ON THE RESILIENCE OF NETWORK CODING IN PEER-TO-PEER NETWORKS AND ITS APPLICATIONS

BY

Di Niu

A THESIS SUBMITTED IN CONFORMITY WITH THE REQUIREMENTS FOR THE DEGREE OF MASTER OF APPLIED SCIENCE, DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, AT THE UNIVERSITY OF TORONTO.

Copyright © 2009 by Di Niu. All Rights Reserved.
Abstract

Most current-generation P2P content distribution protocols use fine-granularity blocks to distribute content in a decentralized fashion. Such systems often suffer from a significant variation in block distributions, such that certain blocks become rare or even unavailable, adversely affecting content availability and download efficiency. This phenomenon is further aggravated by peer dynamics which is inherent in P2P networks. In this thesis, we quantitatively analyze how network coding may improve block availability and introduce resilience to peer dynamics. Since in reality, network coding can only be performed within segments, each containing a subset of blocks, we explore the fundamental tradeoff between the resilience gain of network coding and its inherent coding complexity, as the number of blocks in a segment varies. As another application of the resilience of network coding, we also devise an indirect data collection scheme based on network coding for the purpose of large-scale network measurements.
To my parents
Acknowledgments

I would like to first express my sincere gratitude to my thesis supervisor, Professor Baochun Li, for the invaluable help he has provided to me throughout my time as a Master’s student of his. His patient guidance and encouragement not only greatly aided me in the completion of this thesis, but also significantly helped me build up my understanding and interests into the research of computer networks, a field that I would like to explore further in my PhD studies.

I would also like to thank all the members in iQua research group at University of Toronto, with whom I was involved in frequent discussions which eventually benefited my thesis work. I am particularly thankful to Chen Feng for his helpful suggestions on developing the proofs of several theorems in Chapter 4. Completing this work would have been all the more difficult were it not for the support and friendship provided by other members of iQua research group.

I must express my gratitude to my mother and father for their continued support and encouragement. They accompanied me throughout my Master’s studies at University of Toronto as a student from overseas, and experienced all the ups and downs of my research. Without their support, I would have never been able to focus on my research so much. A big thank you to them!

Finally, I would like to thank Professor Ben Liang and Professor Peter Marbach for their important suggestions on revising the thesis to its final completion. Their questions and suggestions have largely helped me to improve the rigor of the technical materials in Section 4.1, 4.2, and 4.3, especially the presentation of Section 4.2 that concerns with the derivation of differential equations that approximate the underlying Markov processes under certain assumptions.
Contents

Abstract ii

Acknowledgments iii

List of Tables vi

List of Figures x

1 Introduction 1

2 Related Work 5

3 P2P Content Distribution with Network Coding 9
   3.1 Mesh-based P2P Content Distribution 9
   3.2 Segment-based Network Coding 11
   3.3 The Application of Network Coding in P2P Content Distribution 13

4 On the Resilience-Complexity Tradeoff of Network Coding 16
   4.1 Model Description 16
   4.2 Characterizing Asymptotic Systems via Differential Equations 20
      4.2.1 System State Representation 20
4.2.2 Deriving Differential Equations: Replacement Model .......................... 22
4.2.3 Deriving Differential Equations: Poisson Arrival Model ...................... 29
4.3 Steady-State Block Availability ......................................................... 31
   4.3.1 Block Distribution ............................................................... 32
   4.3.2 Block Variation and Discussions .............................................. 36
4.4 Transient Behaviours ......................................................................... 38
4.5 Experimental Results ......................................................................... 42
   4.5.1 Steady-State Block Distribution and Download Time ....................... 44
   4.5.2 Content Persistence in the Absence of Seeds ................................ 48
4.6 Summary ......................................................................................... 49
5 Indirect Data Collection with Network Coding ................................. 50
   5.1 Data Spreading with Gossiping ..................................................... 53
   5.2 Resilience of Random Network Coding ......................................... 58
   5.3 Buffer and Server Algorithms for Stream Data Collection .................. 64
   5.4 A Differential Equation Characterization ....................................... 68
   5.5 Performance Evaluation ............................................................. 73
   5.6 Summary .................................................................................... 82
6 Conclusions and Future Work .......................................................... 83

Bibliography ....................................................................................... 86
List of Tables
List of Figures

4.1 The replacement model for peer dynamics. $L$ denotes the peer lifetime. $A$ denotes the peer age at a given observation time $t_o$. Every new peer joins the session by replacing a departed peer. .................................. 18

4.2 An illustration for the notations $p_i(t)$, $s_i(t)$, $n_i(t)$ and $r_i(t)$. .......................... 21

4.3 Steady-state block distribution (CDF of $I$) for different values of segment size $m$. The number of downloaders $N = 6000$, number of seeds $N_s = 100$, file size $F = 768$ MB, block size $k = 256$ KB, average peer age $\overline{A} = 5$. There are an equal number of peers with upload bandwidth 2 MB/round, 512 KB/round, and 256 KB/round so that the average upload rate is $\tilde{\mu} = \mu/k = 3.7$) ......................................................... 35

4.4 The fraction of content lost immediately upon seed departures. The number of downloaders $N = 1000$, the number of seeds $N_s = 10$, the total number of blocks in the content $M = 1000$, upload rate $\mu/k = 4$. .......................... 40

4.5 The fraction of decodable content at time $t$ after seed departures. $N = 1000$, $N_s = 50$, $M = 1000$, upload rate $\tilde{\mu} = \mu/k = 4$, average peer age $\overline{A} = 5$. ................................................................. 41
4.6 Content lifetimes after seed departures under different segment sizes $m$ and different average peer age $\overline{A}$. Parameters are the same as in the previous figure. ................................................................. 41

4.7 The convergence of the block variation. $N = 6000$, $N_s = 100$, $F = 768$ MB, $k = 256$ KB. $\overline{L} = 5$ rounds. ................................................................. 44

4.8 Steady-state CDF of the number of blocks of each segment found in all the downloaders. Average lifetime of downloaders $\overline{L} = 5$ rounds. ................. 44

4.9 Steady-state block variation as a function of segment size $m$ for the replacement model. $N = 6000$, $N_s = 100$, $F = 768$ MB, $k = 256$ KB. .......... 44

4.10 Steady-state block variation as a function of segment size $m$ for the Poisson arrival model. $N = 3000$, $N_s = 100$, $F = 768$ MB, $k = 256$ KB. .......... 44

4.11 The average time for a long-lived peer to download the entire content in steady state under different segment sizes $m$ for the replacement model. $N = 2000$, $N_s = 30$, $F = 256$ MB, $k = 256$ KB. 1% of all the downloaders are long-lived. ................................................................. 45

4.12 The average time for a long-lived peer to download the entire content in steady state under different segment sizes $m$ for the Poisson arrival model. $N_s = 30$, $F = 256$ MB, $k = 256$ KB, $\lambda = 100$. 1% of all the downloaders are long-lived. ................................................................. 45

4.13 Steady-state block variation under different peer joining rates $\lambda$. $N_s = 100$, $F = 768$ MB, $\overline{L} = 10$ rounds. ................................................................. 46

4.14 The average time for a long-lived peer to download the entire content in the steady state. ................................................................. 46
4.15 The number of blocks that each segment has at different times $t$ (rounds).
   a) non-coding scheme (segment size $m = 1$). b) network coding with
   segment size $m = 4$. Servers leave in the steady state (round 400), and we
   set $t = 0$ again upon seeds departure. $N = 6000$, $N_s = 100$, $F = 768$ MB,
   $k = 256$ KB, $L = 20$ (rounds). ........................................... 47

4.16 The fraction of decodable content after server (seed) departures as time
   passes. Seeds leave in the steady state (round 400), and we set $t = 0$ again
   upon seed departures. $N = 6000$, $N_s = 100$, $F = 768$ MB, $k = 256$ KB,
   $L = 20$ (rounds). ........................................... 48

5.1  (a) Using the logging servers to directly “pull” from all the peers may
   not be scalable, as server bandwidth needs to accommodate the peak load
   in a flash crowd scenario (such as during evening prime times in P2P
   streaming). (b) If we are able to tolerate some delays, peers can be asked
   to exchange data blocks in a distributed fashion to achieve better data
   persistence. ............................................................... 54

5.2  The expected number of original blocks lost permanently upon each peer
   departure at time $t$ under the gossip protocol. .......................... 57

5.3  A bipartite graph showing the relationship among segments, blocks and
   peers with $m = 6$, $s = 3$. Every edge between a segment and a peer
   represents a (coded) block of this segment in this peer. Solid lines denote
   original blocks, while dashed ones denote transferred blocks. ............ 60

5.4  The expected number of original blocks lost permanently upon each peer
   departure for different segment sizes $s$. .............................. 63

5.5  A bipartite graph representation. ................................. 66
5.6 Session throughput as a function of segment size $s$. $\lambda = 20$, $\mu = 10$, $\gamma = 1$. 76
5.7 Session throughput as a function of $\mu$ under different scenarios. $\lambda = 8$, $\gamma = 1$. 77
5.8 Session throughput as a function of $\mu$ and $\gamma$. $\lambda = 20$, $c = 20$. 78
5.9 Average block delivery delay $T$ for different values of $s$. $\lambda = 20$, $\mu = 10$, $\gamma = 1$. 80
5.10 Average block delay $T$ as a function of $\mu$ and $\gamma$. $\lambda = 20$, $c = 20$. 80
5.11 Data saved in each peer, or the number of original blocks buffered in each peer that have not been collected. $\lambda = 20$, $\mu = 10$, $\gamma = 1$. 81
Chapter 1

Introduction

Peer-to-peer (P2P) content distribution has become the de facto standard in current-generation content distribution protocols. The basic idea in current-generation P2P content distribution protocols such as BitTorrent is to segment large volumes of data (usually hundreds of megabytes or even gigabytes) into fine-granularity blocks, and then distribute these blocks in an efficient manner by letting peers exchange them with one another.

In reality, however, P2P protocols often suffer from severe peer dynamics (i.e., arrivals and departures), as peers are inherently unreliable. As the fundamental philosophy of peer-to-peer protocols is to use resources on the peers to store blocks of content, content distribution sessions may continue to proceed even when the original peers (sometimes referred to as seeds) are no longer available. However, in these cases, blocks may become rare or even unavailable when peers arrive and depart frequently with short lifetimes. Such significant degrees of imbalance with respect to block availability — henceforth referred to as block variation — will adversely affect the availability of content, leading to longer downloading times at each peer.
As end hosts at the edge of the Internet possess abundant computational resources with modern processors, it is natural to take advantage of the power of network coding in P2P applications, by allowing end hosts to not only forward and replicate, but to code as well. Network coding has been originally proposed in information theory [2, 21, 18], and has been more recently used to improve the resilience to peer dynamics in P2P content distribution protocols [13], leading to shorter downloading times. Intuitively, one may observe that, since network coding distributes coded blocks rather than original blocks, and all coded blocks are equally innovative and useful to any peer, the challenge of locating rare original blocks may indeed be addressed.

However, network coding may not realize its benefits without introducing significant computational complexities at peers. It has been shown in recent work [33] that, network coding may not be computationally feasible if one is to code more than a few hundred blocks, even with modern processors! In order to reduce coding complexity, Chou et al. [5] have proposed the concept of group network coding, which performs coding on the blocks within the same group or segment, while each segment contains a prescribed (and arguably small) number of blocks. Though group network coding helps to reduce coding complexity, its negative effects on the resilience to peer dynamics — the main advantage of using network coding in the first place in P2P networks — are not fully understood.

Let us consider two extremes of P2P protocol design. The first one does not use network coding at all, and the second uses network coding across all existing blocks. If we consider the number of blocks in each segment (referred to as the segment size) in group network coding, we may observe that the first extreme corresponds to a segment size of one, with as many segments as blocks, while the other extreme corresponds to the case of grouping all blocks into the same segment. Intuitively, the degree of resilience to
peer dynamics that network coding has to offer improves as we increase the segment size; but the coding complexity increases as well. If we vary the segment size when network coding is used, we are fundamentally moving from one extreme to another, and making our choice in the challenge of resilience-complexity tradeoff of network coding in P2P networks. It would be best if we may operate with an appropriate segment size to enjoy most of the resilience advantage of using network coding, but with acceptable coding complexity.

Motivated by our curiosity on choosing the “sweet spot” in the resilience-complexity tradeoff, we quantitatively evaluate the content availability with different segment sizes, using both theoretical analysis and simulations. In our theoretical analysis, we consider large-scale dynamic P2P systems, where each peer is to have a random lifetime following an arbitrary yet general distribution. We use a linear system of differential equations to asymptotically approximate the underlying Markov population process with regard to block distribution, and derived closed-form results with respect to the variations of block availability. We use simulations to not only substantiate our theoretical conclusions, but also shed further insights into the problem in a wide range of scenarios.

Leveraging the ability of network coding to preserve content availability in face of peer churn, we further apply network coding in a reverse process of content distribution, namely, the statistics data collection process. In most large-scale peer-to-peer (P2P) applications, it is necessary to collect vital statistics data — sometimes referred to as logs — from up to millions of peers. Traditional solutions involve sending large volumes of such data to centralized logging servers, which are not scalable. In addition, they may not be able to retrieve statistics data from departed peers in dynamic P2P systems. We solve this dilemma through an indirect collection mechanism that distributes data
using randomized network coding across the network, from which servers proactively pull such statistics in a delayed fashion. We map the statistics collection process onto a random graph process, which is asymptotically characterized by a set of differential equations. With appropriate assumptions, we derive theoretical bounds showing that network coding can almost exponentially improve data persistence in the presence of peer dynamics, as we increase the number of blocks in a segment for coding. By buffering data in a decentralized fashion with only a small portion of peer resources, we show that our new mechanism provides a “buffering” zone and a “smoothing” factor to collect large volumes of statistics, with appropriate resilience to peer dynamics and scalability to a large peer population.

The remainder of the thesis is organized as the following. We present related work in Chapter 2. Chapter 3 presents a brief background on current-generation P2P content distribution protocols and randomized network coding. In Chapter 4, we evaluate the resilience of network coding in dynamic P2P networks with respect to enhancing block availability and reducing download time as the number of blocks in a segment for coding varies. In Chapter 5, we propose our indirect scheme for large-scale statistics data collection based on randomized network coding and evaluate its performance through both analysis and simulation.
Chapter 2

Related Work

The pioneering work by Ahlswede et al. [2] and Koetter et al. [18] proves that, in a directed network with network coding support, a multicast rate is feasible if and only if it is feasible for a unicast from the sender to each receiver. Li et al. [21] has further proved that linear coding usually suffices in achieving the maximum rate. These results are significant in the sense that, with network coding, the cut-set capacity bounds of unicast flows from the source to each of the receivers can be achieved in a multicast session. In other words, network coding helps to alleviate competition among flows at the bottleneck, thus improving session throughput in general.

Since the landmark paper on randomized network coding by Ho et al. [15], there has been a gradual shift in research focus in the area of network coding, from theoretical studies on achievable flow rates and code assignment algorithms, to more practical studies on applying network coding in a practical setting. Such a shift of focus has been marked by the work of Chou et al. [5], which concludes that randomized network coding can be designed to be robust to random packet loss, delay, as well as any changes in network topology and capacity. Avalanche [13, 11] has further proposed that randomized network
coding can be used for elastic content distribution. However, in recent studies, Chiu et al. [4] demonstrated that network coding can not theoretically further increase throughput in P2P networks. It has also been shown [33] that the coding complexity escalates when an increasing number of blocks are used in network coding, and that even with modern processors, coding more than a few hundred blocks may not be computationally feasible.

In Wu [38], it has been argued that a key advantage of network coding is its inherent ability to adapt to network dynamics, both to ergodic changes such as random packet loss and to non-ergodic changes such as link failures. Koetter et al. [18] also discussed robust networking and analyzed the resilience of network coding to non-ergodic link failures. Lun et al. [22] theoretically analyzed the benefit of network coding on a directed acyclic hypergraph with lossy links. Acedanski et al. [1] showed through analysis that with network coding, a peer downloading a file may randomly connect to fewer other peers to retrieve the entire file. In this thesis, we consider dynamic P2P networks and analyze how peer churn could adversely affect block availability, degrading download performances in a P2P content distribution session. Based on that, we focus on quantifying the resilience gain of network coding in P2P networks under variable computational costs involved in encoding and decoding. To the best of our knowledge, this is a first attempt to quantitatively evaluate the resilience of network coding to block losses due to peer departures, whereas most previous analytical work focused on block losses due to erasure links.

We further take advantage of the block diversity and failure tolerance of random network coding, and apply it in a different setting of statistics data collection from a large number of peers. The problem of statistics collection in distributed systems has been approached with various methods before. Stutzbach et al. [29] design a crawler to
capture snapshots of Gnutella network, which focuses on increasing snapshot accuracy by increasing crawling speed. Such an approach leverages the two-tier topology of Gnutella networks and is difficult to be generalized to arbitrary network topologies. NetProfiler [24] is a peer-to-peer infrastructure proposed for profiling wide-area networks, which aggregates information along DHT-based attribute hierarchies and thus may not adapt well to high peer churn rates. Astrolabe [32] aggregates information for distributed system monitoring, with gossip-based information distribution and replication. Echelon [35] leverages the power of network coding to collect application-specific measurements on each peer, and disseminate them to other peers in a coded form.

Our work differs from all previous work in two aspects. First, our algorithms aim at providing Quality of Service (QoS) guarantees in face of peer dynamics and fluctuating traffic. We demonstrate such QoS guarantee using both theoretical analysis and simulations, while most previous work is only experimental. Furthermore, we give a quantitative description of the fault-tolerance of random network coding in a dynamic network, and demonstrates through a random bipartite graph model that the proposed algorithm guarantees good performance with respect to throughput and delay. Such a modeling effort with theoretical rigor helps us to draw more insights than mere experimental work can do and offers guidelines on choosing system parameters in a wide range of settings.

