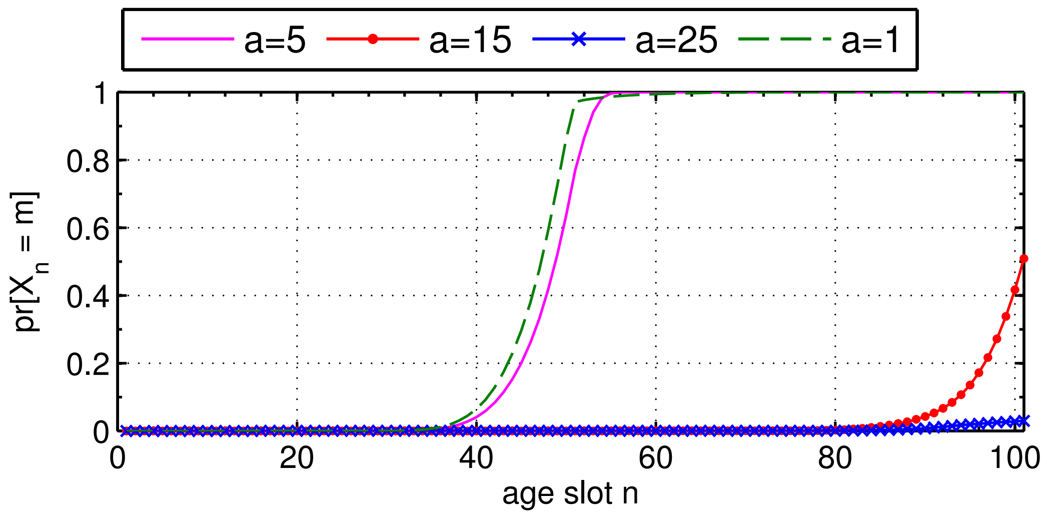


(a) Uniform

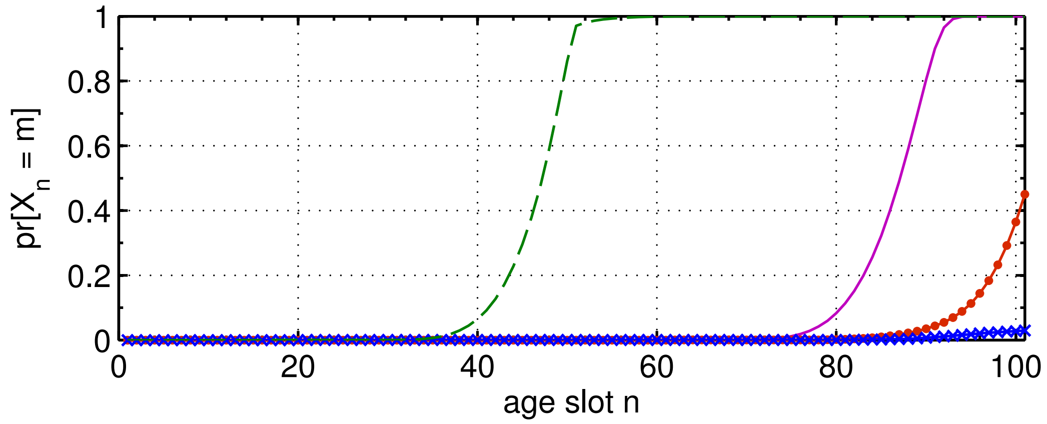


(b) Most Urgent

Figure 3.10: Effect of the number of blocks m per segment on the performance of both strategies when $\frac{E[U_p]}{R_s} = 1.0203$



(a) Uniform Strategy



(b) Most Urgent Strategy

Figure 3.11: Effect of the aggressiveness a on the performance of both strategies when $\frac{E[U_p]}{R_s} = 1.0203$

