AGE OF INFORMATION FOR REAL-TIME NETWORK APPLICATIONS

by

JING ZHONG

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Roy D. Yates and Emina Soljanin

And approved by

New Brunswick, New Jersey

OCTOBER, 2019

ABSTRACT OF THE DISSERTATION

Age of Information for Real-time Network Applications

By JING ZHONG

Dissertation Director:
Roy D. Yates and Emina Soljanin

Driven by recent advances in ubiquitous connectivity and pervasive computing, real-time status updates to interested recipients have become increasingly popular in streaming applications. These status updating systems all share a common need: the recipients want their information about the sources to be as fresh as possible. This thesis aims to analyze a recently proposed information freshness/timeliness metric, age of information (AoI), in various real-time network applications, and optimize the corresponding AoI metric given the network constraints.

In this thesis, we model the real-time status updating system as source-receiver pairs connected through the networks. The first fundamental problem we consider is how timely update messages should be compressed based on the given network capacity. Different from traditional data compression techniques that shorten the average codeword length, we show that the optimal lossless compression scheme for fast message delivery depends on higher moments of the codeword length due to the queueing delay. The AoI-optimal codebook can be constructed by a recursive search algorithm based on the convex AoI penalty function.

In ultra-dense network deployments, real-time updates are expected to be distributed to massive numbers of receivers via the nearby storage nodes at the network edge. Thus, the second fundamental problem we address is how the real-time updates should be replicated and distributed to multiple edge storage nodes through multicast networks.

We answer the question by evaluating the average AoI at a receiver which has access to a random number of edge storage nodes, given the distribution of the random network delay. This system model is also applicable to time-sensitive content updates in Dynamo-type distributed storage systems in which the write and read operations go to multiple storage nodes simultaneously. We derive the AoI-optimal quorum mechanism that balances the data consistency and operation latency.

Beyond the study of the two fundamental problems in AoI, we extend the similar AoI analysis to applications with resource contention and scheduling. We examine a remote cache updating system in which the local server maintains snapshots of the content at different remote sources and updates those snapshots according to a constrained rate. We compare AoI to an alternative Age of Synchronization freshness metric and evaluate the optimal rate allocation scheme for the two different age metrics. We also examine the edge cloud computational offloading system with multiple users, and investigate the scheduling policy for incoming jobs in a vision processing application at an edge server. We show that a greedy scheduling policy is optimal for a class of AoI-related penalty functions.

Acknowledgements

The completion of this Ph.D. dissertation would not have been possible without the invaluable support and guidance from my advisors, Prof. Roy D. Yates and Prof. Emina Soljanin.

My first greatest gratitude goes to Prof. Roy D. Yates. He opened the doors to the world of AoI research for me in our first meeting almost six years ago. Since then, I have always been inspired by the depth of his technical insight, his rigorous approach towards research, and his passion about exploring and solving problems. One of his greatest advice is to first convert any complex problem to a simple core problem before solving it.

I would also like to express my deepest appreciation and admiration to Prof. Emina Soljanin. I was extremely fortunate to be one of her first batch of students at Rutgers, and to have her guidance towards the new world of distributed systems. She brought an important applied perspective to this research and made me aware of how to logically and organizationally approach a research problem.

Besides, I am also very grateful to the rest of the dissertation committee. It's my honor to have Prof. Predrag Spasojevic and Prof. Salim El Rouyaheb to be my committee members since the Ph.D. qualification exam. Their insightful suggestions have driven me to broaden my research and progressively improve this dissertation. I would also like to express my special gratitude to Prof. Bo Ji for serving as my external member and the thoughtful comments.

This journey towards Ph.D. would not have been such amazing and smooth without the support and company from all my friends, fellow students and collaborators. I cannot thank them enough for all the interesting discussions and fun moments we had. And I would like to thank all the WINLAB faculties and staffs for keeping up such a lively and productive environment.

Last but not least, I thank my family for their support and continuous encouragement throughout my life.

Table of Contents

Abstract ii

Acknowledgements iv

List of Tables ix

List of Figures x

1. Introduction 1

1.1. Status Updating 1

1.2. Age of Information Metric 4

1.3. The Ongoing History of the Age of Information 7

1.3.1. AoI Evaluation in Different Network Models 7

1.3.2. AoI-based Optimization and Scheduling 8

1.3.3. AoI and Information Theory 10

1.3.4. Other AoI Applications 11

1.4. Road Map 11

2. Compressing Timely Updates 14

2.1. Streaming Data Compression and Recent Results 14

2.1.1. Notation 17

2.2. Deterministic Symbol Arrivals Without Feedback 18

2.2.1. AoI for Fixed-to-Variable Codes 18

2.2.2. Age-optimal Codes 25

2.2.3. Numerical Evaluation 28

2.3. Deterministic Symbol Arrivals With Feedback 31

2.3.1. Passive Blocklength Adaptation 35

2.3.1.1. Lazy Parsing Policy 35

2.3.1.2. Just-in-time Parsing Policy 36

2.3.2. Constrained Maximum Blocklength 38

2.3.3. Numerical Evaluations 40

2.4. Random Symbol Arrivals 43

2.4.1. Age of Information Stream 45

2.4.2. Age of Source Sequence 49

2.4.2.1. Prefix Encoding 49

2.4.2.2. Embedded Encoding 52

2.4.2.3. Analysis for Geo/G/1 Vacation Queue 53

2.4.2.4. Optimizing Prefix Encoding 58

2.4.2.5. Optimizing Embedded Encoding 59

2.4.3. Numerical Evaluations 61

2.5. Appendix 64

3. Distributing Timely Updates 71

3.1. Motivations 71

3.2. Related Work 75

3.3. Time-average Age Analysis 76

3.4. Multicast with Homogeneous Nodes 84

3.4.1. Earliest-count Restart Policy with Feedback 84

3.4.1.1. Notation for Order Statistics 84

3.4.1.2. Age Calculation 85

3.4.1.3. Shifted Exponential Model 90

3.4.1.4. Pareto Model 93

3.4.1.5. Connection with Dynamo-style Storage Updates 96

3.4.2. Threshold Restart Policy 98

3.4.2.1. Shifted Exponential Model 99

3.4.2.2. Pareto Model 101

3.4.3. Numerical Evaluation 103

3.4.4. Appendix 110

3.5. Multicast with Prioritized Nodes 119

3.5.1. Priority-count Restart Policy with Feedback 119

3.5.2. AoI at Priority Nodes 120

3.5.3. AoI at Non-priority Nodes 122

3.5.4. Numerical Comparisons 125

4. Resource Sharing For Multi-source Updating 127

4.1. Remote Cache Updating 127

4.1.1. Minimizing AoI and AoS 131

4.1.1.1. AoS minimization 134

4.1.1.2. AoI minimization 137

4.1.2. Numerical Comparisons for Two Freshness Metrics 138

4.1.2.1. Special Case: Two Sources 138

4.1.2.2. Many Sources 140

4.1.3. Possible Extensions 142

4.2. Job Scheduling in Real-time Edge Computing 142

4.2.1. Edge Computing Model and Age Penalty Function 144

4.2.2. Scheduling Policy 147

4.2.3. Performance Evaluation 152

4.2.4. Future Work 154

4.2.5. Appendix: Proof of Theorem 4.2.3. 155

5. Conclusions 159

5.1. Summary 159

5.2. Future Work 160

References 161

List of Tables

2.1. Example of codebook using prefix idleness encoding. 50

2.2. Example of codebook using embedded idleness encoding. 52

List of Figures

1.1. An overview of the status updating systems. 2

1.2. The simplest one-to-one active status updating model. X_i indicates the i-th update by the source. 4

1.3. Sample path of the AoI process $\Delta(t)$. The status regarding the source is generated at time marked by $\bullet$, and updated at the receiver at time marked by $\times$. 5

2.1. System diagram for streaming lossless source coding. 15

2.2. Example of streaming source coding system with deterministic symbol arrivals with three source symbols and blocklength $N = 2$. 19

2.3. Comparing experimental average AoI, the upper and lower bounds for deterministic symbol arrivals using Huffman codes. 25

2.4. The illustration of convex hull algorithm and the representation of codebooks in the coordinate. 26

2.5. Comparing the average AoI with Huffman codes (thin solid lines) and Age-optimal codes (dashed thick lines) with different blocklength N. . . 29

2.6. Sample path of the age $\Delta(t)$ (the upper envelope in bold) for backlog-adaptive streaming source coding with the just-in-time (JIT) policy. . . 32

2.7. Comparing backlog-adaptive source codes to fixed blocklength codes for two source models. For each source model, the entropy $H(X)$ is fixed but the channel rate R is varied. 40

2.8. Comparing just-in-time (JIT) parsing policy, its approximation with postponed arrivals, and the lazy update policy using Shannon code. . . 41

2.9. Example of average age for CMB with different τ using Shannon code. . 41

2.10. Examples of average age for CMB as a function of τ using Shannon code. Different ternary source with offered load $\rho = H(X)/R = 0.9$. 42

2.11. Example timeline of randomly arriving symbols generated by the source. The source symbols $X_1X_2X_3$ are shaded in color. 44

2.12. Example of the age of information stream (ISA) process with $|\mathcal{X}| = 3$ and blocklength $N = 2$. 46

2.13. Example of instantaneous age of source sequence (SSA) at the receiver of streaming source coding with $|\mathcal{X}| = 3$. The special idle flag ϵ is encoded only if there is no new arriving symbol and the buffer becomes empty. . 50

2.14. An example of the output process of the FIFO buffer in which the special codeword reserved for empty buffer signal ϵ is 101. 52

2.15. Comparing experimental results and the bounds for SSA with Zipf source using prefix idleness encoding. 61

2.16. Comparison between AoI-optimal codes and Huffman codes. 63

2.17. Comparison between prefix idleness encoding and embedded encoding for average peak SSA. 64

2.18. An example of the nodeset notation (on the left) and its equivalent tree notation (on the right). Each node (i, l) has width $\rho = 2^{-l}$ and weight $\mu = p_i(2l - 1)$. 69