Network coding has also been introduced for distributed storage in [9, 10] for a sensor network scenario. The idea has subsequently been extended to fault-tolerant P2P storage [8]. In this thesis, we utilize a similar idea to spread coded blocks across the network, from which servers pull data proactively. However, our focus is not to maintain a long-lasting distributed storage system. Instead, our goal is to buffer data temporarily in a
distributed fashion so that the server can retrieve data with acceptable delay when its
download capacity is not large enough to collect all the data simultaneously.
Chapter 3

P2P Content Distribution with Network Coding

In this chapter, we present a brief background on the main characteristics of mesh-based P2P bulk content distribution systems and the application of segment-based network coding in these systems.

3.1 Mesh-based P2P Content Distribution

A typical P2P bulk content distribution system such as BitTorrent [6], [26] breaks a single large file (typically several 100 MBytes or several GBytes large) of common interest into fine-granularity data blocks (e.g., each of size 256KB). The participating downloaders which are interested in this particular file are organized into a randomized overlay mesh network, exchanging the blocks with one another. Each peer tries to retrieve the entire file by downloading different blocks of the file from different peers.

A peer is referred to as a seed or a server if it has a complete copy of the content of
interest, otherwise it remains a *downloader* or a *user peer*. Each peer does not have to wait until it has become a seed to upload to other peers. During the download process of a use peer, the significant amount of contribution not only from seeds but also from other downloaders greatly expedite the download process. The original content provider can be any peer in the network. However, the content distribution session may proceed even if the content provider has left the network, as long as all the data blocks constituting the file are still available in the network.

In BitTorrent, there is a centralized software called the *tracker*, which is responsible for topology maintenance. A peer interested in a certain file will first contact the tracker of the file, which will return a list of other peers (usually randomly selected) for this peer to connect to. Each peer maintains connection with 40–80 other peers, which form its *neighborhood*. Data exchanges may only occur between a peer and its neighbors. Normally, a peer is only allowed to upload to a small number (*e.g.*, less than 5) of its neighbors simultaneously, which are called its *downstream peers*, since excessive concurrent uploading may negatively affect throughput [39]. There are a variety of tricks on how to choose downstream peers at a particular time. However, since the focus of this thesis is to analyze the effect of network coding, we will use random neighbor selection and random downstream peer selection as the reference, which largely capture the randomization idea heavily used in mesh-based P2P content distribution systems.

As each peer makes its local decision on what block to transmit to its downstream peers, it is very important from a global perspective to maintain a uniform block distribution in the entire network in order to avoid the possible delay in obtaining “rare” blocks and thus to ensure satisfactory download times. Moreover, when the blocks are
distributed uniformly, it is intuitively sound that the content loss will be kept at minimum upon peer departures following a random pattern. To ensure a uniform block distribution, BitTorrent has a so-called local-rarest-first policy, by which a peer is most likely to upload to its downstream peers a block that is the rarest within the scope of its neighborhood. Peers exchange buffer maps with their neighbors to obtain the block availability information at other peers.

Unfortunately, such a policy is not perfect in that its communication overhead is large due to buffer map exchanges and the local view of a peer may not be impartial enough to represent the global block distribution in the network. Furthermore, it has been reported in various measurement studies [3, 30, 25, 14, 16] that a large number of downloaders in a content distribution session may frequently abort the session before they finish downloading, making the system highly dynamic. As the system has no knowledge on which blocks may be lost as peers leave, some blocks will become rare while other blocks still prevail the network. This is sometimes known as the curse of “rare blocks”, which can often be seen in real BitTorrent networks. Therefore, the dynamic memberships of the peers make it a challenging issue to avoid the curse of “rare blocks”, and furthermore, to ensure a uniform block distribution which benefits the download efficiency.

3.2 Segment-based Network Coding

As coded blocks are equally useful, we may expect the curse of “rare blocks” may indeed be solved by network coding. We now briefly introduce the mechanism of generation or segment-based randomized network coding [5], which is also referred to as group network coding in Avalanche [13], [11], [12]. When a large file of size $F$ bytes (usually of the order of several hundreds of Megabytes or several Gigabytes) is to be broadcast to every online
3.2. SEGMENT-BASED NETWORK CODING

In the network, the content is segmented into $G$ segments, each of which are further broken into $m$ blocks. The number of blocks contained in a segment is referred to as the segment size. Apparently, there are $M = G \cdot m$ different original blocks in the content, each of which has a fixed number of bytes $k = F/M$, referred to as the block size.

When segment-based network coding is used, a random linear code (RLC) is applied to the blocks within a same segment. Assume segment $i$ has original blocks $B^{(i)} = [B_1^i, B_2^i, \ldots, B_m^i]$, then a block $b$, whether it be a coded block or an original block, is said to belong to segment $i$ if it can be represented by a linear combination of $[B_1^i, B_2^i, \ldots, B_m^i]$ in the Galois field $GF(2^8)$. Coding operation is not limited to the source: if a peer (including the source) has $l$ ($l \leq m$) coded blocks from segment $i$ $[b_1^i, b_2^i, \ldots, b_l^i]$, when serving another peer $p$, it independently and randomly chooses a set of coding coefficients $[c_1^p, c_2^p, \ldots, c_l^p]$ in the Galois field $GF(2^8)$, and then encodes all the blocks it has received from segment $i$, and produces one coded block $x$ of $k$ bytes: $x = \sum_{j=1}^{l} c_j^p \cdot b_j^i$.

A coded block $x$ is self-contained, in that the coding coefficients used to encode original blocks to $x$ are embedded in the header of the coded block. Since only blocks in the same segment are allowed to be linearly combined and the embedded coding coefficients are related to the original blocks of that segment, we need a total of $m$ of coefficients, leading to a header overhead of $m$ bytes per coded block. These $m$ coding coefficients to be embedded can easily be computed by multiplying $[c_1^p, \ldots, c_l^p]$ with the $l \times m$ matrix of coding coefficients embedded in the incoming blocks of segment $i$ $[b_1^i, b_2^i, \ldots, b_l^i]$.

As soon as a peer has received a total of $m$ coded blocks from segment $i$ $x = [x_1^i, x_2^i, \ldots, x_m^i]$, it starts decoding segment $i$. To decode segment $i$, it first forms an $m \times m$ matrix $A_i$, using the $m$ coding coefficients embedded in each of the $m$ coded blocks in segment $i$ it has received. Each row in $A_i$ corresponds to $m$ coded coefficients of one
3.3. THE APPLICATION OF NETWORK CODING IN P2P CONTENT DISTRIBUTION

-coded block. It then recovers the original blocks of segment \( i \) \( B^{(i)} = [B_1^i, B_2^i, \ldots, B_m^i] \) by:

\[
B^{(i)} = A_i^{-1}x^T
\]

In such a decoder, we first need to compute the inverse of the \( m \times m \) coefficient matrix \( A_i \) using Gaussian elimination, which requires \( O(m^3) \) operations (or \( O(m^2) \) operations per input block). To obtain the original \( m \) blocks \( B^{(i)} \), it then needs to multiply \( A_i^{-1} \) and \( x \), which takes \( m^2 \cdot k \) multiplications of two bytes in \( \text{GF}(2^8) \), which runs in time \( O(m^2) \) (or \( O(m) \) operations per input original block). It turns out the latter cost dominates the overall decoding time, although the first phase has a higher computational complexity in terms of \( m \). This is because the cost of the latter phase also depends on the block size \( k \), which is usually of the order of Kilobytes. Naturally, as the segment size \( m \) increases from one to \( M \), the overall decoding complexity increases.

3.3 The Application of Network Coding in P2P Content Distribution

Network coding can greatly simplify protocol design in P2P content distribution systems. Apparently, if we apply network coding across all existing blocks (segment size \( m = M \)), peers only need to spread coded blocks “blindly” without exchanging buffer maps with their neighbors, as coded blocks are all equally useful. When network coding is performed in segments and applied to P2P networks, the original block selection scheme is now replaced by a similar segment selection scheme; each peer makes decisions on which segment to encode and transmit to its downstream peer rather than on which original
3.3. THE APPLICATION OF NETWORK CODING IN P2P CONTENT DISTRIBUTION

block to transmit. Thus, a peer’s buffer map now specifies how many blocks the peer has for each segment.

With respect to enhancing block availability in P2P networks, segment selection schemes such as the local-rarest-first policy and network coding interact in a complicated way. It is therefore impossible to assess the benefit of network coding without reference to segment selection schemes. However, in order to analytically reveal the effect of network coding, we will use random segment selection in addition to aforementioned random neighbor and downstream peer selection as the reference. As experimental results from Avalanche [13], [11], [12] show that simple random downstream peer selection and random segment selection mechanisms are sufficient to achieve satisfactory download performances in a network coding system, it is reasonable to believe that the performance will be even better if network coding is to be combined with more advanced segment selection and neighbor selection schemes.

As the segment size $m$ decreases from $M$ to 1, the system is effectively moving in the design space from a pure network coding system to a non-coding system with random block selection. However, as we use smaller segments for network coding, there is a penalty on the block availability in the session. Intuitively, without network coding, if peer churn is severe, departing peers may take away important blocks that have already become rare in the network, inducing much delay for peers to obtain those rare blocks. On the other hand, with pure network coding performed across the entire file, all the blocks are equally useful and there is no such a problem as locating the rare blocks, and the entire content can still persist in face of peer churn as long as there are an enough number of coded blocks in the network for the content to be reconstructible.

As a major contribution of this thesis, we are able to quantify the resilience gain of
network coding in terms of mitigating the variation of the distribution of different blocks and enhancing content persistence in a dynamic network, as the segment size $m$ varies. Such a quantitative description of the resilience gain of network coding will guide system design whenever computational complexity is an important issue.
Chapter 4

On the Resilience-Complexity Tradeoff of Network Coding

In this chapter, we quantitatively evaluate the resilience gain of network coding in dynamic P2P networks as the number of blocks in a segment for coding varies. In Sec. 4.1, we present our system model, formulate the problem and outline the main theoretical findings. In Sec. 4.2, we derive a linear system of differential equations, based on which we obtain asymptotical results in Sec. 4.3 regarding steady-state block availability and block variations when data sources are always available. We then characterize content availability in the absence of data sources in Sec. 4.4. We present simulation results to corroborate our theoretical findings in Sec. 4.5 and summarize the chapter in Sec. 4.6.

4.1 Model Description

In our analysis, we consider dynamic P2P network models which encompass major characteristics of P2P content distribution protocols. Assume an overlay network is formed
by a set of peers, each with an average upload capacity of $\mu$ and a separate downlink of sufficiently large capacity. Let $N$ and $N_s$ denote the number of online downloaders and the number of online seeds respectively. If each block has a size of $k$ bytes, a peer can upload at a capacity of $\tilde{\mu} = \mu/k$ blocks per unit time on average. We consider random target downloader selection and random useful block transmission, which are typical in network-coded content distribution systems such as Avalanche [13].

Specifically, whenever a peer, say peer $A$, is to upload a block, at rate $\tilde{\mu}$ it randomly chooses a downstream downloader peer from among its neighbors to serve. The neighbors are assumed to be uniform samples of the entire network and can be time-varying. Subsequently, the upload of the block is performed in three steps: segment reconciliation, encoding, and transmission. First, peer $A$ will compare its buffer with that of the chosen downstream peer $B$ to find out which segments it has are needed by peer $B$. Recall that a segment is needed by peer $B$ only if it has received less than $m$ linearly independent blocks from this segment. As a result, a difference set of segments between peer $A$ and $B$ is formed. Peer $A$ then randomly chooses a segment in the difference set, and encodes all the blocks it has received for that segment, and then transmits the encoded block to peer $B$.

We consider two models for peer dynamics, namely the replacement model and the Poisson arrival model. In the replacement model, we assume there are always $N$ downloaders and $N_s$ seeds simultaneously online, even though peer departures and joins occur constantly, as illustrated in Fig. 4.1. Each downloader has a random lifetime $L$ following an arbitrary yet general distribution $F_L(x)$ with a finite mean $\mathbb{L}$. Every new empty peer joins the network by replacing a departed downloader. This model for churn has previously been used in [20] to study the reliability of unstructured P2P networks.
There are always $N$ peers online. Every line above is a renewal process.

Online Peer 1
Online Peer 2
Online Peer N
Online Peer 3

an old peer leaves
a new peer joins immediately

$L$ denotes the peer lifetime. $A$ denotes the peer age at a given observation time $t_o$. Every new peer joins the session by replacing a departed peer.

The advantage of using the replacement model is that it can provide a straightforward understanding of the effect of churn. As far as the block availability is concerned, it is affected by both the number of online peers at a given time and how reliable these peers are. There will certainly be an insufficient amount of available data blocks if the number of online peers is insufficient. However, what we want to demonstrate in this chapter is that even if there are always a large number of peers simultaneously online, the dynamic nature of these peers can induce a significant variation among the distributions of different blocks and thus the “rare block” problem. The replacement model not only makes our analysis clean and concise, but more importantly, allows us to focus on the effect of peer churn on block availability variation, while decoupling the impact of the change in online peer number.

In the Poisson arrival model, we consider a more common scenario where downloaders enter the system in a Poisson process with rate $\lambda$. Similarly, each downloader has a random lifetime $L$ following an arbitrary yet general distribution $F_L(x)$. There are always $N_s$ seeds online. We will show both models yield similar results with regard to block availability in an asymptotic system. At a given observation time $t_o$, we use a random
variable $A$ to denote a peer’s age, which is defined as the period from the join of the peer to $t_o$. In our analysis, the peer age will be an important parameter.

To investigate the block availability as the network evolves, we are particularly interested in deriving the block distribution of each segment in all the downloaders, since seeds all have complete copies of the content. If we denote by a random variable $I$ the number of blocks each segment has in all the downloaders, our goal is to determine the distribution for $I$. The derivation of block distribution will allow us to evaluate content persistence in a dynamic network, or the risk that the content may become incomplete in the entire network due to peer departures. We will evaluate such a persistence that network coding offers with different segment sizes in Sec. 4.4.

To evaluate the variation of block availability and how this can affect the time to download the entire content, we define an important parameter called block variation as:

$$\gamma_I^2 = \frac{\text{var}(I)}{E^2\{I\}} = \frac{\sum_{i=0}^{\infty} i^2 p_i - (\sum_{i=0}^{\infty} i \cdot p_i)^2}{(\sum_{i=0}^{\infty} i \cdot p_i)^2}, \quad (4.1)$$

which is essentially the variance of the number of blocks each segment has divided by the square of its mean. $\text{Var}(I)$ alone does not serve as a good indicator of block variation, in that its effectiveness in assessing block distribution imbalance may be biased by the value of $E\{I\}$. Thus, we adopt the above definition of block variation, which is inspired by a typical fairness measure in resource allocation [27]. Block variation $\gamma_I^2$ has the property that its value always lies between 0 and $\infty$, and that it is 0 for the balanced block distribution when all the segments have the same number of blocks in the network.

As a major theoretical conclusion from our analysis, we have shown that, whether the replacement model or Poisson arrival model is assumed, as $N \to \infty$ or $\lambda \to \infty$, and $M \to \infty$, while $L$ remains finite, block variation $\gamma_I^2$ is inversely proportional to the
4.2. CHARACTERIZING ASYMPTOTIC SYSTEMS VIA DIFFERENTIAL EQUATIONS

Segment size $m$ in the steady-state network. This means as we use a larger segment size for coding, the blocks of different segments will be distributed in a more balanced way. We also show through simulations in Sec. 4.5 that in a network composed of a large number of highly dynamic peers, the average time for a long-lived peer to download the entire file is closely related to the block variation. Intuitively, the more significant the block variation is, the longer the download time each downloader has to experience, as the download process is delayed by obtaining those “rare” segments.

4.2 Characterizing Asymptotic Systems via Differential Equations

In this section, we first introduce a system of representations to characterize network states in terms of block availability. We then derive a set of differential equations to asymptotically characterize the system evolution under the replacement model where each departed peer is replaced by a newly joined empty peer. Finally, we extend the results to the Poisson arrival model. The main results concerning block availability and the resilience of network coding will be drawn from the derived differential equations in Sec. 4.3.

4.2.1 System State Representation

For a segment $s$, we use $\text{deg}(s)$ (degree of $s$) to denote the number of blocks $s$ has in all the downloaders. To characterize the distribution of $I$, we introduce the following notations. Since seeds always have complete copies of the content, we are only interested in the block statistics in downloaders:
\( n_i(t) \): the number of segments which have a degree of \( i \). \( \sum_{i=0}^{\infty} n_i(t) = G \).

\( p_i(t) \): the fraction of segments which have a degree of \( i \). \( p_i(t) = n_i(t)/G \).

\( r_i(t) \): the number of segments with a degree of at least \( i \). \( n_i(t) = r_i(t) - r_{i+1}(t) \).

\( s_i(t) \): the fraction of segments which have a degree of at least \( i \). \( s_i(t) = r_i(t)/G \).

Figure 4.2: An illustration for the notations \( p_i(t), s_i(t), n_i(t) \) and \( r_i(t) \).

These notations are illustrated in Fig. 4.2, where the vertical axis \( i \) represents the degree of the segments, while the horizontal axis represents segments sorted in the descending order of \( i \). Apparently, as the segment number \( G \to \infty \), \( p_i(t) \) will approach the probability mass function (PMF) of \( I \). These notations cover both non-coding and coding cases as the segment size \( m \) varies from 1 to \( M \). For simplicity, we may omit \( t \) in the following context. From Fig. 4.2, we also get the following simple facts: at time \( t \), 1) the total number of blocks in downloaders is \( Y(t) = \sum_{i=1}^{\infty} r_i(t) \), and 2) the average degree of a segment is \( \sum_{i=1}^{\infty} s_i(t) = Y(t)/G \), which equals to the area under the line in Fig. 4.2(b).