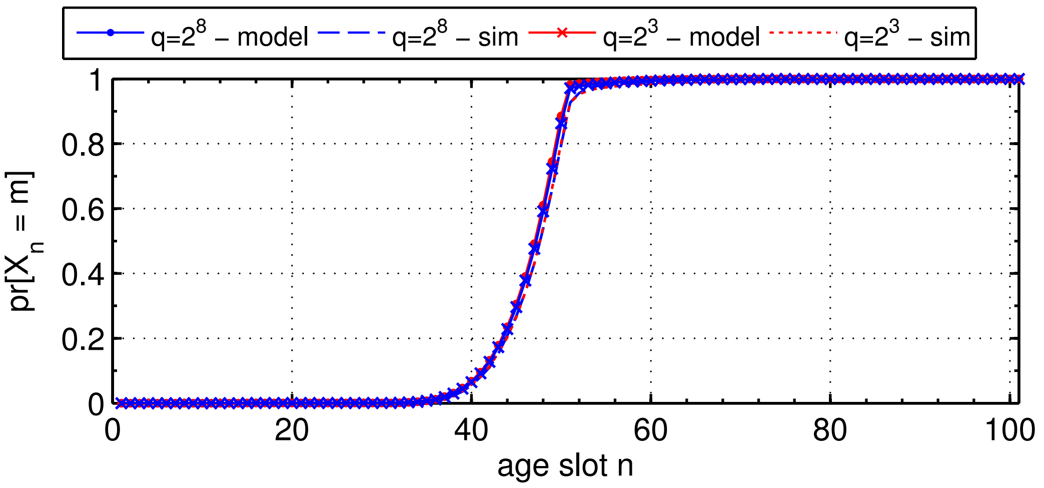
3.2.2 Aggressiveness a

We have found that setting a to as low as 1 coded block would give the best performance for both strategies as shown in Figure 3.11. This is because it makes the segment Ready-to-be-Served as early as possible while the linear dependency of the sent coded blocks does not constitute a major hurdle. Increasing a would lead to not only having fewer peers able to serve the segments early at age 0 but also making other peers take more time to start serving the segment that would especially have much worse effect on the most urgent strategy as shown for $a = 5$ in Figure 3.11.

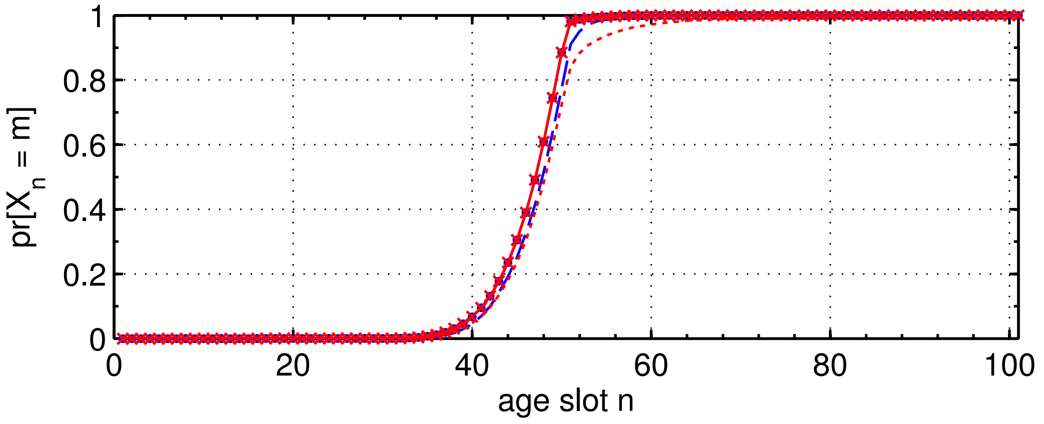
With low values of a, much shorter buffer lengths could be used that would make the overhead of exchanging buffer maps even smaller.

3.2.3 Galois Field Size q

The coefficients used to form a linear combination of the existing blocks of a segment and generate a coded block are chosen uniformly at random and independently from the Galois field of size q. The larger the field size is the more randomization occurs when generating a coded block which increases its chances to be linearly independent. However, the coefficients overhead would increase with larger field sizes as shown in



(a) Uniform Strategy



(b) Most Urgent Strategy

Figure 3.12: Effect of the field size q on the performance of both strategies when $\frac{E[U_p]}{R_s} = 1.0203$