3.1. Multicast systems for timely updates: the source sequentially transmit content updates to multiple storage nodes at the network edge with random delays. The next update $j + 1$ is generated right after the source terminates the transmission of update j. Each receiver has access to a random subset of r nodes and selects the freshest update from them. . . 74

3.2. Sample path of the age $\Delta(t)$ with success probability p. Update delivery instances are marked by $\bullet$ 79

3.3. Comparison of the average age between Earliest-count (EC) and threshold policies using shifted exponential service time with $c = 1$. $\times$ marks the average age as a function of w for EC policy. Solid line marks the approximate average age, and $\circ$ marks the minimized approximate age $\hat{\Delta}$. Dashed line marks the average age using the optimal threshold restart policy. 104

3.4. Comparison of the average age between Earliest-count (EC) and threshold policies using Pareto service time with $b = 1$. $\times$ marks the average age as a function of w for EC policy. Solid line marks the approximate average age, and $\circ$ marks the minimized approximate age $\hat{\Delta}$. Dash line marks the average age using the optimal threshold restart policy. 105

3.5. Minimum average age obtained by optimizing counter w^* for different node numbers n in the system using shifted exponential service model with $c = 1$ and access number $r = 1$. 106

3.6. Comparison of the average age among receivers with different access number r using shifted exponential service time with $c = 1$. $\circ$ marks the minimized average age. Dash line marks the average age using the optimal threshold restart policy. 107

3.7. Minimum average age obtained by optimizing the threshold τ^* as a funciton of the access number r. 109

3.8. Average age as a function of the access number r using different threshold policies. 110

3.9. Alternative system with prioritized-count restart policy: the k nodes in the priority group are shaded. The transmission of update $j + 1$ is initiated only after update j is delivered to all k nodes in the group. . . 120

3.10. Average age versus the priority group size k. circle $\circ$ marks the priority group and cross $\times$ marks the non-priority group. The lower bound for priority group is shown as dashed line. 125

3.11. Average age versus the exponential shift c with $\lambda = 2$. The priority group size $k = 5$. 126

4.1. Diagram of local cache refresh with n remote sources. 128

4.2. Sample paths of the AoS and AoI. ● indicates the update at the remote source and $\times$ indicates the data update is synchronized at the local server. 129

4.3. Sample paths of AoI with infrequent source updates: no update in the 2nd and 3rd intervals. ● indicates the update at the remote source and $\times$ indicates the data update is synchronized at the local server. 132

4.4. AoS vs. the portion of refresh frequency f_1/f in a two sources model with unequal popularities $p_1 = 1, p_2 = 5$, and source update rates $\lambda_1 = 1, \lambda_2 = 10$.138

4.5. AoI vs. the portion of refresh frequency f_1/f in a two sources model with constraint $f = 10$, popularities $p_1 = 1, p_2 = 5$, and source update rates $\lambda_1 = 1, \lambda_2 = 10$. 138

4.6. AoS-optimal allocation policy f_1/f vs. total refresh frequency constraint f. The two source update rates are given as $\lambda_1 = 1$ and $\lambda_2 = 10$. 139

4.7. Optimal average age vs. the ratio of two source update rates $\beta = \lambda_1/\lambda_2$ in a two sources model with equal popularity. The total source update rates are constrained by $\lambda_1 + \lambda_2 = 10$, and the local refresh rate is constrained by $f = 10$. 140

4.8. The average age Δ_i for each source i in a system with $n = 100$ nodes, and Zipf distribution scale parameter $s = 1$. 141

4.9. Example of edge cloud traffic scheduling system. 142

4.10. Sample path of age processes $\delta_i(t)$ and $\Delta_i(t)$ at the buffer and the user monitor. 148

4.11. Average age vs. average job arrival time with different scheduling policies.150

4.12. Average age vs. number of users with different policies. 150

4.13. Average age of each user in MWAR policy. 153

4.14. The fraction of service corresponding to each user in MWAR policy. . . 154

Chapter 1

Introduction

1.1 Status Updating

The recent technological advances in ubiquity and power of computing hardware and communication devices have engendered a wide variety of applications. These range from the pervasive mobile gaming, the rapid development of deep learning on computer vision and natural language processing, the health monitoring system with Internet-of-Things (IoT) devices, to the foreseeable autonomous cars in a connected mobile environment. The global data traffic generated by mobile devices reached 11.5 exabytes per month at the end of 2017, a 17-fold growth over the past 5 years [1]. Among all this mobile data, one significant trend is the rapid growth of the Machine-to-Machine (M2M) connections arising from IoT deployments. Mainly designed to support systems for real-time information monitoring and tracking, these M2M connections are used across different industries such as finance, healthcare, automotive and environmental sensing [1].

These real-time monitoring systems all share a common need: the monitor needs to track the status of the interesting sources and make sure its knowledge of the sources is as fresh and timely as possible. Thus, we refer to this type of system as a *status updating system*. Figure 1.1 illustrates a few examples of status updating systems. Low-cost environmental sensors will be equipped in both rural and urban areas to monitor air quality, soil moisture and vibration levels to prevent natural catastrophes such as landslides, forest fires and etc. For high frequency trading in financial markets, the execution of a trading algorithm relies on low latency to gather the most recent stock price. For instance, microwave long-distance network for ultra-low latency data

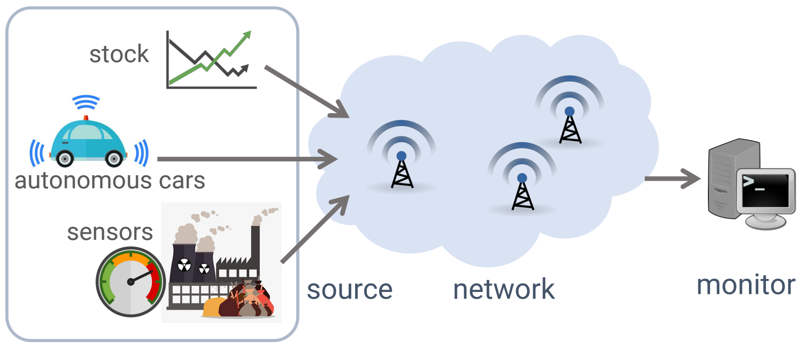


Figure 1.1: An overview of the status updating systems.

transmission have been built between key locations, such as Chicago and New York, to guarantee the timeliness of the information delivered to the traders [2]. Data exchanges between an autonomous vehicle and nearby vehicles and infrastructure can help to achieve better route planning and emergency controls.

In these examples, the source and the interested monitor are usually connected through a communication system. The information at the monitor lags behind the true source mainly due to the following two reasons. First is the inactivity of the source. Since the physical status at the source is unknown and generally varying in time, the source is required to sample its state and send it to the monitor as a status update. If the source hasn't update the monitor for a long time period, the monitor's knowledge of the source becomes stale. Second is the physical constraint of the network. Due to the uncertainty of the network, an update can be dropped in the network. Even though the update is successfully delivered to the monitor, it may experience a random delay in the network due to network congestion and propagation delay. In this case, the information update delivered to the monitor can be outdated if it spends a long time in the network.

The challenge in designing a status updating system leads to a more fundamental question: what should be the appropriate metric to quantify the timeliness, or information freshness, of the status updating systems. Since the monitor maintains a more-or-less outdated version of the status of the source, the correct metric should reflect how much the information at the monitor lags behind the source. Unfortunately, we will see that most commonly used network performance metrics, such as delay and throughput, are not suitable for describing the timeliness.

- **Delay:** Network delay measures the time elapsed from when a transmission starts to when that transmission is completed. Delay quantifies how quickly each update can be delivered to the monitor if an update is successfully delivered to the monitor. But in a lossy system in which some update packets are dropped in the network randomly, the average delay is undefined. Moreover, small network delay doesn't imply information freshness. For example, an autonomous car sends an update to another car every 10s. Each update may be delivered rapidly, but the information at the receiving car becomes stale since it doesn't hear from the sending car for a long time.

- **Throughput:** We refer to throughput as the average rate that information is successfully communicated through the channel/network. Maximizing the information throughput is not equivalent to making those information fresh. For example, in a queueing-type network that handles the arriving packets in a first-come-first-served way, one can always maximize the throughput by keeping the system busy all the time. In this case, an update is very likely to experience long network delay and the information conveyed in the update may becomes old.

- **Distortion/Error:** One can naturally connect the information freshness to a distortion or error metric, which can be defined as the distance between the ground truth status x at the source and an "estimate" of the status $\tilde{x}$ at the monitor. To choose a proper distortion metric, we need to know the statistics of dynamics at the source, but this is usually complicated and application dependent. Therefore, we seek a simpler metric that indicates the loss or uncertainty in time but not relying on the source statistics itself.

Given the our observations above, it's desirable to introduce a temporal freshness metric for real-time status updating systems.

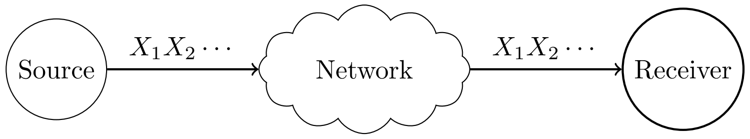


Figure 1.2: The simplest one-to-one active status updating model. X_i indicates the i-th update by the source.

1.2 Age of Information Metric

In complex status updating systems, there may be large number of source-destination pairs. To understand the updating system, we first assume the pairs are independent of each other and focus on the simplest model with one source and one receiver only, as shown in Figure 1.2. Since the source sends updates to the receiver proactively, we call this an *active* updating system. In contrast, an updating system is called *passive* if the receiver takes the responsibility to fetch the updates from the source. In [3], the age of information (AoI), or simply the *age*, is defined to measure the freshness of the information regarding the source at the monitor. When newest received update at time t at the receiver reflects the status or state of the source at time $u(t) \leq t$, we define the instantaneous age of information, or simply the age, as the random process

$$\Delta(t) = t - u(t). \tag{1.1}$$

We note that $\Delta(t)$ can be either a continuous time or discrete time stochastic process. There are two important observations in (1.1):

1. If the status information at the receiver is not updated, i.e. $u(t)$ is fixed, the age process $\Delta(t)$ grows linearly at unit rate with the passage of time t.