The system state could be represented by the vector \( R(t) = (r_0(t), r_1(t), \ldots) \), \( S(t) = (s_0(t), s_1(t), \ldots) \) or \( P(t) = (p_0(t), p_1(t), \ldots) \). According to the gossiping model mentioned
in Sec. 4.1, \( S(t+h), \forall h > 0 \), not only depends on \( S(t) \), but also depends on which peer has what particular subset of the blocks at time \( t \). Thus, \( S(t) \) are complicated processes which are extremely hard to characterize. However, under certain assumptions outlined in Sec. 4.2.2, \( S(t) \) will become a Markov process, whose limiting behavior as \( N \to \infty, M \to \infty \) converges to a deterministic process that can be represented by a set of differential equations. By characterizing \( S(t) \) using differential equations under these assumptions, we are able to derive the distribution of the block availability in the steady state, which was originally extremely hard to achieve because of the large state-space and the dynamic nature of peers. The validity of all the assumptions made are verified by the simulations in Sec. 4.5.

### 4.2.2 Deriving Differential Equations: Replacement Model

Our characterization for the replacement model follows the following path. First, we show under the replacement model, the total number of blocks in the network \( Y(t) \) tends to concentrate around a certain value \( (Y(t)/(N + N_s) \) stays constant) after the network has evolved for sufficiently long time. However, even though the total number of blocks remains the same, the number of blocks of each segment, which is characterized by \( S(t) \), can be changing in an intricate way due to block transfers and peer departures. Thus, we make some appropriate assumptions so that \( S(t) \) forms a density dependent jump Markov process, which can be asymptotically characterized by a set of differential equations.

To characterize \( Y(t) \) in steady state, we need the average peer age given as follows:

**Lemma 1** Let \( L \) denote a peer’s lifetime with mean \( \bar{L} \) and variance \( \sigma^2 \), and \( A \) denote a peer’s age. Under the replacement model, if \( N \) is large, as the observation time \( t_o \to \infty \),
the probability density function of $A$ is given by $f_A(x) = \frac{1 - F_L(x)}{L}$. Moreover, $E[A] = \bar{A} = (L^2 + \sigma^2)/2L$.

Remarks: A complete proof of this lemma can be found, for example, in [28]. It is easy to check that if the peer lifetime $L$ follows an exponential distribution, the distribution of peer age $A$ will be exactly the same as that of $L$. However, in general, the distribution of $A$ is different from that of $L$.

We now show that after the network has evolved for sufficiently long time, $Y(t)/(N + N_s)$ almost always remains constant. Let i.i.d. random variables $A_1, A_2, \ldots, A_N$ denote the ages of online downloaders. We consider the challenging case when the network consists only of short-lived peers, and assume a peer’s mean lifetime $L$ is finite. Since $\mu$ and $L$ are finite, as $N \to \infty$, $M \to \infty$, with very small probability will two online peers have large sets of overlapped blocks in their buffers. Thus, a peer can almost always transmit a useful block to another peer. As a result, the total download bandwidth in the network always equals to the total upload bandwidth $(N + N_s)\mu$. Moreover, we can use identical and mutually independent random processes $\tilde{\mu}_1(t), \tilde{\mu}_2(t), \ldots, \tilde{\mu}_N(t)$ to denote the actual download bandwidth (blocks/unit time) of online downloaders at time $t$. Assume $\tilde{\mu}_i(t)$ are wide-sense stationary and mean-ergodic. We also assume that all $\tilde{\mu}_i(t)$ have the same mean function $E\{\tilde{\mu}_i(t)\}$. Now we are ready to establish Lemma 2 using the strong law of large numbers.

**Lemma 2** Let $Y(t)$ denote the total number of blocks in downloaders from all the segments at a given time $t$. Under the replacement model, as $N \to \infty$, $M \to \infty$, $N = \alpha M$, $N_s = \alpha_s M$, where $\alpha, \alpha_s$ are finite constants,

$$\lim_{N \to \infty} \frac{Y(t)}{N + N_s} = \mu \bar{A} \text{ with probability 1},$$
4.2. CHARACTERIZING ASYMPTOTIC SYSTEMS VIA DIFFERENTIAL EQUATIONS

for $t$ sufficiently large.

Proof: At any given time $t$, we have

$$\mathbf{Y}(t) = \sum_{i=1}^{N} \int_{0}^{A_i} \tilde{\mu}_i(\tau) d\tau.$$ 

According to the strong law of large numbers,

$$\lim_{N \to \infty} \frac{\mathbf{Y}(t)}{N} = \mathbf{E}\{ \int_{0}^{A_i} \tilde{\mu}_i(\tau) d\tau \} \text{ with probability } 1.$$

We have

$$\mathbf{E}\{ \int_{0}^{A_i} \tilde{\mu}_i(\tau) d\tau \} = \mathbf{E}\{ \int_{0}^{A_i} \tilde{\mu}_i(\tau) d\tau | A_i \}$$

$$= \mathbf{E}\{ \int_{0}^{A_i} \mathbf{E}\{ \tilde{\mu}_i(\tau) \} d\tau | A_i \}$$

$$= \mathbf{E}\{ A_i \cdot \mathbf{E}\{ \tilde{\mu}_i(\tau) \} \}$$

$$= \overline{A} \cdot \mathbf{E}\{ \tilde{\mu}_i(\tau) \}$$

$$= \frac{\sum_{j=1}^{N} \tilde{\mu}_j(\tau)}{N}, \text{ with probability } 1 \text{ as } N \to \infty$$

$$= \frac{(N + N_s)\tilde{\mu}\overline{A}}{N},$$

where the second equality holds because $\tilde{\mu}_i(\tau)$ is mean-ergodic, and the last equality holds because $\tilde{\mu}_i(\tau)$ are wide-sense stationary and the total download bandwidth in the network always equals to the total upload bandwidth $(N + N_s)\tilde{\mu}$. Therefore, we have shown $\lim_{N \to \infty} \frac{\mathbf{Y}(t)}{(N + N_s)} = \mu\overline{A}$ with probability 1 for any $t$ sufficiently large. $\square$

From Lemma 2, we can see that, if the file is segmented into a large number of segments, that is $G$ is large enough, the average number of blocks each segment has in
all the downloaders is

\[ \sum_{i=1}^{\infty} s_i(t) = \frac{Y(t)}{G} = \frac{(N + N_s)\bar{\mu}\bar{A}}{G} = (\alpha + \alpha_s)m\bar{\mu}\bar{A}, \]

where \( \bar{A} \) is given by Lemma 1.

We now characterize the evolution of \( S(t) \) when \( t \) is sufficiently large. Note that there are three factors that contribute to the change of \( S(t) \), namely seed uploads, downloader uploads, and downloader departures. No matter what distribution the time for a peer to upload a block follows, the aggregate uploads from all peers form a Poisson process of rate \( (N + N_s)\bar{\mu} \), as N \( \to \infty \), \( N_s \to \infty \) (refer to [17] pp. 221). At each upload of the aggregate upload process, it is a random peer in the network that uploads, because the upload times of all peers follow a general distribution. Similarly, no matter what distribution the lifetime of a peer \( L \) follows, the aggregate departures of downloaders will form a Poisson process with rate \( N/\bar{L} \). Even with the Markovian property of uploads and download departures, it is still difficult to write the transition rates for \( S(t) \) due to the dependency among the actions of different peers. To further introduce analytical tractability that makes \( S(t) \) a Markov process with linear intensities, we make the following linear approximations on the effect of random downloader selection and random useful block selection, and the effect of downloader departures:

**Assumption 1 (Linear Approximation for Uploads)** Whenever a seed uploads a useful block, the effect is approximated by choosing uniformly from all the segments a random segment \( s \) and increasing \( \deg(s) \) by one; whenever a downloader uploads a useful block, the effect is approximated by choosing from all the segments a random segment \( s \) with probability \( \deg(s)/\sum_s \deg(s) = \deg(s)/Y(t) \) and increasing \( \deg(s) \) by one.
Remarks: Since blocks of all segments are distributed in seeds in a balanced way, and blocks of all segments are largely needed by downloaders due to a finite $\mathcal{L}$, each segment has an equal chance of being chosen during the upload from a seed. However, when a download uploads, each segment does not enjoy an equal chance of being chosen, as blocks of different segments are not distributed in a balanced way in downloaders. On the contrary, the more blocks a segment has in all the downloaders, the more frequently it will be chosen and encoded for transfer. We verify the validity of this approximation in Sec. 4.5 through simulations.

Let $e_i$ denote a unit vector of the same dimension as $R(t)$ with its $i$th element being 1, and all other elements being 0. Now it is easy to write the intensities $q_{R,R+e_i}^{(G)}$ for the transitions on $R$ due to uploads, when the total number of segments is $G$:

$$R \rightarrow R + e_i, \quad q_{R,R+e_i}^{(G)} = N_s\tilde{\mu}p_{i-1} + N\tilde{\mu} \cdot \frac{(i-1)n_{i-1}}{Y} = N_s\tilde{\mu}p_{i-1} + \frac{\alpha}{\alpha + \alpha_s} \frac{(i-1)n_{i-1}}{A}$$

for all finite $i$, where $N/M = \alpha$, $N_s/M = \alpha_s$, and the second equality is due to Lemma 2.

The rationale behind is that seeds upload a block at a total rate of $N_s\tilde{\mu}$ with all segments being chosen for encoding equally likely, while downloaders upload a block at a total rate of $N\tilde{\mu}$ with segments being chosen according to Approximation 1. Note that in (4.3), the actual upload rates of peers are taken to be the capacities of the peers, which have an average value of $\tilde{\mu}$, because the lifetimes of peers are finite as $M \rightarrow \infty$ so that almost no peer will have its entire set of blocks as a subset of another peer’s block set. Furthermore, almost no linear dependency will occur when a peer is uploading to another with network coding, since according to Lemma 2.1 in [7], a random linear combination
of all the blocks from the same segment at a peer \( p \) is useful to another randomly chosen peer in the network with probability at least \( 1 - 1/q \) if network coding is done in \( \mathbb{F}_q \).

And this argument is true regardless of whether peer \( p \) is a seed or a downloader. In this chapter, the field size \( q \) has been assumed to be \( 2^8 \).

Let us consider blocks losses due to downloader departures. The total number of downloader departures in a small interval \( \Delta t \) is \( N \Delta t / \bar{L} \). Each downloader downloads at a rate of \( \frac{(N + N_s)\tilde{\mu}}{N} \) on average. Upon departure, a downloader takes away \( \bar{L} \cdot \frac{(N + N_s)\tilde{\mu}}{N} \) blocks on average. Therefore, we lose \( \frac{N \Delta t}{\bar{L}} \cdot \frac{(N + N_s)\tilde{\mu}}{N} = (N + N_s)\tilde{\mu} \Delta t \) blocks in \( \Delta t \) in total. We make a similar linear approximation to block losses of different segments:

**Assumption 2 (Linear Approximation for Downloader Departures)** As \( N \to \infty, N_s \to \infty, M \to \infty, \) among all the blocks lost due to downloader departures in a small time interval \( \Delta t = O(1/\sqrt{N}) \), the proportion of the blocks that segments of degree \( i \) lose is \( \frac{i \cdot n_i}{\sum_i i \cdot n_i} = \frac{i \cdot n_i}{\sum_i i \cdot n_i} = \frac{i \cdot n_i}{Y(t)} \).

Remarks: This essentially means that the more blocks a segment has in all the downloaders, the more blocks it loses due to peer departures. The choice of \( \Delta t = O(1/\sqrt{N}) \) ensures each segment can lose at most one block in \( \Delta t \). The validity of this approximation is verified in Sec. 4.5 through simulations.

Thus, in \( \Delta t \), the total block loss of segments of degree \( i \) is

\[
(N_s + N)\tilde{\mu} \Delta t \cdot \frac{i \cdot n_i}{Y} = \frac{i \cdot n_i}{A} \cdot \Delta t
\]

As \( N \to \infty \) while \( \bar{L} \) remains finite, such a block loss pattern will be equivalent to having
4.2. CHARACTERIZING ASYMPTOTIC SYSTEMS VIA DIFFERENTIAL EQUATIONS

individual blocks of different segments lost by the following transitions:

\[ \mathbf{R} \rightarrow \mathbf{R} - e_i, \quad q^{(G)}_{\mathbf{R}, \mathbf{R} - e_i} = \frac{i \cdot n_i}{A} \]  

(4.4)

for all finite \( i \).

Let \( S^{(G)}(t) = \{ s_0^{(G)}, s_1^{(G)}, s_2^{(G)} \ldots \} := R^{(G)}(t)/G \) denote the normalized process when there are \( G \) segments. Note that the intensities for \( \mathbf{R} \) can be written in the form of

\[ q^{(G)}_{\mathbf{R}, \mathbf{R} + l} = G \cdot \beta_l(\frac{\mathbf{R}}{G}) = G \cdot \beta_l(S), \quad l = +e_i, -e_i \quad \text{for all finite} \ i, \]

where

\[ \beta_{+e_i}(S) = \frac{N_s \tilde{\mu} \mu_{i-1}}{G} + \frac{\alpha}{\alpha + \alpha_s} \cdot \frac{(i - 1)n_{i-1}}{AG} \]

\[ = \alpha_s m \tilde{\mu} \mu_{i-1} + \frac{\alpha}{\alpha + \alpha_s} \cdot \frac{(i - 1)p_{i-1}}{A}, \]

(4.5)

and

\[ \beta_{-e_i}(S) = \frac{i \cdot n_i}{AG} = \frac{i \cdot p_i}{A}. \]  

(4.6)

Thus, \( S^{(G)}(t) \) forms a density dependent jump Markov process (refer to [19], pp.51). We set

\[ F(x) = \sum_l l \beta_l(x) = \sum_i e_i(\beta_{+e_i}(x) - \beta_{-e_i}(x)). \]

(4.7)

By Kurtz Theorem (Theorem 8.1 in [19]), for \( S^{(G)}(t) \) to converge to a deterministic fluid, we need that \( S^{(G)}(0) \) converges to a certain value \( S(0) \) that does not depend on \( G \). Since \( S^{(G)}(0) = (1, 0, 0, \ldots) \), this condition is satisfied. Moreover, we need the boundedness and Lipschitz continuity of \( F(x) \). These conditions are guaranteed in our model by the
finiteness of $\sum_{i=1}^{\infty} s_i(t) \to (\alpha + \alpha_s) m \bar{\mu} \bar{A}$ in probability, which remains finite due to a finite $\bar{L}$. Therefore, by Kurtz Theorem (Theorem 8.1 in [19]), under the above three conditions that are satisfied here, $S^{(G)}(t)$ converges almost surely to the deterministic fluid $S(t) = \{s_0, s_1, s_2, \ldots\}$ for large $G$, which is governed by the following equation:

$$S(t) = S(0) + \int_0^t F(S(u)) \, du, \quad t \geq 0,$$

(4.8)

Substituting (4.7), (4.5), and (4.6) into (4.8), we have shown that under the condition that $M \to \infty$, $N \to \infty$, $N_s \to \infty$, $N/M \to \alpha$, $N_s/M \to \alpha_s$, and $m$ remains finite, the behavior of $S(t)$ in replacement model for $t$ sufficiently large converges to the following system of ODEs with an arbitrarily small error:

$$\begin{cases}
A \cdot \frac{ds_i}{dt} &= \alpha_s m \bar{\mu} \bar{A} \cdot p_{i-1} + \frac{\alpha}{\alpha + \alpha_s} (i - 1) p_{i-1} - i \cdot p_i, \quad \forall \ i \geq 1 \\
 s_0 &= 1
\end{cases}$$

(4.9)

In addition, we have also separately considered the effect of the block size $k$ on the system evolution. Intuitively speaking, if the block size $k$ is large, when a peer departs while still uploading a block, more data will be lost due to the lack of granularity in the upload process. However, it turns out that the block size will almost not affect the differential equations (4.9), as long as $\mu \bar{A} \gg k$, which is satisfied in usual cases ($\mu \bar{A} \gg k$ does not conflict with the finiteness of $\bar{\mu} \bar{A}$).

4.2.3 Deriving Differential Equations: Poisson Arrival Model

Let us now establish a system of differential equations for the Poisson arrival model. If the number of online downloaders $N$ is not fixed, and downloaders enter the system according to a Poisson process with rate $\lambda$, it turns out the distribution of $A$ can be
4.2. CHARACTERIZING ASYMPTOTIC SYSTEMS VIA DIFFERENTIAL EQUATIONS

determined in the same formula as in Lemma 1. We state this result in the following
lemma, the proof of which can be found in [31] Lemma 3.

**Lemma 3** Under the Poisson arrival model with rate $\lambda$, as the observation time $t_o \to \infty$,
the probability density function of $A$ is given by $f_A(x) = [1 - F_L(x)]/\mathcal{L}$, regardless of $\lambda$.
Moreover,

$$E[A] = \mathcal{A} = \frac{\mathcal{L}^2 + \sigma^2}{2\mathcal{L}}.$$

Similarly, if the joining rate $\lambda$ of the peers are large enough, the network can reach a
stationary stage, where the total number of blocks in the network almost always remains
the same:

**Lemma 4** Let $Y(t)$ denote the total number of blocks in downloaders from all the seg-
ments at a given time $t$. Under the Poisson arrival model, as $\lambda \to \infty, M \to \infty$,
$\lambda\mathcal{L} = \alpha'M, N_s = \alpha_sM$, where $\alpha', \alpha_s$ are finite constants,

$$\lim_{\lambda \to \infty} \frac{Y(t)}{\lambda\mathcal{L} + N_s} = \tilde{\mu}A \text{ with probability 1}, \quad (4.10)$$

for $t$ sufficiently large.