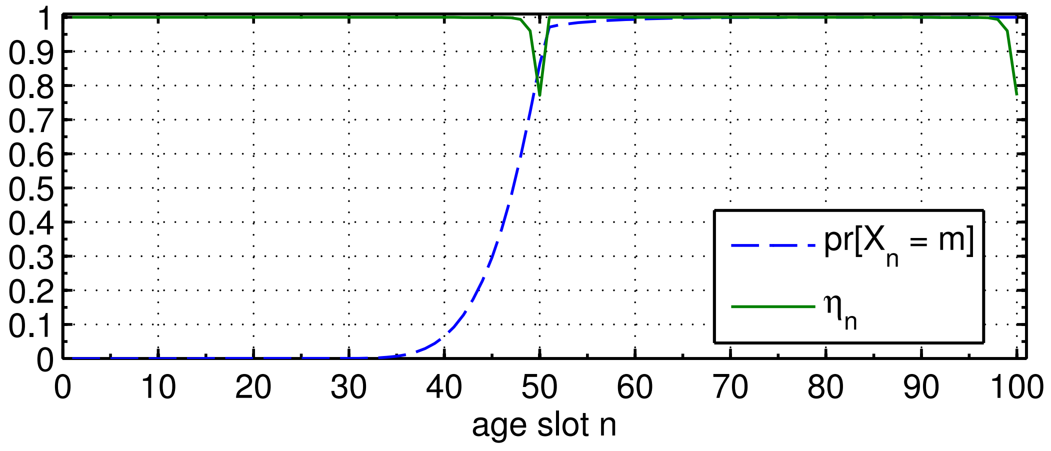
eq. (2.6). Having said that, we have found out that as long as q is not very small, the opposite effects of linear independence and coefficients overhead will balance each other out such that the total amount of wasted bandwidth would almost be the same when varying q between 2^3 to 2^8. This would lead to having almost the same performance for different values of q as shown in Figure 3.12 making the choice of the field size be not that critical. For our standard configuration when $\frac{E[U_p]}{R_s} = 1.0203$, we have found in the uniform strategy that the coefficients overhead increases from 1.9139 kb/s for $q = 8$ to 5.1108 for $q = 256$ while the bandwidth wasted on linearly dependent blocks decreases from 7.7086 kb/s to 3.6422 kb/s for $q = 8$ and $q = 256$ respectively. In addition the total extra bandwidth needed that includes redundant blocks is 22.89 kb/s and 22.93 kb/s for $q = 8$ and $q = 256$ respectively. The numbers for the most urgent strategy are very close to the uniform's.

3.3 Efficiency η

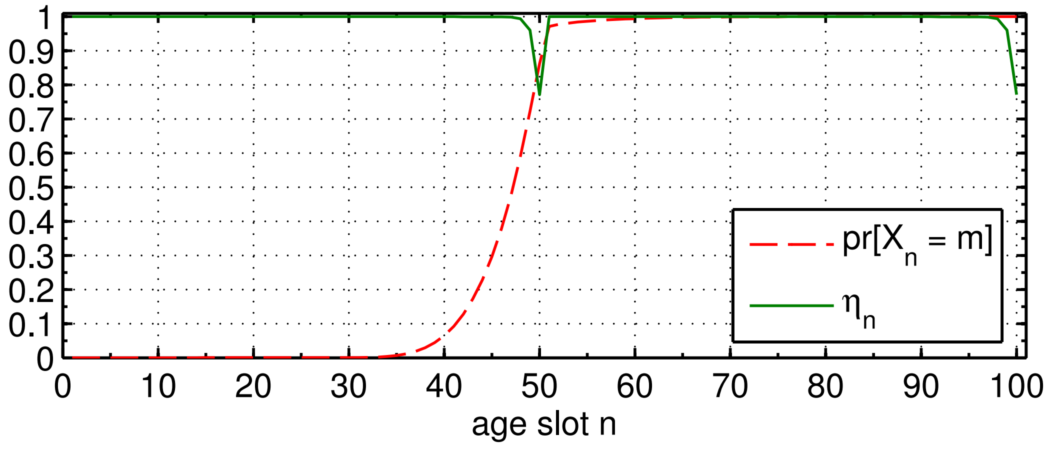
Network coding has been found to increase the efficiency of multicast sessions on certain network topologies as we mentioned in the first chapter. This also holds true for P2P live video streaming systems. We have found that when system parameters are tuned up properly, network coding achieves an average efficiency of more than 95% in most cases.

Figure 3.13 shows the efficiency η_n as a function of the age slot for both strategies when the mean peer upload bandwidth is greater than the streaming rate. In this case, the total average bandwidth supply available per peer is 662.933 kb/s which is the sum of the average peer capacity $E[U_p]$ of 653 kb/s and the average server contribution per peer of about 9.933 kb/s calculated from equation (2.20).