2. If the status information at the receiver is updated at time t, $u(t)$ is advanced to a new value, let's call it $u'(t) \geq u(t)$. The age process $\Delta(t)$ has a reduction from $t - u(t)$ to $t - u'(t)$.

Suppose the source generates status updates about its own information and sends these updates to the receiver through a communication network with random delay. Each update is time-stamped when it was generated at the source. Denoting the generation

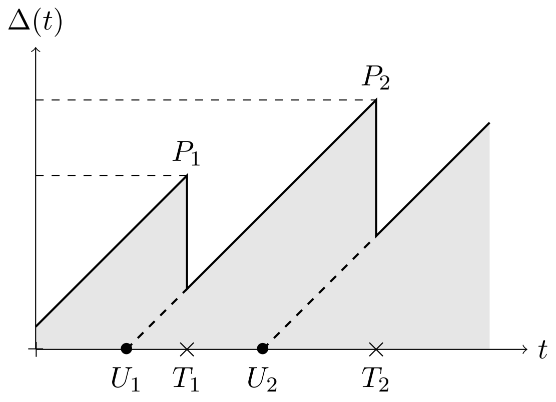


Figure 1.3: Sample path of the AoI process $\Delta(t)$. The status regarding the source is generated at time marked by $\bullet$, and updated at the receiver at time marked by $\times$.

time of each update i as U_i and the delivery time at the receiver as T_i, Figure 1.3 demonstrates a sample path of the age process in this type of system. The age at the receiver $\Delta(t)$ increases linearly with slope 1 without new update delivery. Once a new update i from the source is delivered to the receiver, the receiver's information regarding the source is refreshed, and the age is reset to $T_i - U_i$, which is the age of update i. In this thesis, we define the age of information (AoI) specifically as the instantaneous age $\Delta(t)$.

Definition 1.2.1. *The average AoI, or simply the average age, at the receiver is defined as the time-average of the stochastic process* $\Delta(t)$*,*

$$\Delta = \lim_{\tau\to\infty} \frac{1}{\tau} \int_{t=0}^{\tau} \Delta(t)\mathrm{d}t. \tag{1.2}$$

The average AoI was first introduced and evaluated for status updating through a single server with exponential service times and first-come-first-served (FCFS) discipline in [3]. That is, we simply replace the network in Figure 1.1 with a FCFS queue. It was shown that although the end-to-end delay is an essential part of the age, minimizing the age is different from simply reducing the average delay of every update. For a given service rate μ, large delay can be avoided by sending updates infrequently, which lowers the system utilization. However, this could result in higher age since updates are delivered infrequently and the most recent update at the receiver will frequently be stale.

On the other hand, sending updates too frequently will build up a long queue at the server, which leads to high average age since every update arrives at a congested queue and suffers a long waiting time. If updates arrive at the source in a memoryless way, it was shown that there exist an optimal update arriving rate $\lambda \approx 0.53\mu$ that minimizes the average age.

We also observe in Figure 1.3 that the age value $\Delta(t)$ reaches a local peak value right before the receiver's information is refreshed by a new update i at time T_i. Thus we define the *peak age* for each update i as

$$P_i = \lim_{t \to T_i, t < T_i} \Delta(t). \tag{1.3}$$

Different from the instantaneous age $\Delta(t)$, the peak age reflects the worst case of information staleness in the system [4].

Definition 1.2.2. *The average peak AoI, or simply the average peak age, at the receiver is defined as the time-averaged of the sequence P_i,*

$$\Delta^P = \lim_{N \to \infty} \frac{1}{N} \sum_{i=0}^{N} P_i. \tag{1.4}$$

The average AoI and the average peak AoI are the two most common age metrics in the literature, since these straightforward definitions quantify the freshness of the system strictly in time difference.

To further quantify the cost or penalty incurred by information staleness, we can define the cost as a function or functional of the age. The age dependent penalty function was first proposed in [5] and later generalized with the term cost of update delay (CoUD) in [6]. Some examples of the age penalty function f are illustrated as follows:

1. The exponential function of the age $f(\Delta) = e^{a\Delta} - 1$ where $a \geq 0$. This suits applications in which the need for information refresh is more desired as the information gets stale [7].

2. The fraction function $f(\Delta) = (a\Delta)/(a\Delta + b)$ that maps the age to the binary interval $[0, 1]$. This converts the age to a mission failure probability for many

attack-defense problems and remote control applications. [8]

If the destination is keeping track the status of a finite set of n sources, we may generalize the penalty to the age vectors $\boldsymbol{\Delta}(t) = [\Delta_1(t), \ldots, \Delta_n(t)]$. An example penalty $\mathcal{P}$ is the time-averaged of the sum of age penalty over all n sources as shown below

$$\mathcal{P} = \lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \int_{t=0}^{\mathcal{T}} \sum_{i=1}^{n} f(\Delta_i(t))\mathrm{d}t. \tag{1.5}$$

1.3 The Ongoing History of the Age of Information

1.3.1 AoI Evaluation in Different Network Models

The evaluation of the age metric in different networking systems is non-trivial since the analysis relies on many aspects of the system model, including but not limited to how the updates are generated at the source, how the transmitter or scheduler send the updates and how the service facility in the network processes the updates. In most cases, the service facility is modeled as a queue (or queues) with limited service capacity. Most of the age analysis tried to obtain an average age or average peak age expression and understand how the metric depends on different factors in the system model.

As mentioned in the begining, the study of status updating systems using the age metrics originates from [3], in which a source generates update messages as a Poisson process and send them to the destination through a FCFS queue with memoryless service times. Similar analysis is applied to status updates through a Last-Come-First-Served (LCFS) queue in [9]. It was shown that allowing preemption in service or waiting leads to lower average age, implying that newly generated updates should be served first to maintain freshness in most circumstances. In [10], multiple sources submits updates through a shared FCFS queue to a single destination, and the region of feasible ages at the receiver is derived by exploiting game theoretic results. The transmission of status updates through a random network with infinite servers is studied in [11]. Due to the random service time at different servers, the update may arrive at the destination out of order and the receiver distinguishes whether an arrived update is informative or not. Motivated by the observation that older packets are not informative to the status updating system, status updates through a M/M/1/1 and M/M/1/2 systems were examined in [4]. Here we note that M/M/1/k queue stands for a single server that only allows at most k customers / jobs in the queueing system, and M/M/1/1 stands for a queue with no waiting room for incoming customers.

Similar to the difficulty in queueing analysis involve general arrival or service distribution, the age analysis is usually quite challenge. In [12], the average peak age for multi-class status updating through a M/G/1 queue is obtained. The average age expression for Gamma distributed service time is derived in [13]. For Gamma service time, it's interesting to observe that preemption is not necessarily the optimal choice for the age. The study of M/G/1/1 preemptive queue for single source stream and multiple streams were conducted in [14] and [15]. In [16], the stationary distribution of the age for a G/G/1 queue is obtained by a sample path analysis of the age process, which holds for a wide range of arrival and service distributions.

In [17], the problem of multi-sources updating was revisited and a relatively simple mechanism named Stochastic Hybrid System (SHS) is introduced to compute the moments of the age. This method enables the calculation of age in rather complicated systems with multiple servers and complex network topologies such as [18]. In [19] and [20], SHS method was applied to the age evaluation of status updating with an energy harvesting server that collects randomly arriving energy units to transmit updates.

1.3.2 AoI-based Optimization and Scheduling

The analytical evaluation of age answers the question of how different factors in the system affects the age metric. Most literature seeks the optimal updating rates that minimizes the age assuming the source has no knowledge about the state of each update and all the updates are generated in a randomized fashion. An alternative question was asked in [3]: if the service time of each update is known to the source, and the source can control when to generate a new update to avoid jamming in the network, what is the optimal updating policy? Since our objective is to maintain the information freshness at the receiver, an intuitive policy is: when the previous update is delivered to the receiver and the channel is idle and ready for the next update, the source should generate an update immediately and send it to the receiver. This policy is usually called *just-in-time* (JIT) or *zero-wait* policy, which surprisingly turns out to be suboptimal under certain circumstances as shown in [21] and later generalized in [5]. If the service time of the previous update is small, submitting a new update right away doesn't bring the receiver fresh enough information but may waste the network resources for transmission. Therefore, the optimal policy is to be lazy and wait for a short time period if the previous update is delivered quickly. The detailed conditions for when this lazy policy stays optimal are in [5].

Compared to the evaluation of age, the scheduling and optimization research in the age literature tackles another aspect of the problem, which is when to send, and even what to send if there are multiple update candidates. In [22], it was shown that a preemptive Last-Generated-First-Served (LGFS) scheduling policy is optimal for an updating system with multiple servers. This policy is later shown to be optimal for any non-decreasing functional of the age in multi-hop network [23]. Status updating through multi-hop networks with link interference constraints was considered in [24], where a simple stationary policies in which links are activated according to a stationary probability distribution is considered. The scheduling of updates in an unreliable broadcast network with a base station and multiple receivers is considered in [25]. In this system, the base station accumulates updates from different sources but can only update at most one receiver at a time. It was shown that a greedy policy, which updates the source with maximum age, is the optimal policy for a symmetric network, and a Whittle Index form policy is obtained for general networks by converting the age minimization problem to a multi-arm bandit problem [26]. In [27], a similar problem was considered, except for the difference that an information update will be discarded if it is not selected by the base station for transmission. A single-hop network with several nodes transmitting updates to a base station (BS) through independent wireless links was considered in [28]. In this model, the BS selects at most one node for transmission at each time slot, with a minimum throughput/frequency constraint for the each node. Four different scheduling policies were compared in terms of the weighted sum of the average age.