**Proof:** We first show $\lim_{\lambda \to \infty} N(t)/\lambda\mathcal{L} = 1$ and then $\frac{Y(t)}{N(t)+N_s} \to \tilde{\mu}A$ to prove the
theorem. Since the arrival process is Poisson, it can be split into $n$ i.i.d. Poisson processes,
each with rate $\lambda_0 = \lambda/n$, and the system can be split into $n$ sub-systems, each being
a $M/G/\infty$ queue. Let $N_i(t)$ be the number of peers in the $i$th subsystem. To prove
$\lim_{\lambda \to \infty} N(t)/\lambda\mathcal{L} = 1$ for large $t$, we only need to show that $\lim_{n \to \infty} N(t)/n\lambda_0\mathcal{L} = 1$ with
probability 1 for any positive and finite $\lambda_0$. This is true because

$$\lim_{n \to \infty} \lim_{t \to \infty} \frac{N(t)}{n\lambda_0\mathcal{L}} = \lim_{n \to \infty} \lim_{t \to \infty} \frac{\sum_{i=1}^{n} N_i(t)}{n} \cdot \frac{1}{\lambda_0\mathcal{L}} = \lim_{t \to \infty} \frac{E\{N_i(t)\}}{\lambda_0\mathcal{L}} = \frac{\lambda_0\mathcal{L}}{\lambda_0\mathcal{L}} = 1,$$
where the second equality holds w.p. 1 by the strong law of large numbers, and the third equality holds due to Little’s Theorem. Hence, we have
\[
\lim_{t \to \infty} \lim_{\lambda \to \infty} \frac{Y(t)}{\lambda L + N_s} = \lim_{t \to \infty} \lim_{N(t) \to \infty} \frac{Y(t)}{N(t) + N_s} \cdot \lim_{t \to \infty} \lim_{\lambda \to \infty} \frac{N(t) + N_s}{\lambda L + N_s} = \tilde{\mu} A \cdot 1 \text{ w.p. 1},
\]
where \(\lim_{t \to \infty} \lim_{N(t) \to \infty} \frac{Y(t)}{N(t) + N_s}\) can be shown to approach \(\tilde{\mu} A\) following the same argument as in the proof of Lemma 2 by replacing \(N\) with \(N(t)\). Thus, we have shown that
\[
\lim_{\lambda \to \infty} \frac{Y(t)}{\lambda L + N_s} = \tilde{\mu} A \text{ with probability 1 for large } t. \tag{4.11}
\]

A system of differential equations can be established in a similar way for the Poisson arrival model. **Under the condition that** \(M \to \infty, \lambda \to \infty, N_s \to \infty, \lambda L / M \to \alpha', N_s / M \to \alpha_s, \text{ and } m \text{ remains finite},\)** the block distributions under the Poisson arrival model are characterized by the following system of ODEs:

\[
\begin{aligned}
A \cdot \frac{d s_i}{dt} &= \alpha_s m \tilde{\mu} A \cdot p_{i-1} + \frac{\alpha'}{\alpha' + \alpha_s} (i - 1) p_{i-1} - i \cdot p_i \quad \forall \ i \geq 1 \\
 s_0 &= 1
\end{aligned}
\tag{4.11}
\]

The derivation of (4.11) follows the same approach as in the derivation of (4.9). The only difference here is we replace \(N\) in the replacement model with \(\lambda L\) and apply Lemma 4 instead of Lemma 2 when we derive (4.11).

### 4.3 Steady-State Block Availability

Based on the differential equations derived in Sec. 4.2, in this section, we determine the block availability distribution and block variation in steady-state networks for both the
reduction model and Poisson arrival model. Block variation is an important performance indicator that is closely related to download times, which we will show through simulations in Sec. 4.5, while the derivation of block distribution allows us to evaluate the content persistence in a dynamic network in Sec. 4.4 as the segment size for network coding varies. We will show that the steady-state block variation \( \gamma^2 \) is inversely proportional to the segment size \( m \) for network coding, and that the block distribution follows a form of negative binomial distribution, which can be well approximated by a Gaussian distribution.

### 4.3.1 Block Distribution

Assume all the \( N_s \) seeds are online while downloaders are churning. Let us first consider the replacement model in its steady state. Denote the steady-state solutions to (4.9) by \( \overline{p}_0, \overline{p}_1, \ldots, \overline{p}_G \). By setting \( \frac{d\overline{p}_i}{dt} = 0 \) in (4.9), we get \( \overline{p}_i = [B + \beta (i - 1)] \cdot p_{i-1}/i \), where

\[
B = \alpha_s m \bar{\mu} A, \quad \beta = \frac{\alpha}{\alpha + \alpha_s}.
\]

Hence, the steady-state block distribution is given by,

\[
\overline{p}_i = \overline{p}_0 \cdot \prod_{j=1}^{i} \left( \beta + \frac{B - \beta}{j} \right), \quad \text{for} \ i \geq 1, \quad (4.12)
\]

where \( \sum_i \overline{p}_i = 1 \). It is in general very difficult to obtain closed form solutions for \( \overline{p}_i \). However, under certain mild conditions appropriate for engineering purposes, we can convert (4.12) into its closed form as follows:

**Theorem 5 (Steady-State Block Distribution for Replacement Model)** Let \( B = \alpha_s m \bar{\mu} A \) and \( \beta = \frac{\alpha}{\alpha + \alpha_s} \). If \( \beta \) is a positive rational number, and \( \frac{B}{\beta} \in \mathbb{Z}^{++} \) (\( \mathbb{Z}^{++} = \{1, 2, \ldots\} \)), then in the steady state, the fraction of segments with a total number of \( i \)
blocks in downloaders is given by,

\[ p_i = \left(\frac{i + B}{\beta} - 1\right)\beta^i(1 - \beta)^\frac{B}{\beta}, \quad i = 0, 1, 2, \ldots \] (4.13)

In particular, the fraction of empty segments (segments which have no block in downloaders) is given by \( p_0 = (1 - \beta)^\frac{B}{\beta} \).

Remarks: Note that \( P\{I = i\} = p_i \) as \( M \to \infty \), while \( m \) remains finite. This theorem says under certain conditions on \( B \) and \( \beta \), \( I \) follows the negative binomial distribution, which can be well approximated by a Gaussian distribution when \( B \to \infty \). In practice, the requirement that \( \beta \) is a rational number is naturally satisfied, since both the seed number and the downloader number in the network are integers. Moreover, \( B \) is usually much larger than \( \beta \) and thus substituting \( B/\beta \) with the integer nearest to it will hardly affect the outcome of \( p_i \). Therefore, the value of \( p_i \) for all valid \( \beta \) and \( B \) can be approximated by using the nearest \( \beta \) and \( B \) satisfying the above conditions.

Proof: Based on the steady-state solutions to (4.9), let us further derive \( p_i \) in a simple closed form under the conditions set for \( B \) and \( \beta \). Let \( C = B/\beta - 1 \). We first prove \( p_i = \beta^i \binom{i + C}{i} \cdot p_0 \), for \( i \geq 1 \), and then determine \( p_0 \) from the fact that \( \sum_{i=0}^{\infty} p_i = 1 \). Let \( \beta = \frac{l}{n} < 1 \), where \( l, n \in \mathbb{N} \). Because \( C \) is a non-negative integer, we have

\[ p_i = p_0 \prod_{j=1}^{i} \left(\frac{l}{n} + \frac{B - \beta}{j}\right). \]

By straightforward induction, we can finally get

\[ p_i = p_0 \left(\frac{l}{n}\right)^i \cdot \frac{(i + C)!}{i!C!} = \beta^i \binom{i + C}{i} \cdot p_0. \] (4.14)
Let us now determine $\overline{p}_0$. Let $a_i = \overline{p}_i/\overline{p}_0 = \beta_i(i + C)$, $i = 0, 1, 2, \ldots$. Then we have $1/\overline{p}_0 = \sum_{i=0}^\infty a_i$. Hence, what we need to do is to determine $\sum_{i=0}^\infty a_i$. Because

$$\binom{i + C}{i} = \left(\binom{C + i - 1}{i}\right) + \binom{C + i - 1}{i - 1}, \quad i \geq 1,$$

we have

$$\sum_{i=0}^\infty a_i = \sum_{i=1}^\infty \beta^i \left(\binom{C - 1}{i} + i\right) + \beta \sum_{i=0}^\infty \beta^i \left(\binom{C + i - 1}{i - 1}\right) + a_0$$

If we view $a_i$ as a function of $C$ and let $f(C) = \sum_{i=1}^\infty a_i(C) = \sum_{i=1}^\infty \beta^i(i + C)$, we can get the following iteration for $f(C)$:

$$\begin{cases} (\beta - 1)f(C) + f(C - 1) + \beta = 0, & \text{for } C \geq 1 \\ f(0) = \sum_{i=1}^\infty \beta^i = \frac{\beta}{1-\beta} \end{cases}$$

By induction, it is not hard to get

$$f(C) = (1 - \beta)^{-(C+1)} - 1, \quad C = 0, 1, 2, \ldots.$$ 

Hence,

$$\sum_{i=0}^\infty a_i = f(C) + a_0 = (1 - \beta)^{-(C+1)}$$

Because $\sum_{i=0}^\infty \overline{p}_i = \overline{p}_0 \sum_{i=0}^\infty a_i = 1$, we get $\overline{p}_0 = (1 - \beta)^{\frac{1}{\beta}}$. Substituting $\overline{p}_0$ into (4.14) will yield $\overline{p}_i$. \hfill \Box

We have plotted the Cumulative Distribution Function (CDF) of $I$ in Fig. 4.3 according to Theorem 5 for different values of segment size $m$. When segment size $m$ is large enough, we can see the CDF of $I$ approaches a Gaussian distribution. A more important implication from Fig. 4.3 is that as segment size $m$ increases, there will be fewer rare
4.3. **STEADY-STATE BLOCK AVAILABILITY**

Figure 4.3: Steady-state block distribution (CDF of $I$) for different values of segment size $m$. The number of downloaders $N = 6000$, number of seeds $N_s = 100$, file size $F = 768$ MB, block size $k = 256$ KB, average peer age $A = 5$. There are an equal number of peers with upload bandwidth 2 MB/round, 512 KB/round, and 256 KB/round so that the average upload rate is $\tilde{\mu} = \mu/k = 3.7$.

blocks in the network, and thus the risk that the content becomes incomplete upon seed departures will be reduced. For example, for non-coding schemes ($m = 1$), 20% of all the segments do not have any blocks among downloaders in the steady state, which means that if all the seeds leave, 20% of the segments will become undecodable immediately. As the segment size $m$ increases, we observe an increasing trend in the mean as well as minimum number of blocks each segment has among downloaders. When segment size $m = 16$, the fraction of segments with less than 200 blocks among downloaders is roughly 0. This means even if seeds leave altogether, the content will not become incomplete immediately. In addition, the variation in block distribution is subdued as the segment size increases, which we will demonstrate later.

Finally, the steady-state block distribution for the Poisson arrival model can be derived in a similar way as for the replacement model. Denote the steady-state solutions to (4.11) by $\bar{p}_0, \bar{p}_1, \ldots, \bar{p}_G$. Under certain conditions, we can obtain closed form solutions for $\bar{p}_0, \bar{p}_1, \ldots, \bar{p}_G$. 
4.3. STEADY-STATE BLOCK AVAILABILITY

**Theorem 6 (Steady-State Block Distribution for Poisson Arrival Model)** Let $B = \alpha s m \tilde{\mu} A$ and $\beta' = \frac{\alpha'}{\alpha' + \alpha_s}$. If $\beta'$ is a positive rational number, and $\frac{B}{\beta'} \in \mathbb{Z}^+$ ($\mathbb{Z}^+ = \{1,2,\ldots\}$), then in the steady state, the fraction of segments with a total number of $i$ blocks in downloaders is given by,

$$p'_i = \left(i + \frac{B}{\beta'} - 1\right) \beta'^i (1 - \beta') \frac{B}{\beta'}, \quad i = 0,1,2,\ldots \quad (4.15)$$

In particular, the fraction of empty segments (segments which have no block in downloaders) is given by $p'_0 = (1 - \beta') \frac{B}{\beta'}$.

*Proof:* $p'_i$ can be determined in the same way as $p_i$ in Theorem 5. ☐

### 4.3.2 Block Variation and Discussions

Recall that *block variation* is defined as $\gamma^2 = \frac{\sigma^2}{\mu^2}$, where $\mu_I$ is the mean of $I$ and $\sigma^2_I$ is its variance. Intuitively, as the segment size increases from one to a big value, the total number of distinct segments in the network decreases. Hence, one may observe a less salient variation with respect to the block availability of different segments, and the problem of obtaining rare blocks is mitigated. We now quantitatively characterize the block variation in steady state. It turns out for both the replacement model and Poisson arrival models, there is a unique expression for block variation $\gamma^2_I$.

**Theorem 7 (Steady-State Block Variation)** Regardless of whether the replacement model or the Poisson arrival model is used, in steady-state network we have

$$\gamma^2_I = \frac{F}{N_s m \mu A}, \quad (4.16)$$
4.3. STEADY-STATE BLOCK AVAILABILITY

where $\overline{A}$ is the average age of a downloader given by Lemma 1. All the parameters are assumed to satisfy the conditions set in Theorem 5 or Theorem 6.

**Proof:** In the replacement model, $I$ in steady state follows negative binomial distribution as shown in (4.13). By straightforward deduction, we have $\mu_I = \frac{B}{1-\beta}$ and $\sigma^2_I = \frac{B}{(1-\beta)^2}$, where $B = \alpha_s m \overline{A}$, $\beta = \frac{\alpha}{\alpha + \alpha_s}$. Thus, $\gamma^2_I = \frac{\sigma^2_I}{\mu^2_I} = \frac{1}{\alpha_s m \overline{A}} = \frac{F}{N_s m \overline{A}}$. Similarly, for the Poisson arrival model, we have $\mu_I = \frac{B}{1-\beta'}$ and $\sigma^2_I = \frac{B}{(1-\beta')^2}$, where $\beta' = \frac{\alpha'}{\alpha' + \alpha_s}$. Thus, $\gamma^2_I = \frac{\sigma^2_I}{\mu^2_I} = \frac{1}{\overline{A}^2} = \frac{F}{N_s m \mu A}$, which is the same as in the replacement model. 

Let us now discuss the important insights behind Theorem 7. First, we can see that network coding helps to ensure a balanced distribution of blocks from different segments in the network. When segment size $m$ is large, block variation $\gamma^2_I$ approaches zero, which is the case of balanced block distribution. In contrast, in the non-coding case ($m = 1$), there is always a certain degree of block variation, which is inversely proportional to average peer age $\overline{A}$. Specifically, the more severe is the peer churn, the more salient is block variation and thus the “rare block” problem.

As segment size $m$ increases from 1 to $M$, the block variation is subdued. Thus, Theorem 7 has demonstrated that network coding indeed helps to mitigate the “rare block” problem in a dynamic network, and such a resilience to peer churn grows with the segment size $m$. Furthermore, as block variation is inversely proportional to segment size $m$, there can be a sweet spot in the curve of resilience-complexity tradeoff, which suffices to yield the major benefit of network coding at the cost of an acceptable coding complexity. We will show this in Sec. 4.5.

It is quite counter-intuitive that the block size $k$ (Bytes) does not affect block variation at all. One might believe as block size $k$ is increased, the total number of blocks in the content is reduced, and thus the variation with regard to block distribution may be
subdued. However, as $k$ increases, blocks will be disseminated at a lower rate ($\bar{\mu} = \mu/k$ decreases) given limited upload bandwidth at each peer. Both effects counteract with each other, resulting in no change in block variation.

The content size $F$ (Bytes) and the number of online seeds $N_s$ are also critical parameters that affect the imbalance of the availability of different blocks. The lack of seeds will increase such an imbalance, not because seeds hold more blocks than downloaders, but because the upload behavior of seeds are fundamentally different from that of downloaders, as we have pointed out through the analysis in Sec. 4.2. It is also worth noting that the number of online downloaders $N$ does not affect block variation at all. This is the reason why we have the same expression of block variation for both the replacement and Poisson arrival models.

4.4 Transient Behaviours

In a dynamic network, as peers leave the session, they may take away data blocks that are already rare in the network, increasing the risk that the content becomes incomplete. The content is especially prone to loss if the seeds have departed.

Let us consider an extreme case where seeds are absent. In the absence of seeds, it can no longer be guaranteed that any segment always remains decodable in the network. If a certain segment has less than $m$ blocks in the network due to peer departures, the content will become incomplete henceforth. However, if the content can be kept complete merely by downloaders for a sufficiently long time, it is possible for the system to evolve into a new equilibrium before the content turns incomplete, as some downloaders will become seeds later. Therefore, it is important to analyze the content persistence in a dynamic network when seeds are absent. In this section, we again consider a challenging
4.4. TRANSIENT BEHAVIOURS

case when peer churn is severe, and evaluate how network coding can enhance content persistence as the segment size $m$ varies.

We assume that the seeds leave the network altogether in the steady state and set time $t = 0$ on their departure. We only consider the replacement model in this section. To evaluate the block availability after seed departures, we set $N_s = 0$, or $\alpha_s = 0$, and thus obtain the differential equations for the system without seeds:

\[
\bar{A} \cdot \frac{ds_i(t)}{dt} = (i - 1)p_{i-1}(t) - ip_i(t), \quad \text{for } i \geq 1
\]
\[
s_0(t) = 1. \quad (4.17)
\]

Since seeds leave in the steady state, we obtain initial conditions at time $t = 0$:

\[
p_i(0) = \bar{p}_i = \left( \frac{i + \frac{B}{\beta} - 1}{\beta} \right) \beta^i (1 - \beta)^{\frac{B}{\beta}}, \quad i = 0, 1, 2, \ldots, \quad (4.18)
\]

where $B = \alpha_s \mu \bar{A}$, $\beta = \frac{\alpha}{\alpha + \alpha_s}$, $\beta$ is a positive and rational number and $\frac{B}{\beta} \in \mathbb{Z}^{++}$. Hence, the problem of determining the block availability at a given time after seed departures has been converted to the problem of solving the system of ODEs (4.17) subject to the initial conditions (4.18).