As we discussed previously, we can observe that the efficiency is periodic over any period that is equal to segment duration. We can also observe that in both strategies,



(a) Uniform Strategy



(b) Most Urgent Strategy

Figure 3.13: Efficiency of both strategies when $\frac{E[U_p]}{R_s} = 1.0203$

the efficiency starts to decline towards the last few age slots in the period until it drops just under 80% at the end of the period. This decline could be explained easily. As the segment in the third position is complete with high probability throughout almost all of its stay in this position, both strategies will select the only other remaining servable segment in the second buffer position. However, as this segment gets closer to its stay in the second position, its probability to be complete increases to above 50% making it less servable and causing the efficiency to drop at that time.

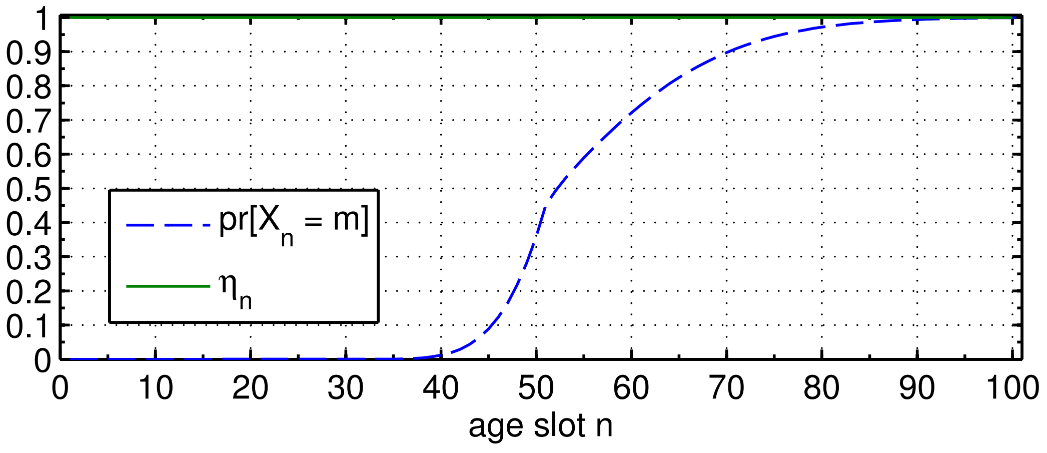
The average efficiency for both strategies in Figure 3.13 as calculated from equation (2.67) is almost 99.44%. However, this efficiency only accounts for linearly dependent blocks and does not include the effect of the overhead and redundant blocks. We

have calculated the extra bandwidth taken up by the overhead and redundant blocks using equations (2.75) and (2.76) and found it to be approximately 5.11 kb/s and 14.2 kb/s for the overhead and redundant blocks respectively for both strategies in Figure 3.13. If we include this extra wasted bandwidth, we get an average efficiency of approximately 96.54% for both strategies.
For the system to be stable, 96.54% of the total bandwidth supply per peer (662.933 kb/s) which amounts to 639.99 kb/s should be equal to the actual consumption rate per peer. Since the continuity is almost 1 in Figure 3.13, the actual amount of useful data consumed per second on a peer is equal to the full streaming rate of 640 kb/s which verifies the system is stable.