1.3.3 AoI and Information Theory

Since AoI only quantifies the difference in time between the remote source and the receiver, several attempts have been made to connect AoI to distortion measure using information theoretic methods for time-correlated sources. In [29], the mutual information between the real-time source value and the delivered samples at the receiver is used as the distortion metric in time. It has been shown that the mutual information is a non-increasing function of the age, and an optimal sampling method that minimizes the time-averaged mutual information was obtained. A similar problem that allows the receiver to construct real-time estimate of the Wiener process at the source based on causally received samples was investigated in [30]. The optimal sampling policy for the Wiener process that minimizes the mean square error (MSE) was shown to be threshold type, and this problem can be reduced to an AoI optimization problem if the sampling times are independent of the observed Wiener process at the source.

Most of the recent research focuses on the queueing theoretic aspect of the age problem and assumes all the update packets are of equal size and can be transmitted through the network reliably. However, these assumptions are not always guaranteed in a communication system, especially in wireless links with constrained throughput. Thus, it is suggested to apply source and channel coding to the update messages, in order to limit the network resource usage and overcome channel errors or erasures. Nevertheless, traditional source and channel coding results are in fact not adequate for the age problem, since they are mostly obtained in an asymptotic regime which can be potentially bad for a ultra-low latency requirement. From an age perspective, it is more favorable to apply the information theoretic results in the finite blocklength regime.

A main focus of this thesis is on the lossless source coding on update messages. The related work on source coding and our contribution will be discussed in detail in Chapter 2. In [31], differential coding scheme was applied to the transmission of correlated update messages through erasure channel. Although the source can decide between the differential and the actual information, it was shown that the differential encoding scheme improves the age performance only if the receiver feedback is available at the source. Erasure codes were applied to the update packet to increase the probability of packet recovery in a binary erasure channel in [32]. It has been shown that the redundancy has to be carefully optimized, such that the updates can be recovered with sufficiently high probability, and the information contained in the packet is still fresh enough at the time of recovery. The results in [32] motivates our study about a more general timely update distribution model in Chapter 3. In [14], a similar coded update transmission problem with memoryless arriving updates at the source was studied as a special case of the M/G/1/1 queueing model.

1.3.4 Other AoI Applications

The evaluation of AoI has been applied to different real-time applications. In [33], the authors investigated the coexistence of DSRC and WiFi in vehicular network, in which the DSRC network aims to minimize the AoI of vehicular updates, and the WiFi devices seek to maximize the throughput. The random access problem with two networks was formed from a game theoretic perspective. In mobile crowd-learning services such as the navigation mobile app Google Waze [34], where the service providers rely on the user community to voluntarily collect and report real-time information for a set of points of interest (PoI), an important factor affecting the large-scale adoption of the service is the freshness of the crowd-learned information [35]. In order to ensure the freshness of the PoI states, a simple AoI-based reward mechanism was designed to incentive the selfish users to report the information in time. In [36], an AoI-optimized trajectory planning problem was studied in wireless sensor networks, where an unmanned aerial vehicle (UAV) is dispatched to collect data from the ground sensor nodes. We refer to [37] for a more thorough survey about the recent works on AoI research.

1.4 Road Map

In this thesis, we look at the status updating problem from three perspectives, including compressing the updates, distributing the updates, and the shared resource allocation for updates from multiple sources. These also represent three different key components to engineer in different real-time network applications. We investigate how these components affect the information freshness of the system, and design the engineering solutions to optimize the corresponding AoI metric.

In Chapter 2, we put our focus on the first fundamental problem: how the timely update messages should be compressed to ensure the information freshness. We evaluate the average age and average peak age for systems using lossless fixed-to-variable coding schemes, and obtain the AoI-optimal lossless compression scheme by a recursive search algorithm. For deterministic symbol arrivals, the key observation is that transmitting maximally compressed messages with large blocklength is a losing proposition, since the age benefits from the small blocklength and it depends on higher moments of the encoded sequence length due to queueing delay. For random symbol arrivals, the design of an AoI-optimized coding scheme requires some special treatments to encode the idleness at the source.

In Chapter 3, we evaluate the AoI of an updating system that distributes timely updates to multiple recipients through multicast networks. The interested receivers are located at the network edge, and they retrieve the most recent update information by accessing a subset of r edge nodes. We assume the source is able to choose how long it waits for the multicast transmission of the current update before it restarts and generates a new update. If the source waits for a long period for every update, each edge node is more likely to get the latest generated update and pass it to the receiver, but the information contained in the update is more likely to become outdated. In contrast, the receiver may fail to get an update from the connecting edge nodes if the source waits for a short period for every update. We derive the average age for any stationary update restart policy, and obtain the optimal restart strategy for two different policies: 1) the sources waits for the earliest deliveries of every update to a subset of nodes and 2) the source waits for a fixed time threshold. We show that the average age associated with these two policies coincide if both policies are optimized for the age. This system model is also applicable to time-sensitive content updates in a Dynamo-type distributed storage system [38] in which the write and read operations go to multiple storage nodes simultaneously.

In Chapter 4, we extend the AoI analysis to emerging applications in which the update service facility is shared among multiple sources thus resource allocation and scheduling kicks in. We examine a remote cache updating system where the local server maintains snapshots of the content at different remote sources and refreshes those snapshots with a constrained total rate. In order to keep the information about all the sources fresh at the local cache, we formulate the rate allocation problem and compare AoI to another age of synchronization freshness metric. We also investigate the edge cloud computing system supporting multiple users, and evaluate different scheduling policies at the edge server for incoming computing jobs in vision applications. We show that a greedy scheduling policy is optimal for a class of AoI-related penalty functions.

Chapter 2

Compressing Timely Updates

2.1 Streaming Data Compression and Recent Results

Over the past few years, substantial progress has been made on analyzing and optimizing the age in various computing and networking systems. The improvement of the age performance relies mostly on more efficient design of the transmission scheme, such as optimizing the source updating rate, and scheduling the updates with the freshest information as mentioned before. In these works, the information updates are usually assumed to be identically distributed packet lengths and the service times are independent of the content contained in the update. Very limited effort has been made to characterize the information contained in a status update.

In many real-time updating systems, such as live video surveillance and information updating in vehicular networks, the updates generated by the source observers reflect the changes of the system in the real world, which can usually be described by a collection of system states. Thus, efficient compression of the update information based on the probabilistic model is desired in order to fit those source messages into the constrained physical channel and reduce the transmission time. Shannon entropy characterizes the fundamental limits on the amount of information required to represent a source message based on the probability distribution of the source symbols [39]. The process of translation between original source messages and encoded bit sequence is known as *source coding*. Most of the traditional lossless source coding schemes have been focused on minimizing the length of the encoded sequence in order to approach the Shannon entropy of the source [40–43]. In a communication system, this is equivalent to minimizing the average channel usage for transmitting the source messages. The difference between

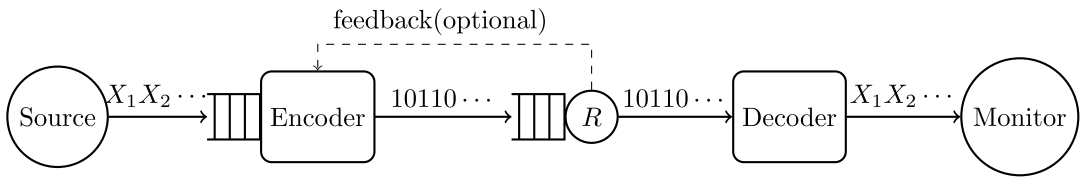


Figure 2.1: System diagram for streaming lossless source coding.

average codeword length and the Shannon entropy is usually called the source coding *redundancy*. If we instead require the source coding system to reconstruct the messages in a timely way, there are two new fundamental problems we need to address:

1. Is timely compression a different problem from traditional data compression that minimizes the redundancy?
2. If the answer to the previous question is NO, what is the optimal lossless source coding for timeliness?

Here we consider the lossless streaming source coding system illustrated in Figure 2.1. Every update generated by the source is transmitted to the receiver through a binary channel with a fixed rate, and the receiver is required to reconstruct the entire message generated by the source in a timely fashion. The encoder and decoder pair has to agree on a specific coding scheme that converts the source messages to bit sequences and vice versa. In Section 2.2, the system with deterministic symbol arrivals is examined, and we show that maintaining timeliness is fundamentally different from minimizing the redundancy. Intuitively, the optimal coding scheme should fully utilize the channel resource and reconstruct the source sequence as timely as possible. That means minimizing the coding redundancy is usually suboptimal since the transmission delay in a constrained channel depends on higher order moments of the encoded sequence length. An alternative system with randomly arriving source symbols is examined in Section 2.4. In this system, we need to carefully define the timeliness metric of the system, since the source stays idle occasionally and whether the idleness i informative or not depends on the application scenario. The age analysis in this case involves the evaluation of the queueing delay of the service facility with random vacations.

The design of streaming source coding has appeared in different contexts. The early work of end-to-end delay study of source coding can be traced back to the analysis of compression for communication concentrators [44]. Randomly arriving source symbols are encoded into d-ary codewords and fed into a buffer that outputs one codeword letter per time unit. It was shown that analyzing the average delay is difficult except for the special case where the buffer size is infinite and the interarrival time of symbols is exponentially distributed. The analysis for a finite buffer is extended in [45]. A variant of the Huffman code was proposed to minimize the probability of buffer overflow, which is shown to depend on the Laplace transform of the random codeword length. The bounds on the probability of buffer overflow originated from the G/G/1 queue waiting time upper bound in [46] and the lower bound in [47]. Similar queueing analysis on the source coding system was conducted by [48], and later inspired the development of the delay optimal prefix code in [49]. Given that the average delay is a function of the first and second moments of the codelength, the algorithm in [49] converts the codebook design problem to a coin collector's problem and constructs the optimal codebook using an alternative nodeset method. In [50], this nodeset method is generalized to the construction of length-limited Huffman codes. The design of source coding scheme for more general penalties was extended in [51]. In [52], the construction of an optimal code based on the nodeset notation is further optimized with smaller time complexity and applied to a larger class of quasi-linear penalty functions of the codelength. While most of the delay analysis are for block coding schemes, the bounded expected delay in sequential arithmetic coding is derived in [53].