It is worth noting that when there is no seed ($N_s = 0$), the content will become incomplete eventually. Actually, we have solved for $\bar{p}_i$ by letting $\frac{ds_i}{dt} = 0$ in the case of $\alpha_s = 0$, and found $\bar{p}_i = 0$ for any finite $i$ in steady state. This indicates that nearly all the segments will finally have no block, while only one segment will have an infinite number of blocks in the network.

Although the system can not hold stable without seeds, the transient behaviors of segments after seed departures are still of interest to us. First, it is important to know how
much percentage of the original content becomes undecodable right upon seed departures. If such a content loss is significant, then the content becomes incomplete as soon as seeds leave, with no possibility to recover the lost content even if peer churn becomes less severe later. Apparently, the fraction of undecodable segments upon seed departures is

\[ 1 - s_m(0) = \sum_{i=0}^{m-1} p_i(0) = \sum_{i=0}^{m-1} p_i. \]  

(4.19)

Figure 4.4: The fraction of content lost immediately upon seed departures. The number of downloaders \( N = 1000 \), the number of seeds \( N_s = 10 \), the total number of blocks in the content \( M = 1000 \), upload rate \( \bar{\mu} = \mu/k = 4 \).

We have plotted the content loss right upon seed departures as a function of segment size in Fig. 4.4. When a larger segment size \( m \) is applied, although each value of the sequence \( p_0, p_1, p_2, \ldots \) will decrease, yet more elements of this sequence will sum to the content loss. The combination of the two counteracting effects results in a decreasing trend in the content loss fraction as \( m \) increases. Besides, as the average peer age \( \bar{A} \) increases, less content will be lost upon seed departures, because the effect of peer churn becomes less severe. It is interesting to see that there exists some knee in the curve,
4.4. TRANSIENT BEHAVIOURS

The fraction of decodable content as a function of time after seed departures.

Figure 4.5: The fraction of decodable content at time \( t \) after seed departures. \( N = 1000, N_s = 50, M = 1000, \) upload rate \( \bar{\mu} = \mu/k = 4, \) average peer age \( \bar{A} = 5. \)

beyond which the content loss upon seed departures will not be further reduced by increasing segment size.

We now determine the fraction of decodable content at time \( t \) \((t > 0)\), which is the fraction of segments with no fewer than \( m \) blocks in the network, or \( s_m(t) \). As we vary the segment size \( m \), two factors will intertwine to affect the fraction of decodable content at time \( t \). The first factor is the initial condition at \( t = 0 \) that depends on \( m \). As \( m \) increases, upon seed departures, either less content will be lost or blocks of different segments are distributed in a more balanced way if the content is still complete. The second factor is the objective function \( s_m(t) \) that represents the fraction of decodable content at time \( t \). We have plotted the fraction of decodable content as a function of time \( t \) after seed departures in Fig. 4.5 under different segment sizes. It is clearly shown that, as the segment size \( m \) increases, not only does the system lose less content right upon seed departures, but the content loss rate after seed departures is reduced greatly as well. Moreover, merely a small increment in \( m \) would result in a salient decrease in
4.5. EXPERIMENTAL RESULTS

Finally, we numerically determine content lifetime in a similar way. We consider the case where the content remains complete upon seed departures. The content lifetime is defined as the period from seed departures to the point when at least one segment becomes not decodable. Assume the segment size is $m$, and the total number of segments is $G = M/m$. The content becomes unavailable at the time when at least one segment has less than $m$ blocks in the network, which is the time when the fraction of segments with less than $m$ blocks becomes greater than $\frac{1}{G}$. Thus, the content lifetime equals to the first hitting time of the function $1 - s_m(t)$ to $\frac{1}{G}$ starting from the initial conditions set by (4.18). The results have been plotted in Fig. 4.6, where the parameters are set to the same values as in the previous figure. Similarly, an increasing trend in content lifetime is observed as the segment size $m$ increases.

The analysis in this section has shown that network coding can enhance the persistence of the content in a dynamic network even when seeds are unavailable. Such a content persistence enhances as the segment size $m$ increases. System designers may tradeoff computational complexity by tuning segment size for a better content availability and resilience to peer departures in networks with dynamic peer memberships.

4.5 Experimental Results

We have developed a simulating environment using C++ to experimentally evaluate the behaviour of P2P content distribution systems with peer churn. In most experiments, more than 2,000,000 peers are involved in total, with up to 6,000 peers being simultaneously online at any given time.
Let us first briefly describe the settings of our simulator. The initial input to the simulator is a set of user peers with lifetimes drawn from a certain distribution, and a set of seeds which hold complete copies of the source content. Each downloader will leave the network once its lifetime has expired or it has finished downloading the content. Recall that in the replacement model, any departed peer will be replaced by a new downloader, whereas in the Poisson arrival model, new downloaders join the session in a Poisson process with rate $\lambda$. The simulator has a tracking server, which helps maintain the neighborhood of a peer. Each peer is connected to at least 40 other randomly chosen peers (in accordance to BitTorrent), which form its neighborhood. However, all other tasks, including downstream peer selection and data exchanges, are performed locally at peers.

The simulator runs in time slots (rounds). In each round, a peer first randomly chooses 4 (as in BitTorrent) of its neighbors to serve in this round. This process is called downstream peer selection. However, under such a scheme, there may exist peers not chosen by any other peers in a round, whereas in the meantime other peers may have a large number of upstream peers. To avoid such a load imbalance, we require each peer to have at most 6 upstream peers. With this policy, most peers turn out to have at least one upstream peer in each round. Afterwards, each peer will upload blocks to each of its downstream peers independently using the three-step approach of segment reconciliation, encoding, and transmission, as has been mentioned in Sec. 4.1. To accommodate heterogeneity, we assume there are an equal number of peers with upload bandwidth 2 MB/round, 512 KB/round and 256 KB/round. A peer of 256 KB/round connectivity can upload one block of 256 KB in a round in total.
4.5. EXPERIMENTAL RESULTS

4.5.1 Steady-State Block Distribution and Download Time

Figure 4.7: The convergence of the block variation. $N = 6000$, $N_s = 100$, $F = 768$ MB, $k = 256$ KB. $\overline{T} = 5$ rounds.

Figure 4.8: Steady-state CDF of the number of blocks of each segment found in all the downloaders. Average lifetime of downloaders $\overline{T} = 5$ rounds.

Figure 4.9: Steady-state block variation as a function of segment size $m$ for the replacement model. $N = 6000$, $N_s = 100$, $F = 768$ MB, $k = 256$ KB.

Figure 4.10: Steady-state block variation as a function of segment size $m$ for the Poisson arrival model. $N = 3000$, $N_s = 100$, $F = 768$ MB, $k = 256$ KB.

First, we show that large systems do have a steady state in which block distribution stays roughly constant, as the session has evolved for sufficiently long time. We have plotted the convergence curves of the block variation $\gamma_2^I$ in Fig. 4.7. It is clearly shown that for a network of 6000 peers, after round 150, block variation $\gamma_2^I$ will converge for all
values of segment size. Henceforth, we can reasonably believe that larger networks will exhibit even steadier behaviors after having evolved for sufficiently long as the asymptotic effects become dominant.

Fig. 4.8 shows the steady-state block distribution for the replacement model. The parameters are set to be the same as in Fig. 4.3. We can see the experimental results in Fig. 4.8 matches the analytical results in Fig. 4.3 very well. This has substantiated the correctness of Theorem 5 and the usefulness of the analysis with differential equations.

Figure 4.11: The average time for a long-lived peer to download the entire content in steady state under different segment sizes $m$ for the replacement model. $N = 2000, N_s = 30, F = 256$ MB, $k = 256$ KB. 1% of all the downloaders are long-lived.

Figure 4.12: The average time for a long-lived peer to download the entire content in steady state under different segment sizes $m$ for the Poisson arrival model. $N_s = 30, F = 256$ MB, $k = 256$ KB, $\lambda = 100$. 1% of all the downloaders are long-lived.

We have calculated the average block variation under each segment size value $m$ in steady state and plotted the results in Fig. 4.9 and Fig. 4.10 for the replacement model and Poisson arrival model respectively. It is clearly shown that the steady-state block variation $\gamma_I^2$ is inversely proportional to the segment size $m$, regardless of which model is used. This means the block number distribution of different segments becomes more balanced as segment size increases. Moreover, when peer churn is more severe, or when the average peer age is lower, the network suffers from greater block variation. Even for
4.5. EXPERIMENTAL RESULTS

Figure 4.13: Steady-state block variation under different peer joining rates $\lambda$. $N_s = 100$, $F = 768$ MB, $L = 10$ rounds.

Figure 4.14: The average time for a long-lived peer to download the entire content in the steady state.

such a heterogeneous environment, simulation results are quite close to the conclusion of Theorem 7, though we do not fully understand why simulation results differ from the analysis by a constant coefficient. This is probably because we have adopted a time-synchronized model in the simulation, whereas in the analysis we have not.

We now evaluate the time required for a long-lived user to download the whole content as the segment size $m$ varies using simulation. It turns out that the download time is closely related to the block variation in the network. The results are plotted in Fig. 4.11 and Fig. 4.12 for the replacement model and Poisson arrival model respectively. 1% of all the user peers are long-lived ones which will not leave until they finish downloading. The other peers are short-lived ones which have lifetimes exponentially distributed. It is clearly shown that as $m$ increases, the average download time required to obtain the entire content is reduced. The relationship of the download time and segment size has a similar form (inverse proportion) as that of the block variation and segment size. This is because as the block variation is subdued, peers will not be hindered in obtaining those rare blocks, incurring less delay in download processes. Besides, there is a sweet spot of
4.5. EXPERIMENTAL RESULTS

Figure 4.15: The number of blocks that each segment has at different times $t$ (rounds).

a) non-coding scheme (segment size $m = 1$). b) network coding with segment size $m = 4$.

Servers leave in the steady state (round 400), and we set $t = 0$ again upon seeds departure. $N = 6000$, $N_s = 100$, $F = 768$ MB, $k = 256$ KB, $L = 20$ (rounds).

segment size, beyond which download time can hardly be further reduced. Thus, the use of a small segment size, such as 10-20, suffices to optimize the download efficiency, with only a moderate computational complexity incurred.

We have also studied the effects of the network size on the system performance. We vary the downloader joining rate $\lambda$ under the Poisson arrival model. Note that the average number of online downloaders equals to $\lambda L$ in the steady state. From Fig. 4.13, we see that simulation results get closer to our asymptotic analysis as $\lambda$ increases, which indicates a larger network size in steady state. Besides, when $\lambda$ is small, there is much randomness in the content distribution session, although a roughly inverse proportional relationship is still observed between the block variation and segment size. Finally, we set $F = 256$ MB, $N_s = 30$, $k = 256$ KB and 1% of all the downloaders to be long-lived. Average lifetime of short-lived peers is set to be 10 rounds. We plot the average time that a long-lived peer downloads the entire content in the steady state as $\lambda$ varies in Fig. 4.14. We can see that as the joining rate $\lambda$ increases, download time is reduced. This is mainly because the block variation is subdued with a bigger $\lambda$, as has been shown in Fig. 4.13.
4.5. EXPERIMENTAL RESULTS

Figure 4.16: The fraction of decodable content after server (seed) departures as time passes. Seeds leave in the steady state (round 400), and we set $t = 0$ again upon seed departures. $N = 6000$, $N_s = 100$, $F = 768$ MB, $k = 256$ KB, $L = 20$ (rounds).

4.5.2 Content Persistence in the Absence of Seeds

We consider the extreme case when seeds leave in the steady state altogether, and set $t$ back to zero upon seed departures. Let us now experimentally evaluate the content persistence in the absence of seeds. First, we show in Fig. 4.15 the block distributions at different times after seed departures. In the non-coding case (segment size $m = 1$), as many as 50% of the segments will become undecodable at round 1000, while for network coding with merely a small segment size $m = 4$, the block distribution stays almost constant within 2000 rounds, with only a small fraction of content lost at round 2000. As has been pointed out in our analysis, this is primarily because the segments are distributed in a more balanced way as segment size increases, and thus each segment suffers from a lower risk of becoming a rare segment after seed departures. We have also plotted the fraction of decodable content as a function of time after seed departures in Fig. 4.16, which agrees with the analysis in Sec. 4.4 in trend.
4.6 Summary

In this chapter, we study the fundamental resilience-complexity tradeoff of applying network coding in P2P content distribution sessions with frequent peer joins and departures. To incorporate different degrees of coding complexity, we analyze a flexible framework of segment-based network coding, one extreme of which is a pure network coding scheme, with only one segment in the source content for coding, while the other extreme of which is a non-coding scheme, with as many segments as blocks. We have quantified the resilience of network coding in terms of gain in block availability, download time, content loss rate and content lifetime under two peer dynamics models, as the number of blocks within a segment varies. Such an analysis is done using a differential equations approach that asymptotically approximates the underlying Markov process in the proposed model. Extensive simulations have been carried out to substantiate our theoretical results and shed further insights into the behaviour of network coding in dynamic P2P networks.

Our findings have shown that small segment sizes — around 20 in our simulations even with high peer volatility — suffice to realize the major benefit of network coding in terms of reducing download times and preserving block diversity when seeds are available. When seeds are absent, no sweet spot of the segment size for coding has been observed. Without seeds, content may persist for a longer time in the network as segment size $m$ increases.
Chapter 5

Indirect Data Collection with Network Coding

Collecting vital statistics data from peers has become critical to monitor and diagnose large-scale peer-to-peer (P2P) multimedia applications, especially when they are in live operation as a commercial platform. Traditionally, these commercial P2P live streaming systems use logging servers to collect vital statistics data from the participating peers periodically. Such statistics data consist of measurements of important performance metrics in the P2P application at each peer, and are used by network administrators and analysts to improve the protocol design or to troubleshoot network outage.

It has recently become apparent that the use of centralized logging servers is not scalable to the large number of participating peers. As has been observed in previous P2P measurement studies such as measurements from UUSee Inc. [37, 36] — a leading provider in mainland China for P2P live streaming solutions — periodic statistics data collection actually consumes a very substantial volume of traffic, especially when the
number of peers in the session increases dramatically in a short period of time. Such periodic reporting essentially morphs into a *de facto* Distributed Denial of Service (DDoS) attack to the logging servers, as the server bandwidth is not sufficient to handle an excessive number of simultaneous uploading flows, with either TCP or UDP as the transport layer.

Naively, these logging servers may be able to mitigate this problem by periodically changing the peers that they proactively pull data from. However, with limited bandwidth, the servers can only download from a small subset of all the peers at the same time, leaving most of the peers waiting for service. Aggravated by the phenomenon of peer dynamics, a large number of peers may have already left the system before any of the servers actually become available to “pull” from them, losing such statistics permanently. Ironically, since peers tend to leave soon after the quality degrades, such statistics from departed peers may be the most useful to diagnose system outages or protocol deficiencies! Even if all peers may eventually be probed by logging servers, the bandwidth available on the pair-wide Internet path between a peer and a central logging server may be too low to successfully report all vital statistics.

In a nutshell, with current solutions using centralized logging servers, the amount of data that can be instantaneously collected from a large-scale P2P system is strictly limited by the server bandwidth. While such bandwidth may be more than sufficient to handle the *average* rate of collecting such vital statistics, it may not be able to accommodate flash crowds of peer arrivals and large-scale peer departures in a short period of time. In order to achieve better scalability, one possible remedy is to increase the number or bandwidth of such logging servers, trying to accommodate the *peak* load of collecting statistics, rather than the *average*. This is extremely costly and fails to take advantage
of statistical multiplexing of resources. Since the task of statistics collection is usually not time-sensitive, we are able to trade-off *timeliness* for bandwidth, by allowing peers to buffer such data in a decentralized fashion, much like a distributed storage for vital statistics.

In this paper, we design, analyze, and simulate an indirect data collection mechanism that is specifically designed for collecting vital statistics in large-scale peer-to-peer systems, such as P2P streaming. Our design has two simple objectives: *scalability*, where the design must be scalable to handle a large number of peers, with some trade-offs on delay; and *resilience*, where losses of vital statistics from peers that have recently departed should be kept to the minimum.

In particular, our new mechanism requires that peers first exchange their statistics data blocks with their neighbors using random network coding and probabilistic gossip protocols. By utilizing peer resources to form a “buffering” pool, the transmission of statistics from peers to logging servers is “cushioned,” such that server bandwidth is provisioned to handle only *average* load, rather than the *peak*. To some extent, our new mechanism is akin to delay tolerant networks: by spreading data blocks across the network, the data that the servers are unable to collect immediately are stored for future delivery in a delayed fashion. Such “cushioning” also plays an important role to prevent losing vital statistics of departing peers. We present theoretical bounds to show that network coding can almost *exponentially* improve data persistence in the presence of peer dynamics, as we increase the number of blocks that are used to generate a coded block. Results from extensive simulations are also shown to corroborate our analysis, as well as to shed more insights with respect to the performance of our proposed mechanism.

The remainder of this chapter is organized as follows. In Sec. 5.1, we propose the basic
idea of indirect data collection mechanism, and analyze its benefits in a simple stochastic model. In Sec. 5.2, we introduce random network coding and bound its resilience as coding complexity varies. In Sec. 5.3, we consider statistics generated at each peer as a data stream and propose peer buffer management and server collection algorithms to make the indirect collection mechanism more practical. In Sec. 5.4, we formulate a system of ODEs to characterize the proposed algorithms. Based on these ODEs, we evaluate the performance of our new mechanism in Sec. 5.5. Simulation results are presented along with the analytical results in each section. We summarize the chapter in Sec. 5.6.