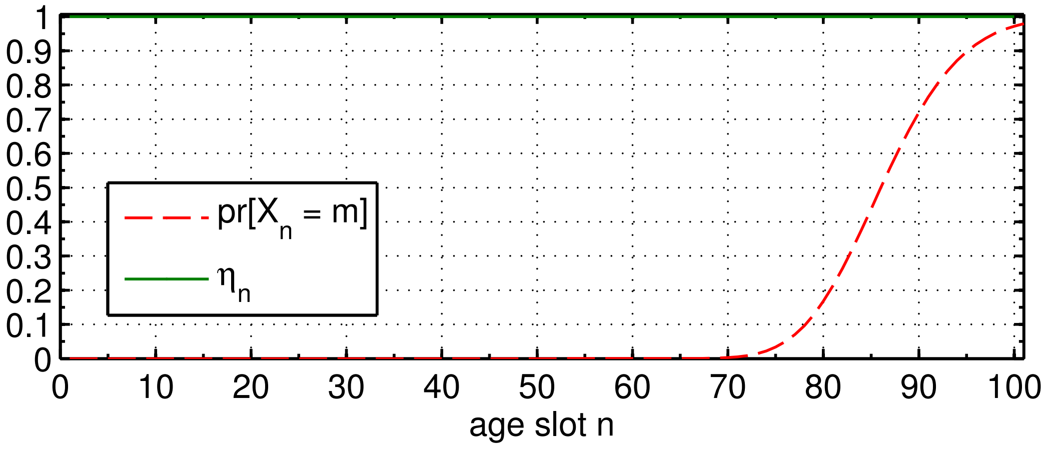
For the case of $E[U_p] = R_s$, we present the efficiency in Figure 3.14 for both strategies. The total average bandwidth supply per peer in this case is $E[U_p]$ + server contribution = 640 + 9.933 = 649.933 kb/s. To achieve continuity of 1, the demand per peer should be equal to the streaming rate (640 kb/s) plus the extra bandwidth wasted on linearly dependent and redundant blocks and coefficients overhead.

As we see from Figure 3.14, both strategies achieve an efficiency close to 1 throughout the period of a segment duration with an average efficiency of about 99.99 for both. This proves the linear independence is a minor downside in this case. There is no drop in the efficiency towards the end of the period here compared to Figure 3.13. This is because in the most urgent strategy, the most urgent segment towards the end of the period is most likely the one in the second position which is certainly not complete yet. As for the uniform strategy, there is an extremely slight drop towards the end of the period since the segment in the third position is certainly complete but it is not noticeable since the segment in the end of the second position is less than 50% complete.

The bandwidth is wasted mainly on the coefficients overhead and redundant blocks. For the uniform strategy, we have calculated them to be about 5.04 and



(a) Uniform Strategy



(b) Most Urgent Strategy

Figure 3.14: Efficiency of both strategies when $E[U_p] = R_s$

4.833 kb/s respectively for a total of 9.87 kb/s. If we include them into the calculation of the efficiency, we get an efficiency of 98.47%. This efficiency would make the amount of useful average bandwidth supply per peer be 639.989 kb/s which is sufficient to provide the peer with almost the full streaming rate of 640 kb/s. This explains the close-to-one continuity the uniform strategy is achieving in Figure 3.13. On the other hand, in the most urgent strategy, we calculated the coefficients overhead and redundant blocks to be 5.04 and 5.636 kb/s respectively for a total of 10.673 kb/s. If we include them into the calculation of the efficiency, we get an efficiency of 98.35% which is slightly less than the uniform strategy. This efficiency would make the amount of useful average bandwidth supply per peer be 639.209 kb/s which is *not* sufficient to provide the peer with the full streaming rate of 640 kb/s. This reveals why the performance of the most urgent strategy is worse than the uniform strategy.

Chapter 4

Conclusion and Future Work

Numbers are confirming what we are witnessing in our everyday lives. Video traffic is dominating the internet now. And with the advances in high quality content creation and more video-based services are becoming standard on the internet such as video conferencing and IPTV, video traffic growth is forecast to continue to explode over the next few years. This will present a challenge for the current delivery architectures to continue to meet the expected explosion in bandwidth demand. P2P video streaming systems have the potential to play a significant role in meeting the anticipated bandwidth demand. They have the ability to leverage the bandwidth resources of end users which would have even more importance since the capacity of broadband links available to home and enterprise users is expected to increase and also reach more customers. Moreover, they provide large-scale capability at low cost.

We believe the introduction of network coding to P2P video streaming was a game changer. It helped resolve some of the issues that plagued previous designs. With proper design, network coding helps utilize the available bandwidth supply more efficiently especially when the supply barely exceeds the demand. And more importantly, it brings new levels of robustness to the P2P system that is known for its dynamic nature and unpredictable user behavior. Above all that, it simplifies the design while keeping the bandwidth redundancy in check.

However, not a lot of research has been done for the integration of network coding with P2P video streaming and that is true especially when it comes to mathematically studying these systems. Our research attempted to fill this gap by providing a stochastic model to analyze the performance of network coding and investigate the influence of the system parameters and design options.