Apart from the delay analysis of the streaming source coding system, there are other significant works on the error-exponent analysis in delayed-decoding system. In [54], the authors examined the system with deterministic symbol arrival in which the source symbols arrive at the encoder sequentially one per time unit, and the receiver is required to reconstruct the source with a fixed end-to-end delay constraint. It was shown that the exponent of the decoding error probability is a function of the decoding delay. The error exponent analysis was later applied to correlated sources using random binning in [55]. In [56], the moderate deviations constants and the bounds on the error exponents were derived for a slightly different streaming system that allows the encoder to know a certain number of future symbols at the time of encoding. The delayed sequential coding for correlated video frames was studied in [57].

Although the problem of streaming source coding has received a lot of attention over the years, only a few papers look at the system from an AoI perspective. It was shown that the differential encoding scheme improves the average age when the channel is unreliable and the feedback is available at the source. In [58], a streaming source coding system with source blocking was examined. Each source symbol represents a timely update message sent by the source, but the symbols that arrive at the encoder while the channel is busy are skipped. An optimal Shannon code was proposed to minimize the average age of the freshest source symbol at the receiver.

The rest of this chapter covers our works in [59–61]. In [59], we examine the average AoI of a system with deterministic source symbol arrivals and lossless fixed-to-variable coding schemes. Different from the lossy update model in [58], here we requires the receiver to decode the entire source sequence. We observed that the encoder must choose an appropriate source blocklength N to balance data compression delays against network congestion deriving from insufficient compression. In [60], similar age analysis was extended to a backlog-adaptive source coding model that makes the busy/idle state at the channel interface available at the source encoder. This enables the encoder to adjust the source blocklength based on the state of the channel and improves the age performance under dynamic channel conditions. The system with random source symbol arrivals was investigated in [61], in which the idleness at the source requires some special encoding mechanism. Here we extend the results in [61] and show that the age analysis is converted to the delay analysis in Geo/G/1 queueing system with service vacations.

2.1.1 Notation

• **Definition: (Probability Generating Function)** For a non-negative discrete random variable X with CDF $F_X(x)$, the probability generating function (PGF) is defined as

$$\hat{X}(z) = \mathrm{E}\big[z^X\big] = \sum_{x=0}^{\infty} z^x \Pr[X = x]. \tag{2.1}$$

The k-th order derivative is

$$\hat{X}^{(k)}(z) = \frac{\mathrm{d}^k}{\mathrm{d}^k z}\hat{X}(z). \tag{2.2}$$

Thus the first-order derivative is simply $\hat{X}^{(1)}(z) = \mathrm{E}\big[Xz^{X-1}\big]$, and the expected value $\mathrm{E}[X]$ and second moment can be obtained by

$$\mathrm{E}[X] = \lim_{z\to 1} \hat{X}^{(1)}(z) \tag{2.3}$$

$$\mathrm{E}\big[X^2\big] = \lim_{z\to 1} \hat{X}^{(2)}(z) + \hat{X}^{(1)}(z). \tag{2.4}$$

2.2 Deterministic Symbol Arrivals Without Feedback

2.2.1 AoI for Fixed-to-Variable Codes

In this section, we focus on the source coding system with deterministic symbol arrivals and without channel feedback. As shown in Figure 2.1, in each time slot (starting from $t = 1$), the source generates a discrete i.i.d. symbol X_i from a finite alphabet $\mathcal{X}$ with PMF $P_X(x)$. The encoder and decoder pair chooses a fixed-to-variable coding scheme, which means the encoder has to wait for the arrival of N symbols at the encoder buffer, and then group them into a single block and maps entire blocks into variable-length bit strings. The k-th symbol block is B_k such that $B_{k+1} = X_{kN+1}X_{kN+2}\cdots X_{(k+1)N}$. The encoded sequence is then fed into the first-in-first-out (FIFO) buffer as shown in Figure 2.1, which outputs one binary bit to the decoder through the channel at each $1/R$ time unit. Because a source block may be encoded into a codeword that is shorter than the block length, it is possible that FIFO buffers becomes empty. Since the encoder knows exactly when the FIFO buffer becomes empty, the encoder outputs a random

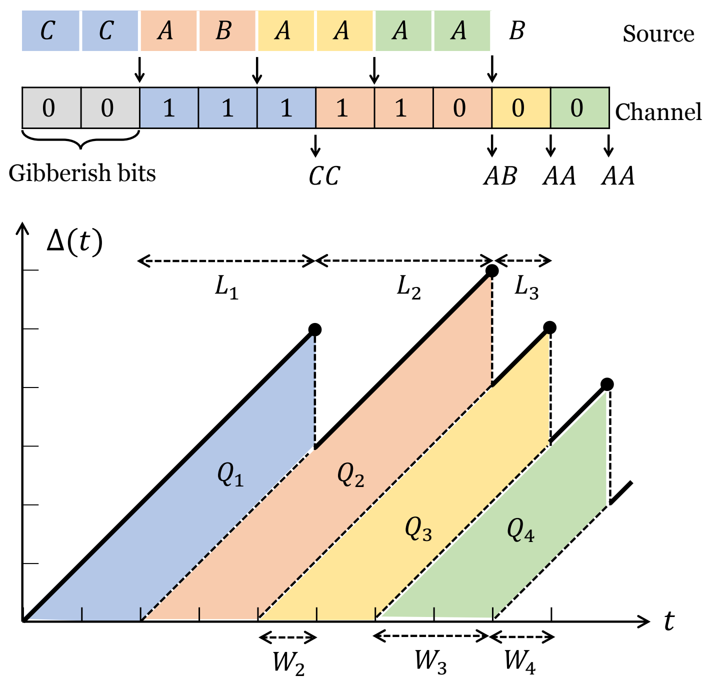


Figure 2.2: Example of streaming source coding system with deterministic symbol arrivals with three source symbols and blocklength $N = 2$.

gibberish bit $\phi \in \{0, 1\}$ independent of any codewords. In fixed-to-variable length coding, the decoder is able to determine whether the next received bit is a gibberish bit or not, since the generation time of next symbol block is known to the decoder [54].

When the decoder receives the prefix-free codeword, it reconstructs the corresponding message. Note that all the message symbols contained in a single message block are decoded at the same time and thus have the same delivery time. The delivery time of block B_k is denoted by D_k The instantaneous age at time t is then reset to the age of the last symbol in the most recently decoded block, since all symbols in the same block are decoded simultaneously. Let $N(t)$ denote the number of symbols observed by the encoder by time t. At every time t, the decoder reconstructs the source sequence up to $X_1, \cdots, X_{N(u(t))}$, where $u(t) < t$ is the time stamp of the most recent decoded information symbol. We note that $u(t)$ is advanced to a new time index only if a new symbol is decoded. The instantaneous age of the information stream $X^{(\epsilon)}_{N(u(t))}$ at the receiver at time t is then given by $\Delta(t) = t - u(t)$. That is, if the most recently decoded symbol at time t is X_i, which was produced by the source at time i, the instantaneous status age is $\Delta(t) = t - i$.

Figure 2.2 depicts an example of the age process $\Delta(t)$ at the receiver with ternary source symbols A, B, C, blocklength $N = 2$ and channel rate $R = 1$. Before the first block arrives, the encoder randomly generates two gibberish bits 00 that are sent to the channel. Since the decoder knows those bits are sent before when the first block arrives, it ignores those gibberish bits and reconstructs nothing. The first two symbols CC are grouped as the first block B_1, and then encoded into a long codeword 111. Thus, the codeword 110 corresponding to the second block AB has to be queued at the buffer first and sent after the transmission completion of the previous block. We observe that the blocklength N is the inter-arrival time between any two blocks at the input of encoder. The age is a sawtooth function that increases linearly in time in the absence of any symbol blocks and is reset to $D_k - kN$ at time D_k when symbol block B_k is decoded at the receiver.

By evaluating Figure 2.2, the average age is the average area under sawtooth diagram, and the integration of the sawtooth area is equivalent to the sum of disjoint polygon areas Q_k. The average AoI of the coding system observed by the receiver is given by

$$\begin{aligned}\Delta &= \lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \int_0^{\mathcal{T}} \Delta(t)\,\mathrm{d}t \\ &= \lim_{K\to\infty} \frac{1}{NK} \sum_{k=1}^{K} Q_k, \end{aligned} \tag{2.5}$$

where the area of the disjoint polygon is

$$Q_k = \frac{[D_k - (k-1)N]^2}{2} - \frac{[D_k - kN]^2}{2}. \tag{2.6}$$

Substituting (2.6) back to (2.5) gives

$$\begin{aligned}\Delta &= \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} (D_k - kN) + \frac{N}{2} \\ &= \mathrm{E}[D_k - kN] + \frac{N}{2}. \end{aligned} \tag{2.7}$$

From a queueing perspective, we can view a block B_k as a customer arriving at time

kN and departing at time D_k. Then $D_k - kN$ is exactly the system response time of the customer. Since $D_k \geq kN$, the average age is always lower bounded by $N/2$. The interarrival times of customers are deterministic since the interarrival time of symbol blocks is exactly the product of the block length and the symbol interarrival time N. As a result, the term $\mathrm{E}[D_k - kN]$ in (2.7) is the expected system time that block B_k spends in the queue. Let $\mathrm{E}[T] \triangleq \mathrm{E}[D_k - kN]$, for an arbitrary customer k when the queue has reached steady-state. The average age is then given by

$$\Delta = \mathrm{E}[T] + \frac{N}{2}. \tag{2.8}$$

Alternatively, we denote Δ_k as the k-th peak value of the age process $\Delta(t)$. The average peak AoI at the receiver is then defined as

$$\Delta^P = \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \Delta_k. \tag{2.9}$$

Evaluating Figure 2.2, the k-th peak age is given by $\Delta_k = D_k - (k-1)N$. Since the system response time $T_k = D_k - kN$, the average peak age is

$$\Delta^P = \mathrm{E}[T] + N. \tag{2.10}$$

Given a blocklength N, we observe that both the average age and the average peak age simply have an additional constant term (N or $N/2$) other than the average system response time $\mathrm{E}[T]$ of the queueing system. Since the interarrival time Y_k between information symbols is deterministic and the service time in the system follows a general distribution depending on the length of the codeword, the entire streaming source coding system can be modeled as a discrete-time D/G/1 queueing system. In streaming block coding, each encoded bit takes $1/R$ time unit to be transmitted by the FIFO buffer, thus the service time of the symbol block B_k with corresponding binary code length L_k is exactly $S_k = L_k/R$. To maintain a stable queueing system, it is required that the arrival rate is strictly less than the service rate, i.e. $1/\mathrm{E}[Y] < 1/\mathrm{E}[S]$ and we have the following claim and definition of *valid* blocklength

Lemma 2.2.1. *Given a block length N, the streaming source coding system is stable if and only if* $\mathrm{E}[L] < NR$.