5.1 Data Spreading with Gossiping

We consider the problem of collecting vital statistics data from a large number of peers. In this section, we introduce our basic idea for indirect data collection, where data are spread across the system, from which servers pull data. Through a simple yet accurate stochastic model, we demonstrate the advantages of the indirect mechanism over using conventional logging servers in a centralized fashion.

Consider a P2P network of $N$ nodes during time periods of peak load (such as during the prime evening time for P2P streaming systems), each with $m$ data blocks that need to be delivered to the server. Each peer has an average upload bandwidth of $\mu$ set aside for reporting statistics. We assume that the aggregate bandwidth of logging servers $C < N\mu$, i.e., such bandwidth at logging servers may not be able to accommodate peak load of reporting vital statistics. Fig. 5.1(a) illustrates the traditional mechanism of using logging servers to “pull” directly from peers.

In the case of collecting vital statistics for postmortem analysis, we are most likely able to trade-off the timeliness of retrieving such data, and tolerate some degree of delay.
5.1. DATA SPREADING WITH GOSSIPING

Figure 5.1: (a) Using the logging servers to directly “pull” from all the peers may not be scalable, as server bandwidth needs to accommodate the peak load in a flash crowd scenario (such as during evening prime times in P2P streaming). (b) If we are able to tolerate some delays, peers can be asked to exchange data blocks in a distributed fashion to achieve better data persistence.

In this case, asking peers to exchange their data blocks in a distributed fashion will achieve better data persistence in the event of extreme peer dynamics. The server may pull data from the network afterwards in a delayed fashion, shown in Fig. 5.1(b). Our basic data spreading algorithm can be described as the following gossip-like protocol:

1) At rate $\mu$, each peer, say peer $A$, chooses a block $b$ uniformly at random (u.a.r.) from among all the blocks in its buffer.

2) $A$ then transmits block $b$ to peer $B$ chosen u.a.r. from among its neighbors which do not have $b$. (The neighbors of peer $A$ are the peers that maintain data connections with $A$.)

A gossip protocol for data replication enjoys a better resilience and scalability than a structured protocol such as building trees, in that data blocks are spread through the network much like epidemics are spread among people. In other words, the randomness introduced by gossiping is resilient to data losses due to any irregular peer departure patterns.
5.1. DATA SPREADING WITH GOSSIPING

We now give a quantitative comparison of the proposed data spreading and the centralized pull mechanisms, in terms of their loss resilience with the presence of peer departures and the overhead of resource consumption. First, it is obvious that for the centralized pull-based mechanism, upon each peer departure, $m$ original data blocks are lost. Moreover, the overhead is zero, as all bandwidth and storage resources are used for local uploads only.

To derive the overhead for the gossip protocol, let $E(t)$ denote the total number of block copies in the network and $e(t) := \frac{1}{N} E(t)$ denote the average number of blocks in each peer at time $t$. We have the following straightforward proposition:

**Proposition 8** At time $t$, the average storage overhead of each peer is $\mu t$. Moreover, the average number of blocks in each peer’s buffer is $e(t) = \mu t + m$.

Proof: As each peer uploads at a rate of $\mu$, the average number of blocks each peer has obtained at time $t$ is simply the total number of uploaded blocks $N\mu t$ divided by the peer number $N$. Since each peer initially has $m$ blocks of its own, we have $e(t) = \mu t + m$. \qed

To analyze how many original data blocks are lost permanently upon each peer departure, we need to know the block availability information at time $t$. For this purpose, let $X_i(t)$ be the number of distinct blocks of degree $i$ at time $t$. A block is of degree $i$ if and only if there are $i$ copies of this block in the network. We also define the rescaled degree sequence as $w_i(t) := \frac{1}{N} X_i(t)$, which we will use later. We can formulate the following set of differential equations that asymptotically approaches the underlying gossip process as $N \to \infty$:

$$w_i'(t) = \frac{(i - 1)w_{i-1}(t) - iw_i(t)}{e(t)} \cdot \mu, \quad i = 1, 2, \ldots ,$$  \hspace{1cm} (5.1)

with initial conditions $w_1(0) = m$, $w_i(0) = 0$, $i = 2, 3, \ldots$. This differential equation system is a special case of (5.12) in Sec. 5.4.
Let us explain the reasoning behind (5.1). First, it is straightforward that \( w_1(0) = X_1(0)/N = mN/N = m \). Consider the expected change of \( X_i(t) \) in a small time interval \( \Delta t \), during which at most one block transfer occurs in the entire network. As the upload time of a block is exponential with mean \( 1/\mu \), the probability that one block is transferred in \( \Delta t \) is \( N\mu\Delta t \), and the source peer of this transfer is randomly distributed among all the peers. In order to see which block is actually replicated in this transfer, we approximate the action of choosing a random block in a random peer to transfer by choosing a block \( b \) directly from all the blocks with probability \( \text{deg}(b)/E(t) \), where \( \text{deg}(b) \) is the degree of block \( b \).

This linear approximation essentially means that the more copies a block has in the network, the greater the chance it gets replicated. The effectiveness of this approximation will be demonstrated through simulation. Since during a replication the probability that a degree \( i \) block is chosen is \( iX_i(t)/E(t) \), we have the expected number of blocks of degree \( i \) changing to degree \( i + 1 \) in \( \Delta t \) is \( (iX_i(t)/E(t)) \cdot N\mu\Delta t \). When a block changes its degree from \( i \) to \( i + 1 \), \( X_i \) decreases by one and \( X_{i+1} \) increases by one. Hence, given \( X(t) \), we have

\[
E\{X_i(t + \Delta t) - X_i(t) | X(t)\} = \frac{(i-1)X_{i-1}(t) - iX_i(t)}{E(t)} \cdot N\mu\Delta t. \tag{5.2}
\]

Dividing by \( N \cdot \Delta t \) on both sides as \( N \to \infty \), we obtain (5.1).

We can derive the expected loss of original data blocks upon each peer departure using (5.1).

**Proposition 9** For the gossip protocol without coding, at time \( t \) the average number
of original data blocks lost permanently from the network upon each peer departure is
\[ \mathcal{L}_t(1) = \frac{m^2}{\mu t + m}, \]
whereas for the centralized pull-based mechanism, this quantity is \( \mathcal{L}_t(0) = m \).

**Proof:** The proof is a special case of the proof for Theorem 10 when \( s = 1 \). \( \square \)

---

![Graph](image.png)

**Figure 5.2:** The expected number of original blocks lost permanently upon each peer departure at time \( t \) under the gossip protocol.

This simple analysis proves to be very accurate. We plot the expected permanent loss of original data upon each peer departure in Fig. 5.2 for both analytical and simulation results. The simulator accepts the input of 1000 peers and organizes them into a random topology where each peer connects to 30 other randomly chosen peers on average. The simulation is round-based. In each round, every peer performs the aforementioned two-step gossip protocol. The average peer bandwidth is \( \mu = 1 \). At the end of each round, we randomly choose 20 peers and see how many original data blocks are lost from the entire network if these chosen peers leave. We have run such a simulation for 40 times to take the average.

Both the analytical and simulation results have demonstrated that upon each peer
departure at time $t$, for every original block lost under the centralized pull-based protocol, the system only loses $\frac{m}{m+\mu t}$ block under the gossip protocol. Such a loss becomes very small as $t$ grows, even if only a small value of peer bandwidth $\mu$ is set for the collection session. As data blocks persist in the network under the gossip protocol, servers can retrieve blocks that it is not able to download at first by probing a subset of peers later.

5.2 Resilience of Random Network Coding

In order to further increase data persistence, we leverage the wisdom of random network coding [15], [5], [13] to spread data blocks across the network in their coded form. Intuitively, as coded blocks are all equally useful, as long as there is an sufficient number of linearly-independent coded blocks in the network, the data can be reconstructed. This is remarkably different from the simple gossip protocol, where certain data blocks could become rare and thus more vulnerable to losses than other blocks upon peer departure.

However, the computational complexity of network coding mandates that, in reality, network coding needs to be performed within segments, each containing a subset of all the blocks produced at each peer. In this section, we introduce segment based network coding into our gossip-based data spreading. We have found an almost exponential increase in the loss resilience as the number of blocks in a segment increases.

As mentioned in Chapter 3, the basic idea of segment based network coding or group network coding [5] is to group the original data blocks produced at each peer into segments, each with a prescribed number of blocks. Let $s$ denote the number of original blocks in a segment, a quantity we refer to as the segment size. Similar to a content distribution system as described in Chapter 3, a random linear code (RLC) is applied to each segment of $s$ blocks in the following way:
5.2. **RESILIENCE OF RANDOM NETWORK CODING**

Assume a certain segment *i* generated at peer *A* has original blocks $B^{(i)} = [B^i_1, B^i_2, \ldots, B^i_s]$, then a coded block *b* from segment *i* is a linear combination of $[B^i_1, B^i_2, \ldots, B^i_s]$ in the Galois field $\text{GF}(2^8)$. Coding operation is not limited to the source: if a peer (including the source) has buffered $l$ ($l \leq s$) coded blocks of segment *i* $[b^i_1, b^i_2, \ldots, b^i_l]$, when transferring to another peer *p*, it independently and randomly chooses a set of coding coefficients $[c^p_1, c^p_2, \ldots, c^p_l]$ in the Galois field $\text{GF}(2^8)$, and encodes all the blocks it has from segment *i*, and then produces one coded block $x$: $x = \sum_{j=1}^{l} c^p_j \cdot b^i_j$. The coding coefficients used to encode original blocks to $x$ are embedded in the header of the coded block. Servers collect coded blocks instead of original blocks when network coding is applied. As soon as the server has collected a total of $s$ coded blocks from segment *i* $x = [x^i_1, x^i_2, \ldots, x^i_s]$ that are linearly independent, it will be able to decode segment *i*. The decoding complexity turns out to be approximately $O(s)$ operations per input block [23]. Naturally, we can vary the coding complexity by changing the segment size. Note that $s = 1$ indicates the non-coding case.

Coded blocks are spread throughout the network by a similar gossip protocol with group network coding:

1) At rate $\mu$, each peer, say peer *A*, chooses a segment *r* u.a.r. from among all the segments which have at least one (encoded) block in its buffer to generate a coded block $q$.

2) *A* then transmits $q$ to peer *B* chosen u.a.r. from among its neighbors which have not received $s$ linearly-independent coded blocks of segment *r*.

Such a process is illustrated in Fig. 5.3. At first, each peer produces 2 segments, each of which consists of 3 original blocks. At a certain time, Peer *B* chooses to transfer an
encoded block of Segment 4 to Peer C. Afterwards, Peer C happens to choose Segment 4 to encode. Since it only has one coded block of Segment 4 that it just received, it transfers that block directly to another randomly chosen peer, that is Peer A.

![Figure 5.3: A bipartite graph showing the relationship among segments, blocks and peers with $m=6$, $s=3$. Every edge between a segment and a peer represents a (coded) block of this segment in this peer. Solid lines denote original blocks, while dashed ones denote transferred blocks.](image)

In what follows, we give bounds on the expected loss of original data upon each peer departure when network coding is applied with a segment size $s$. When $s \geq 2$, it is not hard to see Proposition 8 still holds. Thus, the storage overhead for coded data spreading is still $\mu t$. To characterize block availability, we now use $X_i(t)$ to denote the number of segments of degree $i$ in the network at time $t$. A segment is of degree $i$ if and only if there are $i$ blocks of this segment in the network. Similarly, let $w_i(t) := \frac{1}{N}X_i(t)$ denote the rescaled degree sequence for segments.

We follow the same approach as in Sec. 5.1 and apply a similar linear approximation in the analysis for network coding, i.e., we approximate the action of choosing a random segment in a random peer to encode by choosing a segment $r$ directly from all the segments with probability $\text{deg}(r)/E(t)$. This means the more blocks a segment has in the network, the greater the chance it is chosen and encoded. Thus, similarly, during a block transmission, the probability of choosing a degree $i$ segment is $iX_i(t)/E(t)$.

To simplify the analysis, we assume no linear dependency will occur when a peer is
uploading a coded block to another. This is reasonable because according to Lemma 2.1 in [7], a random linear combination of all the blocks from the same segment at a peer \( p \) is useful to another randomly chosen peer with probability at least \( 1 - \frac{1}{q} \) if network coding is done in \( \mathbb{F}_q \). By following the same argument henceforth as in Sec.5.1 except that we replace the notion of original blocks by segments here, we come up with the same differential equations as (5.1):

\[
 w'_i(t) = \frac{(i-1)w_{i-1}(t) - iw_i(t)}{e(t)} \cdot \mu, \quad i = 1, 2, \ldots, 
\]

though with different initial conditions: \( w_s(0) = m/s, w_i(0) = 0, i \neq s \).

When the segment size is \( s \), all the segments at time \( t = 0 \) have \( s \) blocks, which are actually \( s \) original blocks, and thus all have a degree of \( s \). Since there are \( N \cdot m/s = X_s(0) \) segments in total, we have \( w_s(0) = X_s(0)/N = m/s \) and \( w_i(0) = 0 \) for all \( i \neq s \). We are now ready for bounding the resilience of network coding:

**Theorem 10** For the network coding based gossip protocol with a segment size of \( s \), at time \( t \) the expected number of original data blocks lost permanently from the network upon each peer departure \( L_t(s) \) satisfies

\[
 m \cdot \left( \frac{m}{\mu t + m} \right)^s \leq L_t(s) \leq \min \{m, m \cdot s \left( \frac{m}{\mu t + m} \right)^s \}, 
\]

where \( s = 1 \) indicates a non-coding gossip protocol, and \( s = m \) indicates coding across all the blocks generated at the same peer.

**Proof:** First, note that the degree of any segment cannot be decreasing due to block encoding and transferring, given the initial conditions of (5.3), we have \( w_i(t) = 0 \) for all
Thus, we have \( w'_s(t) = -sw_s(t)/e(t) \), \( w_s(0) = m/s \), the solution of which is

\[
w_s(t) = \frac{m}{s} \cdot \left( \frac{m}{\mu t + m} \right)^s.
\]

A segment is not decodable if it has less than \( s \) (coded) blocks in the network. Hence, the expected number of lost segments after a peer leaves is the expected number of segments of degree \( s \) changing to \( s - 1 \). At time \( t \), the expected loss of (coded) blocks upon each peer departure is \( e(t) = \mu t + m \). By linear approximation, \( e(t) \cdot \frac{sX_s(t)}{E(t)} = sw_s(t) \) of these (coded) blocks belong to segments of degree \( s \). These \( sw_s(t) \) blocks can at most belong to \( sw_s(t) \) different degree-\( s \) segments. This gives an upper bound of the expected number of lost segments as \( sw_s(t) \). By step 2) of the protocol a peer can have at most \( s \) (coded) blocks from the same segment. As these \( sw_s(t) \) blocks are in the same peer, they should at least belong to \( w_s(t) \) different degree-\( s \) segments, which gives a lower bound of the expected segment loss as \( w_s(t) \). As \( s \) original data blocks are lost altogether upon each segment loss, multiplying by \( s \) yields exponential bounds for the expected loss of original blocks as \( sw_s(t) \leq \mathcal{L}_t(s) \leq s^2w_s(t) \).

To derive the trivial upper bound \( m \), we note that \( \mathcal{L}_0(s) = \mathcal{L}_t(0) = m \), \( \forall s, t \). For \( s \geq 1 \), less original data are lost upon each peer departure when \( t > 0 \) as each peer’s data are buffered in other peers.

Theorem 10 suggests an asymptotically almost exponential decreasing trend of \( \mathcal{L}_t(s) \) as the segment size \( s \) grows, and thus establishes a strong case of using random network coding to protect original data from being lost permanently when network is dynamic. To understand the effect of network coding in more details, we have plotted the simulation results regarding average permanent data loss upon each peer departure in Fig. 5.4 for different values of the segment size. In this experiment, we set \( m = 20 \), \( \mu = 1 \), other
parameters are the same as for Fig. 5.2.

Figure 5.4: The expected number of original blocks lost permanently upon each peer departure for different segment sizes $s$.

According to Fig. 5.4, at the beginning, the non-coding gossiping mechanism provides a better loss resilience. This is because with a segment size $s$, a peer has to spread out at least $s$ coded blocks of one of its segment to have one additional "copy" of this segment buffered in the network, whereas without coding, each transmission generates one additional "copy" of the transmitted block. In other words, network coding needs longer initial time to build up the resilience for the system and thus is more vulnerable to data losses due to peer departures at first. However, the loss resilience of network coding is bounded by the trivial bound of $m$ at this stage, as we can see for $t = 0 \sim 10$. As $t$ grows, the curves for network coding decay much faster. Eventually network-coded mechanisms become more resilient than a non-coding mechanism after $t = 25$. We can also see that the larger the segments we use, which indicates a higher coding complexity, the slower is it for the system to build up its resilience, however, the greater is the resilience it can finally provide.
5.3 Buffer and Server Algorithms for Stream Data Collection

In this section, we consider statistics produced as a data stream at each peer, when the upload demands are very high. We propose simple yet effective buffer management and server collecting algorithms to ensure smooth data collection even in the presence of fluctuating traffic, with a limited storage overhead. To aid the analysis, the entire set of proposed algorithms are interpreted on a bipartite graph process at the end of this section. We analyze the performance of the proposed algorithms with respect to throughput, delivery delay, overhead and loss resilience using differential equations and simulations in Sec. 5.4 and Sec. 5.5.