4.1 Conclusion

Our model is unique in that it reveals how the number of blocks in a segment evolve as the segment grows older in the buffer. To do that, we divided the segment lifetime in the buffer into age slots, then defined X_n to be the number of blocks a segment has at age slot n. Once the system is in steady state the distribution of X_n would not change. We have observed that the distribution of any X_n depends only the number of blocks at the previous age slot X_{n-1} and the number of useful blocks received at the current age slot R_n. To obtain R_n, we had to find the probability β_{n-1} that a neighbor at age slot $n-1$ pushes a useful coded block to one of its downstream peers at $n-1$. β depends on the buffer states and neighbor and segment selection strategies. We derived formulas for β for both the uniform and the most urgent segment selection strategies.

From β, the distribution of R_n is found as a binomial distribution with success probability β_{n-1} and total number of trials equal to either $i)$ H in the homogeneous upload bandwidth case and , or equal to $ii)$ $\widehat{H}$, which is the sum of the total number of blocks all neighbors can push at the current time slot, in the heterogeneous bandwidth case. Next, the unconditional distributions of X_n's are obtained by solving a series of equations starting from X_0 at the beginning of the servable-by-peers segment lifetime to X_N at the beginning of the playback position where the video playback

would be continuous if the segment is complete. Thus the probability of continuity is equivalent to $pr[X_n = m]$.
The distribution of X_0 depends solely on the server design. We have defined a simple server design that enabled us to characterize the server contribution in the video streaming session and derive the distribution of X_0. In this design, for each new segment, the server selects a number of peers uniformly at random from all the peers in the overlay and sends only a linearly independent blocks to each. Since the aggressiveness could be set to as low as 1 while keeping usefulness of coded blocks high, our design maximizes the use of the capacity of the server by spreading a new segment to as many peers as possible spread all across the overlay and are able to serve it as quickly as possible.

We have also derived a good approximation of the probability a pushed coded block is linearly independent through studying the relation between the subspaces spanned by the coded blocks on the upstream and downstream peers. This allowed us to study the effect of the linear dependency on the performance. We found out that received coded blocks are useful with high probability even when the aggressiveness is set to a low value.

We concluded the model by studying the efficiency of network coding. We derived formulas for the efficiency of both the uniform and segment selection strategies. We have also derived many performance measures that enabled us to verify the stability of the system as well as quantify the extra bandwidth wasted on the coefficients overhead and redundant blocks.

We extracted numerical results from the model and performed simulations to verify the model. Our simulation results confirmed the behavior we got in the model for all the system parameters and design options.

4.2 Thesis Contribution

Modeling P2P live streaming with network coding with the goal of capturing the influence of the system parameters is a challenging task. The division of segments into blocks and the linear dependency that may exist among these blocks introduce more complexity into the analysis. However, we have been able to develop a unique stochastic model that can reveal the effect of the system parameters and design options on the performance. Moreover, the model could be extended to analyze more design options. We have also proposed a new approximation to study the effect of linearly dependent blocks. Through this model, we have provided in-depth insights into the influence of most of the system parameters and design options.

One of our most important findings is that some design options like the uniform segment selection strategy empowers the protocol to achieve a better performance than the most urgent strategy especially when the bandwidth supply marginally outstrips the demand. Moreover, the uniform strategy takes a better advantage of the available buffering time to enable the system to scale to hundreds of thousands of users with relatively small drops in the performance while maintaining a continuity close to 1.

We have also showed that there is no need to carefully tune up some system parameters such as the number of neighbors H and Galois field size q. While the same could be said about the aggressiveness a in terms of linear dependency, we have revealed that setting it to as low as possible would unleash new levels of performance and expedite the process of disseminating a new segment in the overlay.

We have also discussed the influence of the number of blocks per segment m. Although it was shown in the literature that larger m with smaller block sizes are favored, we have identified more clearly the trade-off between m and the coefficients overhead.

We have also reported that network coding enjoys a good efficiency of more than 95% in most cases. In addition, we found out the uniform strategy is slightly more efficient than the most urgent strategy which would make a huge difference in the performance when the bandwidth supply slightly outrun the demand.