We also have the following definition based on the stability of the system.

• **Definition:** A blocklength is valid if and only if there exist a codebook with average codelength $\mathrm{E}[L] < NR$ for $X \in \mathcal{X}$.

In a stable queueing system, the system time $T = W + S$ is the sum of service time S and waiting time W. That is, the average system time is $\mathrm{E}[T] = \mathrm{E}[S] + \mathrm{E}[W]$. We note that the interarrival time before k-th customer is $Y_k = N$ and the service times S_k are also discrete random variable, the average waiting time for discrete-time D/G/1 queue can be obtained according to [62]

$$\mathrm{E}[W] = \frac{\mathrm{E}\big[S^2\big] - \mathrm{E}[S] - (N^2 - N)}{2(N - \mathrm{E}[S])} + \sum_{r=1}^{N-1} \frac{1}{1 - z_r}, \tag{2.11}$$

where z_r are the unique roots of the equation

$$z^N - \hat{S}(z) = 0, \tag{2.12}$$

that are on or within the unit circle but not equal to 1. Note that solving the above equation and obtain (2.11) is somewhat complicated, we instead apply bounds for G/G/1 queue in [63] such that the average waiting time is upper bounded and lower bounded by

$$\mathrm{E}[W] \le \frac{\mathrm{E}[S^2] - \mathrm{E}^2[S]}{2(N - \mathrm{E}[S])}, \tag{2.13}$$

$$\mathrm{E}[W] \ge \frac{\mathrm{E}[S^2] - \mathrm{E}^2[S]}{2(N - \mathrm{E}[S])} - \frac{\mathrm{E}[S]}{2}. \tag{2.14}$$

It's interesting to point out that the upper bound in (2.13) and the lower bound in (2.14) only differ by the half of the average service time. The upper bound in (2.13) is the average waiting time for D/D/1 queue, while the lower bound in (2.14) is the average waiting time for M/G/1 queue [63]. As a result, we have the following theorems.

Theorem 2.2.2. *The average AoI is*

$$\Delta = \frac{\mathrm{E}\big[L^2\big] - \mathrm{E}[L] - N^2R^2 + NR}{2R(NR - \mathrm{E}[L])} + \sum_{r=1}^{N-1} \frac{1}{(1-z_r)R} + \frac{\mathrm{E}[L]}{R} + \frac{N}{2}, \tag{2.15}$$

where z_r are the unique roots of the equation

$$z^N - \hat{L}(z) = 0 \tag{2.16}$$

that are on or within the unit circle but not equal to 1.

We note the roots z_r occur in complex conjugate pairs. Thus, $\sum_{r=1}^{N-1} 1/(1-z_r)$ is real.

Theorem 2.2.3. *The average AoI is upper and lower bounded by*

$$\Delta \le \bar{\Delta} = \frac{\mathrm{E}[L^2] - \mathrm{E}^2[L]}{2R(NR - \mathrm{E}[L])} + \frac{\mathrm{E}[L]}{R} + \frac{N}{2} \tag{2.17}$$

$$\Delta \ge \underline{\Delta} = \frac{\mathrm{E}[L^2] - \mathrm{E}^2[L]}{2R(NR - \mathrm{E}[L])} + \frac{\mathrm{E}[L]}{2R} + \frac{N}{2}. \tag{2.18}$$

We observe that both the exact expression of average age in Theorem 2.2.2 and the bounds in Theorem 2.2.3 depend on the source block length N. Furthermore, we also note that the distribution of the codeword length $P_L(l)$ depends on the block coding scheme that is determined by the blocklength N. One objective is then to design the AoI-optimal fixed-to-variable codebook with codeword length $P_L(l)$ that minimizes the average age in Theorem 2.2.3. We realize this would be rather complicated since it involves the optimization over the PGF of codelength L. Instead, we can use the upper bound in Theorem 2.2.3 as an approximate of age average and optimize over the first and second moments of the codeword length $\mathrm{E}[L]$ and $\mathrm{E}\big[L^2\big]$ for a given N. This problem can be divided into two subproblems:

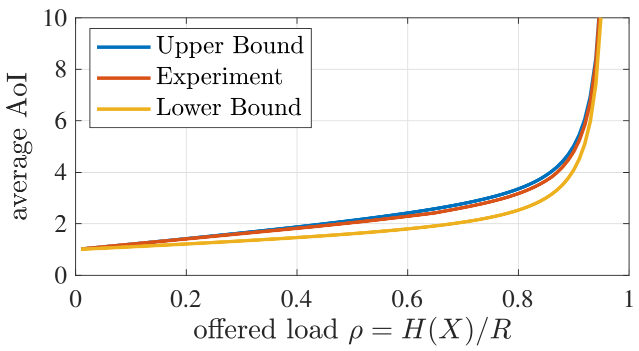
1. For a fixed blocklength N, what is the AoI-optimal fixed-to-variable code?

2. What is the AoI-optimal blocklength N?

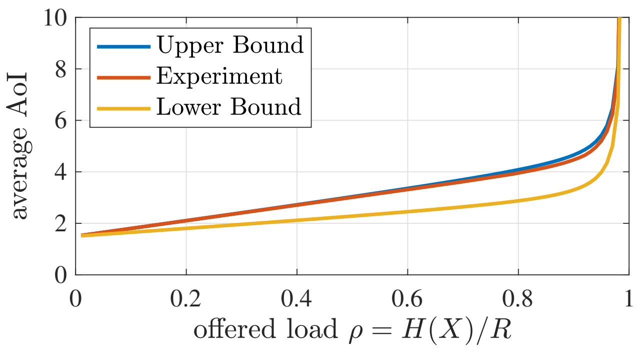
In section 2.2.2, we will discuss the recursive algorithm to search for the age optimal source coding scheme by using the upper bound in Theorem 2.2.3 as the objective function. This search algorithm, originally introduced in [49], is also applicable for a wide range of penalty functions of the codeword length. For the second subproblem, we will instead evaluate numerically and demonstrate the behaviors of varying the blocklength.

Here we also want to emphasize that this is different from traditional source coding schemes that focus on minimizing the average information bit rate in order to approach the Shannon entropy $H(X)$ of the source [39]. The difference between the *compression rate*, which is average codeword length per symbol $\mathrm{E}[L]/N$, and the entropy $H(X)$ is usually called the source coding *redundancy*. It is well known that the redundancy can be minimized and eliminated as the blocklength N grows to infinity. However, we demonstrate in Theorem 2.2.2 and 2.2.3 that the compression rate is not the only factor in the AoI metric. For a given blocklength N and channel rate R , the traditional purpose of minimizing the average codeword length is equivalent to minimizing the average channel usage for transmitting the source. Instead, the minimization of AoI requires the encoder/decoder pair to fully utilize the channel resource and reconstruct the source sequence as timely as possible. In particular, minimizing the coding redundancy is usually suboptimal since the transmission and queueing delays in a constrained channel depend on higher order moments of the codeword length.

Figure 2.3 demonstrates the comparison between the experimental ISA and the upper and lower bounds in Theorem 2.2.3. Here we also define the offered load ρ as the ratio between source entropy $H(X)$ and the channel R. In this experiment we choose two sources with different probability P as shown in 2.3a and 2.3b, and vary the channel rate R. The source symbols are encoded using a Huffman code with blocklength $N = 2$ and $N = 3$, respectively. In both example, we observe that the upper bound is very tight to the experiment results. Since the upper bound and lower bound differ by $\mathrm{E}[L]/2R$, the gap between two bounds becomes larger as R decreases and the offered load ρ approaches to 1. When R is large compared to $H(X)$, the average age grows almost linearly with the blocklength N.



(a) source probability $P = \{0.6, 0.3, 0.1\}$ and $N = 2$.



(b) source probability $P = \{0.4, 0.3, 0.2, 0.1\}$ and $N = 3$.

Figure 2.3: Comparing experimental average AoI, the upper and lower bounds for deterministic symbol arrivals using Huffman codes.

2.2.2 Age-optimal Codes

In this section, we use the AoI metric as a penalty function and obtain a lossless fixed-to-variable coding scheme that minimizes the penalty. Since the average age expression as a function of the distribution of the codelength $P_L(l)$ is rather complicated in Theorem 2.2.2, the problem becomes rather intractable. Instead, if the penalty is a function of the first and second moments of the codelength, $\mathrm{E}[L]$ and $\mathrm{E}\big[L^2\big]$, we can use the recursive searching technique from [49] to obtain the optimal coding scheme that minimizes the penalty. The procedure of obtaining the optimal code is described as follows.