In practice, upload demands at each peer are generated at a fluctuating rate instead of a constant rate. With a fluctuating volume of traffic, traditional logging servers have to be provisioned with very high bandwidth that accommodates the peak load of upload demands rather than the average. In contrast, we will show that using the proposed indirect collection mechanism, it is sufficient for servers to have bandwidth that only handles the average traffic, as data that are not collected during the peak traffic are buffered in a distributed fashion and will be collected when the peak phase ends. Furthermore, a traditional logging server approach has the difficulty of optimally allocating resources to serve different peers when the traffic is fluctuating. With the indirect mechanism, we can show that resources can be automatically allocated in an optimal way due to the randomness in both data spreading and server collection algorithms.

Considering the fluctuating nature of the upload demand at each peer, we assume original statistics data blocks are generated at each peer according to a Poisson process
with rate $\lambda$. When network coding is applied with a segment size of $s$, segments of $s$ blocks are injected at each peer in a Poisson process with rate $\lambda/s$, resulting in a same block rate of $\lambda$.

To ensure smooth data collection and optimal resource allocation in light of fluctuating traffic, we use a coupon-collector-like server algorithm to collect data. Assume there are $N_s$ servers collaborating to collect data, each with a capacity of $c_s$. For the simplicity of notation in the analysis, we define the normalized server capacity as $c := c_s N_s / N$. At rate $c_s = c \cdot N / N_s$, each server chooses a peer $p$ u.a.r. from among all the peers with non-null buffers and chooses a random segment in peer $p$, which then transmits one coded block of this segment to the server. Note that this algorithm incurs little communication overhead as no buffer comparison is made between a server and peers or among the servers. This means servers may collect redundant blocks of a segment which is already decodable. However, we will show that even with the simplicity in this protocol, the collection efficiency, or in other words, the session throughput can be made close to optimal probabilistically by adjusting the segment size for coding and various other parameters.

To prevent redundant data blocks from flooding the network in the gossip protocol, we need an efficient buffer management algorithm that limits the storage overhead at each peer while still maintaining the resilience to block losses in a dynamic network. For this purpose, we require each (coded) block to have a time to live (TTL) at each peer exponentially distributed with mean $1/\gamma$. A (coded) block is deleted once its TTL has expired to make storage spaces for newly generated or obtained blocks. Such an automatic probabilistic purge mechanism incurs no communication overhead and is therefore much more scalable than an acknowledgment based mechanism, where servers flood $ACK$
packets to the network to confirm the reception of a certain block, which may cause congestion. Moreover, each peer’s buffer is set to have a cap of size $B$. If a peer’s buffer is full, it will not accept blocks from its neighbors. Note that the proposed algorithms are very robust to fluctuating workloads, network outages and peer dynamics due to the randomness inherent in these algorithms.

In what follows, we map the entire set of algorithms for indirect data collection onto a random bipartite graph process, which can be treated more conveniently in the analysis. Differential equations are formulated to asymptotically characterize the behavior of the graph process as the number of peers $N \to \infty$ in the next section.

![A bipartite graph representation.](attachment:image.png)

As illustrated in Fig. 5.5, segments and peers correspond to vertices on different sides of a bipartite graph $G$ respectively. For every (coded) block of segment $r$ in peer $A$, there is an edge in $G$ between peer $A$ and segment $r$. Multiple edges can be allowed for a segment-peer pair, as each peer can buffer up to $s$ linearly independent (coded) blocks of a certain segment. For example, in Fig. 5.5, Peer 1 has three blocks from Segment 1 and one block from Segment 3. Servers are not shown in $G$, and we do not differentiate between original and coded blocks.

Denote by $X_i(t)$ the number of segments of degree $i$ at time $t$ and by $Y_i(t)$ the
number of peers of degree $i$. A segment is of degree $i$ if and only if there are $i$ blocks of this segment in the network. A peer is of degree $i$ if and only if it contains $i$ blocks in its buffer. The total number of blocks in the network equals to the total number of edges in $G$, denoted by $E(t)$. Let $z_i := \frac{1}{N}Y_i$ and $w := \frac{1}{N}X_i$ denote the rescaled degree sequences and $e(t) := \frac{1}{N}E(t)$ be the average number of blocks in each peer. Actions of the peers and servers can be interpreted by the following graph operations:

- Segment Injection (Adding Edges and Vertices). At rate $\lambda/s$, $s$ new edges are added to each peer in $G$ whose degree is no more than $B - s$, together with a new segment incident to these $s$ edges.

- Block Encoding and Transfer (Adding Edges). At rate $\mu$, each peer $s$ in $G$ whose degree is non-null picks u.a.r. a segment $p$ from all the segments adjacent to it and picks u.a.r. from all the peers a peer $d$ that needs blocks of segment $p$, and whose degree is less than $B$. Add a new edge $pd$ to $G$.

- Block Deletion (Deleting Edges). At rate $\gamma$, each edge in $G$ is deleted. Particularly, if a segment in $G$ has degree zero after an edge is deleted, then the segment is deleted from $G$.

- Server Collection. Assume each segment is in one of the $s + 1$ states; a segment is in state $i$ if and only if the servers have collected $i$ linearly independent blocks of this segment ($i = 0, 1, \ldots, s$). At rate $c_s = c \cdot N/N_s$, each server chooses a peer $d$ u.a.r. from all the peers with non-zero degrees. It then chooses a segment $p$ u.a.r. from all the segments adjacent to peer $d$. Increase the state value of segment $p$ by one if its state value is less than $s$. Otherwise, do not increase its state value, since a segment in state $s$ can already be decoded and any further collection of blocks
from this segment will be redundant.

We can characterize the system evolution by characterizing the graph process $G_t$. In our problem, the structure of $G_t$ is well represented by the degree sequences $X_i(t)$ and $Y_i(t)$.

## 5.4 A Differential Equation Characterization

In this section, we formulate a system of differential equations on the graph process $G_t$ mentioned above to characterize the asymptotic system behaviors as $N \to \infty$. We take a similar approach of considering the expected changes of $X_i$ and $Y_i$ contributed by different operations as $G_t$ evolves to establish the system behavior in its limiting case. The proof of the correctness of this approach and the asymptotic approximation of the derived differential equations to the underlying process $G_t$ can be found in [34].

Let us first consider block encoding and transfer. Consider the change of the number of degree $i$ peers $Y_i$ in $\Delta t$, during which only one action of edge addition or deletion occurs. When a segment $p$ is chosen for encoding, a peer $d$ is chosen u.a.r. from all the peers with degrees less than $B$ that still need blocks of segment $p$. Edge $pd$ is then added. The probability of choosing a degree $i$ peer is $(Y_i(t) - F_{pi}(t))/(N - Y_B(t) - F_p(t))$, where $F_{pi}(t)$ denotes the number of edges already present between segment $p$ and peers of degree $i$, and $F_p(t)$ denotes the number of edges present between $p$ and all the peers, which equals to the degree of $p$. As the maximum degree of any peer is bounded by $B$, it turns out that $F_{pi}(t)$ and $F_p(t)$ can only take finite values as well, that is $F_{pi}(t) \leq F_p(t) = o(N)$ for
5.4. A DIFFERENTIAL EQUATION CHARACTERIZATION

all \( p \). Thus, the expected number of peers of degree \( i \) changing to \( i + 1 \) in \( \Delta t \) is

\[
1 \cdot \left( \frac{Y_i(t) - o(N)}{N - Y_B(t) - o(N)} \right) \cdot (N - Y_0(t)) \mu \Delta t
\]

\[
= \left( \frac{Y_i(t)}{N - Y_B(t)} + o(1) \right) \cdot (N - Y_0(t)) \mu \Delta t,
\]

where \((N - Y_0(t)) \mu \Delta t\) is the probability that one edge is added in \( \Delta t \). When a peer changes its degree from \( i \) to \( i + 1 \), \( Y_i \) decreases by one and \( Y_{i+1} \) increases by one. Hence,

\[
E\{Y_i(t + \Delta t) - Y_i(t)|G_t\}
\]

\[
= \left( \frac{(1 - \delta_{i0})Y_{i-1}(t) - (1 - \delta_{iB})Y_i(t)}{N - Y_B(t)} + o(1) \right)
\]

\[
\cdot (N - Y_0(t)) \mu \Delta t,
\]

where \( \delta_{ij} = 1 \) if \( i = j \) and \( \delta_{ij} = 0 \) if \( i \neq j \) is the Kronecker delta function. Recall that \( z_i(t) := \frac{1}{N} Y_i(t) \). Dividing by \( N \cdot \Delta t \) on both sides, we get

\[
z_i'(t) = \frac{(1 - \delta_{i0})z_{i-1}(t) - (1 - \delta_{iB})z_i(t)}{1 - z_B(t)} \cdot (1 - z_0(t)) \mu,
\]

(5.5)

The characterization of \( X_i(t) \) under block encoding has been done in Sec. 5.2. The only difference here is that the probability of adding one edge is \((N - Y_0(t)) \mu \Delta t\) instead of \( N \mu \Delta t \). Hence, (5.3) in Sec. 5.2 is replaced by

\[
w'_i(t) = \frac{(i - 1)w_{i-1}(t) - iw_i(t)}{e(t)} \cdot (1 - z_0(t)) \mu.
\]

(5.6)
5.4. A DIFFERENTIAL EQUATION CHARACTERIZATION

Let us now consider block deletion. In $\Delta t$, an edge is deleted with probability $E(t)\gamma \Delta t$ from $G_t$. Since the deletion process is Poisson, this edge is uniformly distributed among all edges. The probability that the deleted edge is adjacent to a degree $i$ peer is $iY_i(t)/E(t)$. Thus, the expected number of degree $i$ peers changing to degree $i-1$ is $1 \cdot (iY_i(t)/E(t)) \cdot E(t)\gamma \Delta t = iY_i(t)\gamma \Delta t$. Hence, the expected change of $Y_i$ due to block deletion is

$$E\{Y_i(t + \Delta t) - Y_i(t)|G_t\} = ((1 - \delta_i B)(i + 1)Y_{i+1}(t) - iY_i(t)) \cdot \gamma \Delta t,$$

and thus,

$$z_i'(t) = ((1 - \delta_i B)(i + 1)z_{i+1}(t) - iz_i(t)) \cdot \gamma,$$

$$i = 0, 1, \ldots, B. \quad (5.7)$$

The probability that the deleted edge is adjacent to a degree $i$ segment is $iX_i(t)/E(t)$. Similarly, we can characterize the rescaled changing trend of $w_i$ as follows:

$$w_i'(t) = ((i + 1)w_{i+1}(t) - iw_i(t)) \cdot \gamma, \quad i = 1, 2, \ldots. \quad (5.8)$$

Finally, we consider the contribution by segment injection. A segment is injected into the network in $\Delta t$ with probability $(N - Y_{(f)}(t)) \cdot \frac{\lambda}{s} \cdot \Delta t$, where $Y_{(f)}(t) = \sum_{k=B-s+1}^{B} Y_k(t)$. This segment appears in a degree-$i$ peer with probability $Y_i(t)/(N - Y_{(f)}(t))$. Thus, the expected number of degree-$i$ peers changing to degree $i+s$ is simply $[(N - Y_{(f)}(t))\lambda \Delta t/s]$. 
5.4. A DIFFERENTIAL EQUATION CHARACTERIZATION

\[ \frac{Y_i(t)}{(N - Y_{(j)}(t))} = Y_i(t)\lambda \Delta t/s. \] Hence, when \( B \) is large enough, we have

\[
E\{Y_i(t + \Delta t) - Y_i(t)|G_t\} =
\begin{cases} 
-Y_i(t)\lambda \Delta t/s & i = 0, \ldots, s - 1 \\
(Y_{i-s}(t) - Y_i(t))\lambda \Delta t/s & i = s, s + 1, \ldots
\end{cases}
\]

Dividing by \( N \cdot \Delta t \) on both sides, we get

\[ z_i'(t) = \begin{cases} 
-z_i(t) \cdot \frac{\lambda}{s} & i = 0, 1, \ldots, s - 1 \\
(z_{i-s}(t) - z_i(t)) \cdot \frac{\lambda}{s} & s, s + 1, \ldots
\end{cases} \quad (5.9)
\]

As for the change of \( X_i \), a new segment with degree \( s \) is added whenever a segment is injected. Thus, the expected block increase is \((N - Y_{(j)}(t)) \cdot \frac{\lambda}{s} \cdot \Delta t\) for degree-\( s \) segments and zero for other segments. When \( B \) is large, we have

\[ w_i'(t) = \delta_{is} \cdot \frac{\lambda}{s} \cdot (1 - z_{(f)}(t)) = \delta_{is} \cdot \frac{\lambda}{s}, \quad i = 1, 2, \ldots. \quad (5.10)\]

Putting (5.5), (5.7), and (5.9) together, and (5.6), (5.8), and (5.10) together, we finally arrive at the following differential equation systems:

\[
\begin{align*}
Z_0' &= \frac{-z_0}{1-z_B} \cdot (1-z_0)\mu - z_0 \cdot \frac{\lambda}{s} + z_1 \gamma, \\
Z_i' &= \frac{z_{i-1} - z_i}{1-z_B} \cdot (1-z_0)\mu - z_i \cdot \frac{\lambda}{s} + ((i+1)z_{i+1} - iz_i)\gamma, \\
& \quad i = 1, 2, \ldots, s - 1, \\
Z_i' &= \frac{z_{i-1} - z_i}{1-z_B} \cdot (1-z_0)\mu + (z_{i-s} - z_i) \cdot \frac{\lambda}{s} \\
& \quad + ((i+1)z_{i+1} - iz_i)\gamma, \quad i = s, s + 1, \ldots, \quad (5.11)
\end{align*}
\]
5.4. A DIFFERENTIAL EQUATION CHARACTERIZATION

\[
\begin{align*}
    w_i' &= \frac{(i - 1)w_{i-1} - iw_i}{e(t)} \cdot (1 - z_0)\mu + (\gamma + (i + 1)w_{i+1} - iw_i)\gamma \\
         &\quad + \delta_{is} \cdot \frac{\lambda}{s}, \quad i = 1, 2, 3, \ldots 
\end{align*}
\]

(5.12)

To characterize the server collection process, we introduce the segment collection matrix \( M = (M^j_i(t)) \), \( i = 1, 2, \ldots, j = 0, 1, \ldots, s \), where \( M^j_i(t) \) denotes the number of degree-\( i \) segments that have \( j \) (coded) blocks collected by the servers already. Let \( m = (m^j_i(t)) := \left( \frac{1}{N} M^j_i(t) \right) \) be the rescaled segment collection matrix. If we only consider the effects of segment encoding, transfer, and block deletion, the behavior of \( m^j_i(t) \) is similar to that of \( w_i(t) \):

\[
\begin{align*}
    \frac{dm^j_i}{dt} &= \frac{(i - 1)m^j_{i-1} - im^j_i}{e(t)} \cdot (1 - z_0)\mu + (\gamma + (i + 1)m^j_{i+1} - im^j_i)\gamma, 
\end{align*}
\]

(5.13)

Consider server collection operation. By linear approximation, each block is collected by the servers with a probability proportional to its degree in \( G_t \). Since \( M^j_i \) increases by one and \( M^{j-1}_i \) decreases by one whenever a server obtains a block from a degree-\( i \) segment which already has \( (j - 1) \) blocks collected by the servers, the expected change of \( M^j_i(t) \) is

\[
    \mathbf{E}\{M_i(t + \Delta t) - M_i(t) | G_t\} = \frac{i(M^j_{i-1}(t) - M^j_i(t))}{E(t)} \cdot cN\Delta t
\]

(5.14)

Dividing by \( N \cdot \Delta t \) on both sides, including the cases for \( j = 0, s \), and also considering
the effect of segment injection, we get for $i = 1, 2, \ldots$

\[
\frac{dm^j_i(t)}{dt} = \begin{cases} 
-c \cdot \frac{im^0_i(t)}{e(t)} + \frac{\delta_is}{s} & j = 0, \\
\frac{c}{e(t)} \cdot i(m^j_i(t) - m^j_i(t)) & j = 1, \ldots, s - 1, \\
\frac{c}{e(t)} \cdot im^j_i(t) & j = s.
\end{cases}
\] (5.15)

Combining all the effects in (5.13) and (5.15) together, we have for $i = 1, 2, \ldots$:

\[
\begin{align*}
\frac{dm^0_i(t)}{dt} &= \frac{(i - 1)m^0_{i-1} - im^0_i}{e(t)} \cdot (1 - z_0(t)) \mu \\
&\quad + ((i + 1)m^0_i - im^0_i) \cdot \gamma - \frac{im^0_i(t)}{e(t)} \cdot c + \frac{\delta_is}{s} \cdot \frac{\lambda}{s}, \\
\frac{dm^j_i(t)}{dt} &= \frac{(i - 1)m^j_{i-1} - im^j_i}{e(t)} \cdot (1 - z_0(t)) \mu \\
&\quad + ((i + 1)m^j_i - im^j_i) \cdot \gamma + \frac{i(m^{j-1}_i - m^j_i(t))}{e(t)} \cdot c, \\
\frac{dm^s_i(t)}{dt} &= \frac{(i - 1)m^s_{i-1} - im^s_i}{e(t)} \cdot (1 - z_0(t)) \mu \\
&\quad + ((i + 1)m^s_i - im^s_i) \cdot \gamma + \frac{im^{s-1}_i}{e(t)} \cdot c.
\end{align*}
\] (5.16)

Our analytical results will be drawn from (5.11), (5.12), (5.16), which characterize system evolution.

\section*{5.5 Performance Evaluation}

In this section, we evaluate the proposed protocol in terms of session throughput, data delivery delay, resource consumption overhead, and loss resilience. We derive analytical
results with respect to these metrics from the differential equations and provide more insights with simulations. Based on the analytical and simulation results in this section, we discuss how to choose parameters for the proposed protocol in various settings.