4.3 Limitations and Future work

Our model captures the effect of most of the system parameters and manages to reveal the unique nature of network coding in which segments may receive blocks throughout their stay in the buffer. Having said that, there is still room for improvement. There are many aspects that could be improved and many more design options that could be modeled. We list next some of the most important limitations that we wish to address in the future:

- We assumed the peers are synchronized in the sense they are playing the same video instant at approximately the same time. Although such tight synchronization has its benefits in allowing the buffers of the peers to overlap as much as possible and some protocol designs like R^2 call for such synchronization, achieving this level of synchronization is difficult in a real implementation. Therefore, it would be of great interest to investigate the effect of relaxing the synchronization. It would be sufficient to relax the synchronization assumption just a little bit such that peers playback pointers are within the same segment. This could be easier to implement since network coding uses larger segment durations. However judging from the results we obtained for the influence of the number of neighbors, having large enough number of neighbors would compensate for the effect of the lack of synchronization as a segment is most likely to still have a few neighbors able to serve it.

- More peer and segment selection strategies could be modeled. For instance, for peer selection, a peer could favor neighbors that have more non-complete segments which may be helpful for peers that have just joined the session. As for the segment selection, a hybrid strategy of both the most urgent and uniform could be modeled. For instance the strategy used in R^2 divides the buffer into two regions and uses uniform selection for one region and some form of most urgent selection for the other region. It would be interesting to find out how this strategy used in R^2 stacks up against the pure uniform or most urgent strategies. Our model could be easily extended to study such a strategy.

- Network coding designs are known for their robustness to peer dynamics and random packet loss and delay. Nonetheless, it would be interesting to quantify this in our model. We assumed a constant number of neighbors H that does not change throughout the streaming session. We could instead model the neighbor lifetime by some sort of distribution such the exponential distribution and then study the effect of a random number of neighbors. However, judging from our results, the protocol is still able to achieve a good performance even for a very small number of neighbors which may indicate that maintaining a relatively large number of neighbors would prepare the protocol to deal with extreme peer churn.

- Redundant blocks that are received after the segment is complete is a downside of network coding. Although their effect could be mitigated as we showed, it would still be of great interest to design schemes to try to avoid receiving redundant blocks. For instance, having the peer when a segment is about to get complete send stop signal to gradually stop its neighbors and start with neighbors that have lower upload bandwidth.

We hope to address these limitations in future work.

Bibliography

[1] "Cisco visual networking index: Forecast and methodology, 2012 – 2017," White Paper, Cisco, 2012.

[2] Wikipedia. Napster. [Online]. Available: https://en.wikipedia.org/wiki/Napster

[3] R. Stern, "Napster: a walking copyright infringement?" *Micro, IEEE*, vol. 20, no. 6, pp. 4–5, 95, 2000.

[4] Wikipedia. Gnutella. [Online]. Available: https://en.wikipedia.org/wiki/Gnutella

[5] C. Wang and B. Li, "Peer-to-peer overlay networks: A survey," Tech. Rep., 2003.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 161–172, Aug. 2001.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '01. New York, NY, USA: ACM, 2001, pp. 149–160.

[8] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Revised Papers from the First International*

Workshop on Peer-to-Peer Systems, ser. IPTPS '01. London, UK, UK: Springer-Verlag, 2002, pp. 53–65.

[9] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended lans," *ACM Trans. Comput. Syst.*, vol. 8, no. 2, pp. 85–110, May 1990.

[10] Y.-h. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast (keynote address)," *SIGMETRICS Perform. Eval. Rev.*, vol. 28, no. 1, pp. 1–12, Jun. 2000.

[11] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr., "Overcast: reliable multicasting with on overlay network," in *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation - Volume 4*, ser. OSDI'00. Berkeley, CA, USA: USENIX Association, 2000, pp. 14–14.

[12] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 298–313, Oct. 2003.

[13] B. Li, Y. Feng, and B. Li, "Rise and fall of the peer-to-peer empire," *Tsinghua Science and Technology*, vol. 17, no. 1, pp. 1–16, 2012.

[14] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: eliminating trees from overlay multicast," in *Proceedings of the 4th international conference on Peer-to-Peer Systems*, ser. IPTPS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 127–140.

[15] X. Zhang, J. Liu, B. Li, and T. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *INFOCOM 2005. 24th*

Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 3, 2005, pp. 2102–2111 vol. 3.

[16] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, “Network information flow,” *Information Theory, IEEE Transactions on*, vol. 46, no. 4, pp. 1204–1216, 2000.