Here we define an encoder to be *monotonic* if for any pair of symbol probabilities $P_X(x_1) \geq P_X(x_2)$, the corresponding codeword length $L_X(x_1) \geq L_X(x_1)$. For any monotonic fixed-to-variable codebook encoder ε, we plot the codebook in the (x, y) plane as a point $(x, y) = \big(\mathrm{E}[L](\varepsilon), \mathrm{E}\big[L^2\big](\varepsilon)\big)$ as shown in Figure 2.4. Here we define $\mathcal{E}$ as the set of all monotonic encoders $\mathcal{E} = \{\varepsilon_1, \varepsilon_2, \ldots\}$. Note that the set $\mathcal{E}$ is finite for a

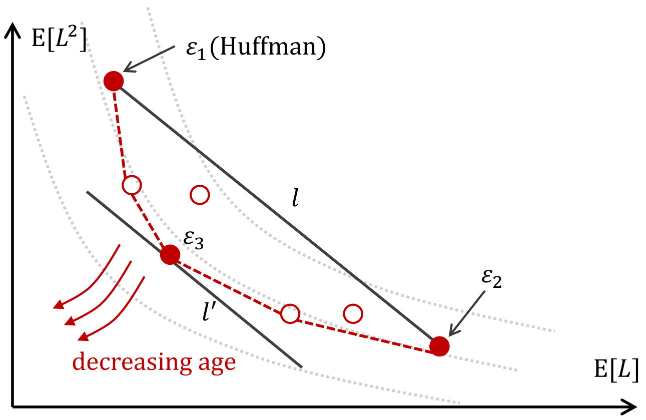


Figure 2.4: The illustration of convex hull algorithm and the representation of codebooks in the coordinate.

finite size source alphabet.

We recall that a set $C \in \mathbb{R}^2$ is convex if the line segment connnecting any two points of C is still within C. For any set $\mathcal{E}$ in $\mathbb{R}^2$, the convex hull of $\mathcal{E}$ is then defined as the smallest convex set containing $\mathcal{E}$. Specifically, we denote ε_1 as the point that minimizes $\mathrm{E}[L]$, which is lying at the left most end of the convex hull, and ε_2 as the point that minimizes $\mathrm{E}\left[L^2\right]$ and lying at the bottom of the convex hull. We note that ε_1 corresponds to a Huffman code since it's the optimal fixed-to-variable code to minimize the average codeword length $\mathrm{E}[L]$. We then denote $B(\mathcal{E})$ as the lower left boundary of the convex hull, which is a polygonal arc connecting ε_1 and ε_2 as shown in Figure 2.4.

For average AoI, we use the upper bound in Theorem 2.2.3 as the penalty function. Denoting $(x, y) = \left(\mathrm{E}[L], \mathrm{E}\left[L^2\right]\right)$, the upper bound in (2.13) becomes the penalty function

$$f_1(x, y) = \frac{y - x^2}{2R(NR - x)} + \frac{x}{R} + \frac{N}{2}, \tag{2.19}$$

where $0 \leq x \leq NR$ according to Lemma 2.2.1. For a fixed $z = f(x, y)$, the parabolic curve $y(x)$, which is shown in Figure 2.4, represents an equal penalty z contour. If the contour curve $y(x)$ has negative first derivative $y'(x) < 0$ and positive second derivative $y''(x) > 0$ in the feasible range of x, then we can show that the optimal codebook ε^* lies within the lower left boundary of the convex hull $B(\mathcal{E})$. If we draw a line through the optimal codebook ε^* tangent to the contour curve $y(x)$, all other codebooks are no lower than the tangent line. In our case, we prove that both (2.19) and (2.88) satisfy $y'(x) < 0$ and $y''(x) > 0$ in the appendix. The goal is then to search all the codebooks lying on $B(\mathcal{E})$ and extract the optimal ε^* for a given penalty function f.

The search of codebooks on $B(\mathcal{E})$ can be done by a recursive linear programming algorithm. First we define a linear function

$$g(L) = \alpha\,\mathrm{E}[L] + \beta\,\mathrm{E}[L^2], \tag{2.20}$$

with two weighting parameters $\alpha, \beta \in [0, 1]$. We note that (2.20) represents any tangent line with slope $-\alpha/\beta$ in Figure 2.4. It was shown in [52] that the optimal code that minimizes a linear penalty function in (2.20) can be obtained by converting the problem to the coin collector's problem, which was firstly introduced to describe a similar problem in length-limited Huffman code in [50]. The coin collector's problem can be efficiently solved by a package-merge algorithm in $O(|\mathcal{X}|^3)$ time in [49] and $O(|\mathcal{X}|^2)$ time in [52]. A brief introduction to the algorithm is in the appendix on page 67.

We denote the subroutine that gets the codebook that minimizes (2.20) as MinLinear(). The codebook returned by MinLinear() is one of the codebooks ε lying at the boundary $B(\mathcal{E})$. Given that MinLinear() can be solved efficiently for any pair of parameters α and β, we can obtain all the possible codebooks on $B(\mathcal{E})$ by varying α and β recursively.

The recursive search algorithm is outlined in Algorithm 1. We start from two extreme cases: finding the optimal codebooks $\varepsilon_1, \varepsilon_2$ by calling the subroutine MinLinear() with penalty function g with two sets of parameters $(\alpha, \beta) = (\alpha_1, \beta_1) = (1, 0)$ and $(\alpha, \beta) = (\alpha_2, \beta_2) = (0, 1)$. (α_1, β_1) corresponds to a penalty function that returns a prefix code ε_1 that minimizes the average code length $\mathrm{E}[L]$, which is exactly the Huffman code. One the other hand, (α_2, β_2) corresponds to a penalty function that returns a prefix code ε_2 that minimizes the second moment of the codeword length $\mathrm{E}\big[L^2\big]$. Then the algorithm recursively searches available codebooks between two points ε_1 and ε_2 by calling a procedure Span$(\varepsilon_1, \varepsilon_2)$. In the Span procedure, the value of α and β are updated in steps 10 and 11, respectively. The subroutine MinLinear() is used again

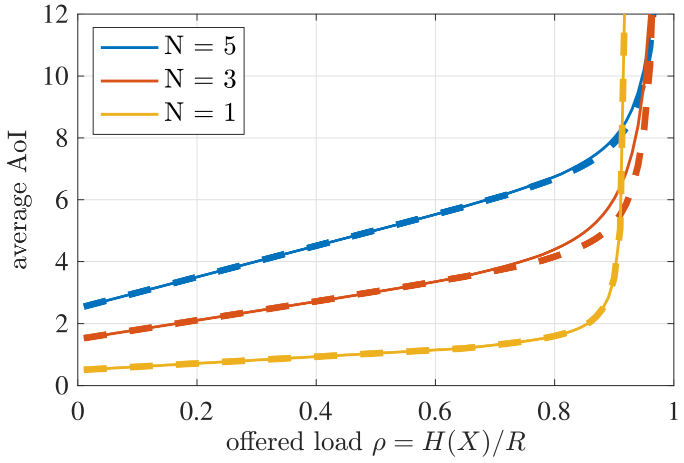
Algorithm 1 Convex Hull Boundary Search Algorithm

```
Require: MinLinear($\alpha$, $\beta$)
 1: $S \leftarrow$ empty
 2: $\varepsilon_1 \leftarrow$ MinLinear(1, 0)
 3: $\varepsilon_2 \leftarrow$ MinLinear(0, 1)
 4: if $\varepsilon_1 = \varepsilon_2$ then
 5:     return $\varepsilon_1$
 6: else
 7:     $S \leftarrow \{\varepsilon_1 \ \& \ \varepsilon_2 \ \& \ \text{Span}(\varepsilon_1, \varepsilon_2)\}$
 8:     return $\varepsilon^* = \arg\min_{\varepsilon \in S} f(L(\varepsilon))$
 9: procedure Span($\varepsilon_1, \varepsilon_2$)
10:     $\alpha \leftarrow \mathrm{E}[L^2](\varepsilon_1) - \mathrm{E}[L^2](\varepsilon_2)$
11:     $\beta \leftarrow \mathrm{E}[L](\varepsilon_2) - \mathrm{E}[L](\varepsilon_1)$
12:     $\varepsilon_3 \leftarrow$ MinLinear($\alpha, \beta$)
13:     if $\varepsilon_3 = \varepsilon_1$ or $\varepsilon_3 = \varepsilon_2$ then
14:         return empty
15:     else
16:         return $\varepsilon_3$ & Span($\varepsilon_1, \varepsilon_3$) & Span($\varepsilon_2, \varepsilon_3$)
```

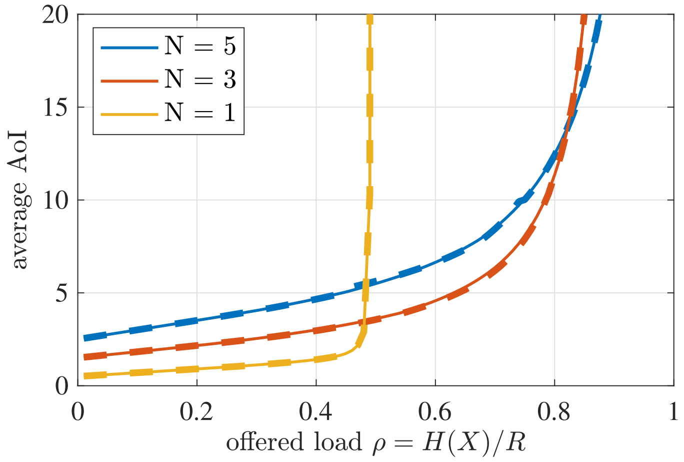
to obtain the optimal codebook for the new pairs of α and β. Graphically, this step is equivalent to drawing a line segment l that connects the points corresponding to ε_1 and ε_2, and then searching for the lowest line l' parallel to l that touches the boundary $B(\mathcal{E})$. If l' lies below l, then a new codebook ε_3 is contained in the line l'. Otherwise, the recursion stops if we couldn't find a new codebook ε_3. The procedure then repeats for the new pair of codebooks $(\varepsilon_1, \varepsilon_3)$ and $(\varepsilon_2, \varepsilon_3)$ until all the codebooks lying on boundary $B(\mathcal{E})$ are found. After collecting all the codebooks, we compare all of them and select the one that minimizes the penalty function f.

2.2.3 Numerical Evaluation

Figure 2.5 demonstrates the average AoI as a function of the channel rate R for different blocklength N. In Figure 2.5a, we use the three symbols A, B, C with $P(A) = 0.6, P(B) = 0.3, P(C) = 0.1$, and vary the channel rate R above $H(X)$ to vary the offered load ρ. The thin solid line marks the average AoI using Huffman codes, while the thick dashed line marks that using our Age-optimal codes. When R is large and ρ is small, encoding with large blocklength is a losing proposition, since what we gain by reducing the output rate of the encoder is forfeited because the delay of the system is



(a) source probability $P = \{0.6, 0.3, 0.1\}$.