Let us consider the steady-state network by setting the derivatives in (5.11), (5.12), (5.16) to zero. Let \( \tilde{z}_i(t) \), \( \tilde{w}_i(t) \), \( \tilde{m}_{ij}(t) \) and \( \tilde{e}(t) \) denote the steady-state value for \( z_i(t) \), \( w_i(t) \), \( m_{ij}(t) \) and \( e(t) \). First, we can show the storage overhead of the indirect collection protocol is upper bounded by \( \mu/\gamma \) in the following theorem.

**Theorem 11 (Storage Overhead)** Assume the buffer size \( B \) is large enough, regardless of the value of \( s \), in steady-state network, the average number of (coded) blocks in a peer’s buffer is

\[
\rho = (1 - \tilde{z}_0)\mu/\gamma + \lambda/\gamma,
\]

(5.17)

Moreover, the average storage overhead of a peer is

\[
\text{Overhead} = (1 - \tilde{z}_0)\mu/\gamma < \mu/\gamma.
\]

(5.18)

where \( \tilde{z}_0 \) is given by \( \tilde{z}_0 = e^{-(1-\tilde{z}_0)\mu/\gamma-\lambda/\gamma} \) for \( s = 1 \), and is given by the steady-state solution to (5.11) for \( s \geq 2 \).

**Proof:** Setting the derivatives in (5.11) to zeros, after tedious yet straightforward deduction, we can get \( \tilde{z}_i = \tilde{z}_0 \rho^i/i! \), \( i = 0, 1, \ldots, B \), where \( \rho = (1 - \tilde{z}_0)\mu/(1 - \tilde{z}_B)\gamma + \lambda/\gamma \). And \( \tilde{z}_0 \) can be determined through \( \sum_{i=0}^{B} \tilde{z}_i = 1 \). If buffer size \( B \) is large enough, we have \( 1 = \sum_{i=0}^{B} \tilde{z}_i = \tilde{z}_0 \cdot \sum_{i=0}^{B} \rho^i/i! \approx \tilde{z}_0 \cdot e^\rho \). Thus, \( \tilde{z}_0 \approx e^{-\rho} \). Since \( \tilde{z}_B \approx 0 \) when \( B \) is large, we have \( \rho = (1 - \tilde{z}_0)\mu/\gamma + \lambda/\gamma \).

Take the summation of all \( \tilde{z}_i \), we get the rescaled total number of edges in the steady state as

\[
\tilde{e}(t) = \sum_{i=0}^{B} i\tilde{z}_i = \rho \cdot \sum_{i=1}^{B-1} \tilde{z}_0 \rho^{i-1}/(i-1)! = \sum_{j=0}^{B-1} \tilde{z}_j \cdot \rho = (1 - \tilde{z}_B)\rho \approx \rho,
\]
when $B$ is large. Since the number of edges in $G$ equals to the total number of block copies in the network, we have the average number of block copies at each peer equals to
\[ \frac{1}{N} \cdot \sum_{i=0}^{B} i \tilde{Y}_i = \sum_{i=0}^{B} i \cdot \tilde{Y}_i = \sum_{i=0}^{B} i \cdot \tilde{z}_i = \tilde{e}(t) = \rho. \]

Theorem 11 shows the storage overhead of the proposed protocol can be limited to a relatively small value by setting bandwidth $\mu$ and block deletion rate $\gamma$ appropriately. In our simulations, the $\mu/\gamma$ is set to be less than 20.

We define the session throughput as the actual rate (blocks/unit time) at which servers obtain original data, which is the rate at which the servers reconstruct segments of statistics data multiplied by segment size $s$. We can show that if the fluctuating traffic is modeled as Poisson processes at the peers, with the proposed indirect collection mechanism, servers provisioned with bandwidth that only handles an average traffic can achieve a throughput close to the throughput capacity for the session.

**Theorem 12 (Session Throughput)** The throughput capacity of the session is the aggregate server capacity: $C = c_s \cdot N_s = c \cdot N$. For the non-coding case ($s = 1$), in steady-state network, the session throughput is given by
\[ \text{Throughput}(1) = N \lambda \cdot (1 - \frac{1}{\theta_+}), \] (5.19)

where $\theta_+$ is the maximum root of $\alpha_2 x^2 + \alpha_1 x + \alpha_0 = 0$, $\alpha_0 = -q \gamma$, $\alpha_1 = q \gamma + \gamma + \frac{\sigma}{\rho}$, $\alpha_2 = -\gamma$, and $q = 1 - \lambda/\rho \gamma$. Whereas for the coded case with a segment size $s$ ($s \geq 2$),
\[ \text{Throughput}(s) = N c \cdot (1 - \sum_{i=1}^{\infty} i \tilde{m}_s^i(t)/\rho), \] (5.20)

where $\tilde{m}_s^i(t)$ is the steady-state solution to (5.16).

**Proof:** The collection efficiency of servers $\eta$ in the long run equals to the probability
5.5. PERFORMANCE EVALUATION

that a server collects a block from a segment needed by the servers at each trial in the steady state, which equals to the probability that a server picks a segment whose blocks have been pulled for less than \( s \) times. Thus, we have \( \eta = 1 - \sum_{i=1}^{\infty} \tilde{m}_i(t)/\tilde{e}(t) \). Thus, the session throughput is given by \( C \cdot \eta = Nc\eta \). When \( s = 1 \), we can explicitly solve for \( \tilde{m}_i(t) \) and \( \tilde{e}(t) \) to obtain \( \eta(1) = \frac{\lambda}{c} \left( 1 - \frac{1}{\theta c} \right) \) (5.19).

![Figure 5.6: Session throughput as a function of segment size \( s \). \( \lambda = 20, \mu = 10, \gamma = 1. \)]

We plot the numerical results on throughput according to Theorem 12 as well as the simulation results in Fig. 5.6. In the \( Y \)-axis, we normalize the session throughput by dividing it by \( N \cdot \lambda \), that is the aggregate peer upload demand. Each dashed horizontal line in the figure denotes the throughput capacity for a certain value of \( c \).

It is clearly shown in Fig. 5.6 that by increasing segment size \( s \), session throughput can be made close to the throughput capacity under that \( c \) value. This is because when network coding is used, all coded blocks from a segment are equally useful, and the probability that servers collect redundant blocks using the coupon-collector algorithm falls down. This is the essential insight behind Theorem 12. Moreover, the use of a small segment size (e.g. around 20~30) is sufficient to achieve high throughput as shown in the figure, which indicates an acceptable computational complexity. It is also worth
noting that it is harder for the throughput to approach its capacity (dashed lines) as the normalized server capacity $c$ increases, since the benefit of an indirect mechanism is more salient when server capacity is insufficient.

![Figure 5.7: Session throughput as a function of $\mu$ under different scenarios. $\lambda = 8$, $\gamma = 1$.](image)

We also simulate to study the session throughput in a dynamic network. Peer dynamics is simulated by a replacement model [20], [23]: each peer is assigned a random lifetime $L$ and a peer leaves the network upon the expiration of its lifetime. A new peer will join at the same time to replace the departed peer. Peer lifetime follows the exponential distribution here with mean $L$. Such a model allows us to decouple the impact of the change in the number of online peers and fully focus on the effect of their dynamic nature.

The impact of churn on session throughput is plotted in Fig. 5.7 under various parameter settings. When server capacity compares to user demands ($c = 8$), throughput is degraded by severe peer churn (dashed lines) when a larger segment size is used, and when more buffering is performed with a larger $\mu$. This is because, buffering by gossiping and network coding is actually not needed when server capacity is high enough, whereas the use of a larger segment size in this case could make segments become undecodable.
when peers abort too frequently. However, when server capacity is insufficient, i.e., $c/\lambda$ is small, servers cannot collect all the generated data immediately anyway. Introducing a higher data redundancy will aid the collection of these data in a delayed fashion before they disappear. Thus, when server capacity is insufficient, throughput benefits from having a larger segment size $s$ and higher peer bandwidth $\mu$, even in the presence of peer churn. In addition, we can also see from Fig. 5.8 that changing $\gamma$ does not affect the throughput significantly.

![Figure 5.8: Session throughput as a function of $\mu$ and $\gamma$. $\lambda = 20$, $c = 20$.](image)

We define block delay as the delivery delay of a segment divided by the segment size, that is the average delivery delay of each original block. We can derive block delay using Little’s Theorem in queueing theory based on the session throughput.

**Theorem 13** (Block Delivery Delay) For the non-coding case ($s = 1$), in steady-state network, if an original block is eventually delivered to the server, the average time from the injection of this block to its delivery is

$$T(1) = -\frac{1}{\gamma q} \cdot \frac{1}{1 - \theta_+} \ln(1 - q) - \frac{1}{\sigma \gamma q} \ln(1 - \theta_-)$$

(5.21)
5.5. PERFORMANCE EVALUATION

where \( \theta_+ (\theta_-) \) is the maximum (minimum) root of \( \alpha_2 x^2 + \alpha_1 x + \alpha_0 = 0 \), \( \alpha_0 = -q \gamma \), \( \alpha_1 = q \gamma + \gamma + \frac{\xi}{\rho} \), \( \alpha_2 = -\gamma \), and \( q = 1 - \lambda / \rho \gamma \). Whereas for the coded case \( s \geq 2 \), the average delivery delay for each original data block is

\[
T(s) = \frac{\sum_{i=1}^{\infty} \tilde{w}_i}{\lambda} - \frac{\sum_{i=1}^{\infty} \tilde{m}_i}{\lambda \sigma(s)}, \tag{5.22}
\]

where \( \tilde{w}_i \) is the steady-state solution to (5.12), \( \sigma(s) = \text{Throughput}(s)/N \lambda \).

Proof: Let \( \sigma(s) = \text{Throughput}(s)/N \lambda \). Let \( T_L \) denote the average time from the injection of a segment to its extinction from the network. By Little’s Theorem, \( T_L = \sum_{i=1}^{\infty} \tilde{X}_i / \lambda N \lambda = s \sum_{i=1}^{\infty} \tilde{w}_i / \lambda \), where \( \sum_{i=1}^{\infty} \tilde{X}_i \) is the number of distinct segments in the network and \( N \lambda / s \) is segment injection rate. We call a segment a good segment if the segment has been pulled for \( s \) times and it is still available in the network. Similarly, the average time during which a segment is a good segment is \( T_M = \sum_{i=1}^{\infty} \tilde{M}_i / \lambda N \lambda = s \sum_{i=1}^{\infty} \tilde{m}_i / \lambda \sigma(s) \). Hence, the average block delay \( T = (T_L - T_M)/s = \sum_{i=1}^{\infty} \tilde{w}_i / \lambda - \sum_{i=1}^{\infty} \tilde{m}_i / \lambda \sigma(s) \). Similarly, we can explicitly solve the differential equations (5.11), (5.12), (5.16) to obtain (5.21).

Fig. 5.9 shows the numerical results on block delay. We can see that there is a peak of block delay around \( s = 5 \). The reason is that without coding \( s = 1 \), although many blocks are lost due to the small throughput, yet a block is collected immediately if it can be collected as servers do not have to buffer an sufficient number of (coded) blocks for a segment to be reconstructed. When a larger segment size is used, servers tend to collect blocks from different segments in an alternating fashion, resulting in longer delay to reconstruct either one of them. However, when \( s \) is sufficiently large, the delay decreases again, as blocks in the network tend to belong to a few number of large segments. Taking into consideration of both throughput and delay, a segment size between 20 and 40 is
Figure 5.9: Average block delivery delay $T$ for different values of $s$. $\lambda = 20$, $\mu = 10$, $\gamma = 1$.

preferred. Fig. 5.10 shows a longer TTL for the blocks, that is a small $\gamma$, would induce longer delay. However, since the timeliness is not strictly required in statistics collection, we still want to set $\gamma$ to small values to keep a certain buffering level in the network for better resilience to peer departures.

Figure 5.10: Average block delay $T$ as a function of $\mu$ and $\gamma$. $\lambda = 20$, $c = 20$.

Finally, we show that at the end of the stream collection session, there is a guarantee of a certain amount of data buffered in the network so that the servers can still collect them in a delayed fashion.
Theorem 14 Suppose the streams of upload requests end in steady-state network, then the amount of data in terms of original blocks saved up in the network for future delivery is

\[ S = N \cdot s \sum_{i=s}^{\infty} (\tilde{w}_i - \tilde{m}_i^s), \quad (5.23) \]

where \( \tilde{w}_i \) is the steady-state solution to (5.12).

Proof: The number of segments buffered for future delivery equals to the number of decodable segments that have not been reconstructed by the servers yet. The number of decodable segments is \( \sum_{i=s}^{\infty} \tilde{X}_i = N \cdot \sum_{i=s}^{\infty} \tilde{w}_i \). Among these segments there are \( \sum_{i=s}^{\infty} \tilde{M}_i^s = N \sum_{i=s}^{\infty} \tilde{m}_i^s \) already reconstructed by the servers. Upon the decoding of a segment, \( s \) original data blocks are reconstructed at the servers. Thus, we have \( S = N \cdot s \sum_{i=s}^{\infty} (\tilde{w}_i - \tilde{m}_i^s) \). \( \square \)

Figure 5.11: Data saved in each peer, or the number of original blocks buffered in each peer that have not been collected. \( \lambda = 20, \mu = 10, \gamma = 1 \).

We plot the number of original blocks buffered in each peer that have not been collected. This amount of data is saved up for future delivery when statistics data are streamed at each peer at a much lower rate. According to Theorem 11, the total amount of data buffered in the peers are the same when we use different segment sizes. However,
as the segment size increases, network throughput increases according to Theorem 12. As more data in the network are already reconstructed in the stream collection session, each peer buffers less “fresh” segments that have not been reconstructed by the servers yet. However, regardless of the segment size, the system benefits by using an indirect collection mechanism which always buffers a guaranteed amount of data for future delivery when the volume of traffic falls down.

5.6 Summary

In this chapter, we propose an indirect statistics collection mechanism to ensure the collection of large volumes of data from a large number of end hosts in a delayed fashion. Specifically, we require that peers exchange data blocks with their neighbors using random network coding and probabilistic gossip protocols. We further propose efficient buffer management and server collecting algorithms to make the indirect mechanism practical in a more demanding setting, where statistics data are generated at each peer as a stream. We analyze the system performance based on a random bipartite graph process and use simulations to corroborate our analytical results as well as to shed more insights into system performance when peers join and leave the session frequently.
Chapter 6

Conclusions and Future Work

In this thesis, we have studied the fundamental benefit of randomized network coding as a completely decentralized data exchange algorithm in P2P content dissemination systems, including P2P file sharing and indirect statistics collection applications. We focus on analyzing how network coding can preserve and enhance data diversity and availability in such systems, which are crucial factors that affect data dissemination efficiency, especially when the peers are inherently volatile and the departures of peers may cause data loss from the network in an unpredictable way.

As it is computationally costly to do coding operations instead of simply forwarding original data blocks, network coding is normally done within segments, each of which only contains a subset of the entire block set for dissemination. With such computational cost in consideration, we give a quantitative description of the resilience gain that network coding can provide in a dynamic network environment, as segment size, which is the number of blocks in a segment for coding, varies from one to a large number. By varying the segment size, we are effectively developing an analytical framework that incorporates non-coding gossiping algorithms on one end and gossiping with pure network coding on
CHAPTER 6. CONCLUSIONS AND FUTURE WORK

We introduce a typical fairness measurement originally used in classical resource allocation problems to evaluate the variation with regard to the distributions of different blocks, and find by stochastic modeling that this block distribution variation is inversely proportional to the segment size for coding in steady-state networks that employ a gossiping-like transmission protocol. We further relate the block distribution variation with the download time, that is the time needed for an individual peer to download the entire content in such a network. A similar inversely proportional relationship between the download time and segment size is also observed through simulations.

Our findings have shown that small segment sizes — around 20 in our simulations even with high peer volatility — suffice to realize the major benefit of network coding in terms of reducing download times and preserving block diversity when seeds are available. If the seeds depart the system at a certain point, although the content will eventually become incomplete due to peer dynamics, content lifetime increases and content loss rate decreases in trend as the segment size for coding increases.

The second part of the thesis concerns with the application of network coding for statistics data collection in large-scale peer-to-peer (P2P) applications, which is necessary in such systems for network administrators to diagnose system performance and network outage. With current logging server solutions, such statistics collection is not scalable to a up to millions of participating peers, due to the bottleneck at servers. Moreover, due to dynamic peer memberships, the data at a certain peer may not even have been collected before it has departed from the network.

To solve these problems, we propose an indirect data collection mechanism, which requires peers to exchange data blocks with their neighbors using probabilistic gossiping.
protocols. The servers can therefore collect large volumes of data from a large number of end hosts in a delayed fashion. Leveraging the resilience gain that network coding can provide in dynamic P2P networks, we further let peers exchange data in their coded form using segment-based network coding and find an almost exponential improvement of data persistence in dynamic networks as we increase the segment size for coding.

We have shown that by utilizing only a small portion of peer resources to form a “buffering” pool, the transmission of statistics from peers to logging servers is “cushioned,” such that server bandwidth is provisioned to handle only average load, rather than the peak. In addition, as data blocks are distributed across the network, now the servers can retrieve buffered data in a delayed fashion that they were not able to collect during a flash crowd scenario.

Equipped with our new understanding into the resilience gain that segment-based network coding can offer in P2P content distribution with dynamic peer memberships, we may use network coding to benefit the design of similar systems in future work. One possible application scenario is the P2P video on demand (VOD) systems. Such systems are similar to P2P file-sharing systems in that they also rely on the unreliable and highly dynamic participating peers to store all the data blocks. Segment-based network coding can therefore help to enhance data persistence and block diversity in the such networks. However, P2P VOD systems have more challenging design objectives of ensuring sequential playback and the ability to start playback from an arbitrary point in the video. These challenges have saved a possible direction for our future research.
Bibliography