[17] R. Koetter and M. Medard, “An algebraic approach to network coding,” *Networking, IEEE/ACM Transactions on*, vol. 11, no. 5, pp. 782–795, 2003.

[18] S.-Y. Li, R. Yeung, and N. Cai, “Linear network coding,” *Information Theory, IEEE Transactions on*, vol. 49, no. 2, pp. 371–381, 2003.

[19] T. Ho, R. Koetter, M. Mdard, D. Karger, and M. Effros, “The benefits of coding over routing in a randomized setting,” 2003, p. 442, cited By (since 1996)160.

[20] C. Gkantsidis and P. Rodriguez, “Network coding for large scale content distribution,” in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, 2005, pp. 2235–2245 vol. 4.

[21] C. Gkantsidis, J. Miller, and P. Rodriguez, “Anatomy of a P2P Content Distribution system with Network Coding,” Feb. 2006.

[22] M. Wang and B. Li, “Lava: A reality check of network coding in peer-to-peer live streaming,” in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 1082–1090.

[23] M. Wang and B. Li, “R2: Random push with random network coding in live peer-to-peer streaming,” *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 9, pp. 1655–1666, 2007.

[24] Z. Liu, C. Wu, B. Li, and S. Zhao, "Uusee: Large-scale operational on-demand streaming with random network coding," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1–9.

[25] P. Maymounkov, N. J. A. Harvey, and D. S. Lun, "Methods for Efficient Network Coding," 2006.

[26] C. Feng and B. Li, "On large-scale peer-to-peer streaming systems with network coding," in *Proceedings of the 16th ACM international conference on Multimedia*, ser. MM '08. New York, NY, USA: ACM, 2008, pp. 269–278.

[27] L. Chang and J. Pan, "On the system parameters of peer-to-peer video streaming with network coding," in *Communications (ICC), 2010 IEEE International Conference on*, 2010, pp. 1–5.

[28] S. Deb, M. Medard, and C. Choute, "Algebraic gossip: a network coding approach to optimal multiple rumor mongering," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2486 – 2507, june 2006.

[29] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network coding: an instant primer," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, Jan. 2006.

[30] K. M. Greenan and E. L. Miller, "Analysis and construction of galois fields for efficient storage reliability," 2007.

[31] J. Luo, K. D. Bowers, A. Oprea, and L. Xu, "Efficient software implementations of large finite fields gf(2n) for secure storage applications," *Trans. Storage*, vol. 8, no. 1, pp. 2:1–2:27, Feb. 2012.

[32] A. Nagarajan, M. Schulte, and P. Ramanathan, "Galois field hardware architectures for network coding," in *Architectures for Networking and Communications Systems (ANCS), 2010 ACM/IEEE Symposium on*, 2010, pp. 1–9.

[33] H. Shojania and B. Li, "Parallelized progressive network coding with hardware acceleration," in *Quality of Service, 2007 Fifteenth IEEE International Workshop on*, 2007, pp. 47–55.

[34] X. Chu, K. Zhao, and M. Wang, "Massively parallel network coding on gpus," in *Performance, Computing and Communications Conference, 2008. IPCCC 2008. IEEE International*, 2008, pp. 144–151.

[35] H. Shojania, B. Li, and X. Wang, "Nuclei: Gpu-accelerated many-core network coding," in *INFOCOM 2009, IEEE*, 2009, pp. 459–467.

[36] Z. Zhang, R. Hou, H. Chen, J. Zhou, and J. Li, "Network coding based live peer-to-peer streaming towards minimizing buffering delay," in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, vol. 4, 2010, pp. V4–136–V4–140.

[37] Y. Hong, "On computing the distribution function for the poisson binomial distribution." *Computational Statistics & Data Analysis*, vol. 59, pp. 41 – 51, 2013.

[38] R. Barlow and K. Heidtmann, "Computing k-out-of-n system reliability," *Reliability, IEEE Transactions on*, vol. R-33, no. 4, pp. 322–323, 1984.

[39] O. Trullols-Cruces, J. Barcelo-Ordinas, and M. Fiore, "Exact decoding probability under random linear network coding," *Communications Letters, IEEE*, vol. 15, no. 1, pp. 67 –69, january 2011.

[40] D. Qiu and R. Srikant, “Modeling and performance analysis of bittorrent-like peer-to-peer networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 367–378, Aug. 2004.