(b) source probability $P = \{0.9, 0.08, 0.02\}$.

Figure 2.5: Comparing the average AoI with Huffman codes (thin solid lines) and Age-optimal codes (dashed thick lines) with different blocklength N.

dominated by long interarrival times of large blocks. The arrival process at the FIFO buffer is bursty with long codewords that arrive infrequently. Hence, the optimal strategy in the high FIFO rate region is to choose the smallest $N = 1$. On the other hand, as R decreases and the offered load ρ approaches to the stability limit given a blocklength N, the average AoI curve quickly blows up. This sharp transition effect occurs earlier for smaller N since the corresponding average code length is larger. Since the redundancy of block coding decays with the blocklength N, the threshold of the transition approaches $\rho = 1$ as N increases. In this region of transition, it is complicated to obtain the optimal blocklength analytically, since the minimum average AoI given a blocklength N depends on the specific codebook design, which is determined by the source probabilities. But we observe that choosing the smallest possible valid blocklength N mostly gives the minimum average AoI. For example, at $\rho \approx 0.95$, we observe that $N = 1$ is no longer valid, so we need to select a large N to obtain smaller compression rate $\mathrm{E}[L]/N$, and the next valid N in our example is $N = 3$.

Figure 2.5a also demonstrates the difference between the compression-optimal Huffman codes, and our age-optimal code. When ρ is small, the age-optimal codes obtained by our search algorithm are identical to the Huffman codes. This is because the average AoI is mainly dominated by the average codeword length when R is small as shown in Theorem 2.2.3. As ρ becomes large, the age-optimal codes start to outperform the Huffman codes for $N = 3$, due to the fact that the queueing delay that comes from higher moments of the codelength starts to play an important role in the average AoI. However, we also note that age-optimal codes are still close to the Huffman code for other values of N. This is due to the fact that the variance of codelength is limited by the tree structure of prefix-free codes. An age-optimal codebook can alleviate the penalty arises from the queueing delay, but it won't significantly reduce the penalty.

Figure 2.5b demonstrates another example with three symbols A, B, C with $P(A) = 0.9, P(B) = 0.08, P(C) = 0.02$. In this case, the entropy of the source is lower compared to the previous example in Figure 2.5a. We have similar observations in this example, except that the system becomes unstable in a faster way for all three different N. The age-optimal codes are mostly identical to the Huffman codes. We suspect this is due to the huge difference among the source probabilities. The optimization algorithm has very little freedom in terms of choosing the codelength for source symbols. Since the average codelength is a dominating factor in the age minimization, the encoder has to assign short codewords to symbols with large probabilities, and long codewords to symbols with small probabilities. We also suspect that the optimization is more likely to have significant performance gain when the probabilities are somewhere between equally likely and very steep.

2.3 Deterministic Symbol Arrivals With Feedback

For fixed-to-variable coding schemes, generating a new codeword when the channel is busy is a losing proposition since it injects additional workload and waiting time to the system. It was shown in [5, 10] that the average AoI can be reduced by controlling the update generation time given the state of the channel. In this section, we borrow the idea from [5, 10] and consider a similar streaming source coding system with deterministic arrivals and channel idle/busy feedback as illustrated in Figure 2.1. The busy/idle state at the channel interface is available at the source encoder. The source encoder buffers arriving symbols and submits a new codeword only when the channel is idle. When there is a backlog of b symbols, the encoder chooses the corresponding fixed-to-variable codebook with a blocklength of at most b. In this scheme, the channel state feedback allows the encoder to avoid subjecting codewords to unnecessary waiting at the channel interface and to also mitigate the traffic load by compressing longer source blocks when necessary. As opposed to traditional fixed-to-variable coding that maintains constant blocklength for the entire source sequence, backlog-adaptive block coding is a variable-to-variable scheme in which the methodology of fixed-to-variable coding is preserved but the blocklength varies according to encoder backlog. This scheme is feasible in practice because of the synchronization between the encoder and the decoder as described below.

The streaming source coding system with deterministic arrivals and channel idle/busy feedback as illustrated in Figure 2.1 Starting at time $t = 1$, discrete memoryless source symbols X_t arrive at each time unit sequentially. We assume that the encoder and decoder agree on an initial source block length b, and they are synchronized in the sense that both of them know the number of symbols queued at the encoder input buffer. The encoder will wait until time n and then use a prefix-free fixed-to-variable code with block length n for the input sequence X^n. The encoded sequence is then fed into a first-in-first-out (FIFO) channel buffer, which outputs one binary bit to the decoder every $1/R$ seconds. Since the codewords are prefix-free, the decoder collects received bits and reconstructs the message block immediately when it sees a codeword. Since the decoder knows when the entire codeword is received and when the last transmitted

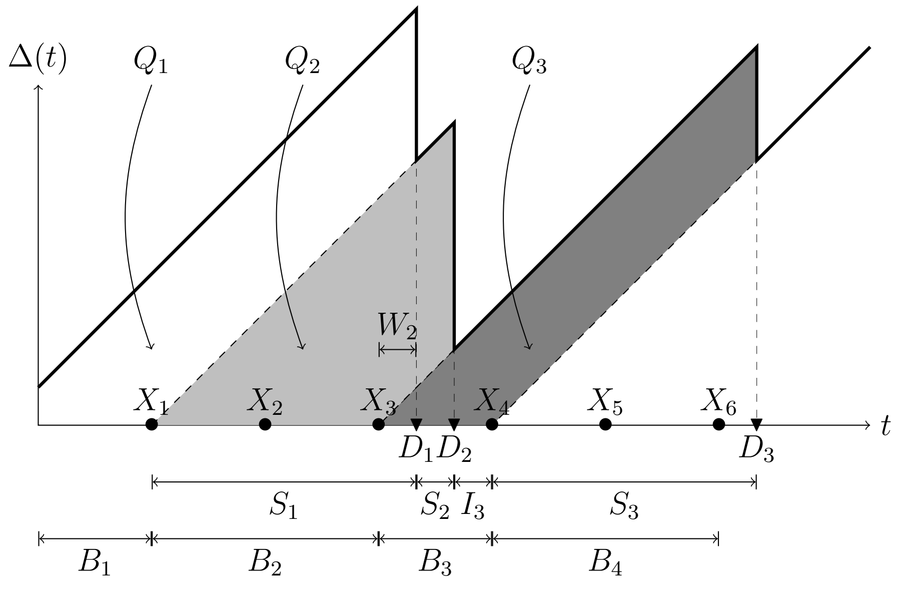


Figure 2.6: Sample path of the age $\Delta(t)$ (the upper envelope in bold) for backlog-adaptive streaming source coding with the just-in-time (JIT) policy.

symbol of that codeword was sent, the decoder also knows there are u input symbols queued at the encoder input buffer at that instant. Having agreeing on a pre-determined function $n' = f(b, n)$, both the encoder and decoder know that blocklength $f(b, n)$ will be used to send the next block.

Figure 2.6 shows a sample path of the status age at the decoder. Every time the encoder sees an idle channel, it stops collecting symbols and immediately submits codeword for all b symbols in the encoder backlog, i.e., $f(b, n) = b$. We refer to this scheme as the just-in-time (JIT) sequence parsing policy, which will be revisited later. Let's denote the k^{th} source symbol block as Y_k. This consists of N_k symbols starting from symbol X_i, i.e., $Y_k = X_i X_{i+1} \cdots X_{i+N_k-1}$. Suppose source block Y_k is encoded as a codeword with length $L_k(Y_k)$. Since the source blocks Y_k are i.i.d. given the blocklength N_k, the code length L_k depends on Y_k only through N_k and we rewrite $L_k(Y_k)$ as $L_k(N_k)$. The transmission time of the codeword through the channel is $S_k = L_k(N_k)/R$. The delivery time of the block Y_k is denoted by $D_k = k + S_k$. Note that all message symbols contained in a single message block are decoded at the same time and thus have the same delivery time. We also denote the encoder backlog at the time instance when the block Y_k is encoded as B_k; thus $B_k \geq N_k$.

In Figure 2.6, the initial blocklength is $N_1 = 1$ and the first source block is encoded as a long codeword which takes a large service time S_1 in the channel. When the entire codeword of the block Y_1 is delivered at time D_1, source symbol X_2 and X_3 are queued at the encoder buffer and ready for be encoded. Thus, in the just-in-time policy we have $N_2 = B_2 = 2$. The waiting time gap from when symbol X_3 is generated to when $Y_2 = X_2X_3$ is encoded, is denoted by W_2. We note that although this is a best effort scheme to alleviate the traffic, it doesn't eliminate the source symbol waiting time completely. Y_2 is then encoded into a short codeword and delivered at time D_2, which is earlier than the arrival time of the next symbol X_4. Since the channel is idle after D_2, the encoder generates a new codeword once the next symbol X_4 arrives, resulting in $N_3 = 1$. The channel idle time after D_2 is denoted by $I_3 = 4 - D_2$.

At the encoder, block Y_k incurs a waiting time W_k from the arrival of the first symbol of Y_k until the completion of the previous block, and then requires service time S_k for transmission. The channel idle time measures the time gap from when the previous block is decoded to when the next new symbol arrives. Denoting the system response time for the k^{th} block as $T_k = W_k + S_k$, the idle time is

$$I_k = \begin{cases} 0, & T_{k-1} \geq 1, \\ \lceil T_{k-1} \rceil - T_{k-1}, & \text{otherwise.} \end{cases} \tag{2.21}$$

The age is a sawtooth function that increases linearly in time in the absence of any decoded symbol blocks and is reset to $B_k - N_k + S_k + W_k$ at time D_k when symbol block Y_k is decoded at the receiver. Following the approach in Section 2.2, the integration of the sawtooth area is equivalent to the sum of disjoint polygonal areas Q_k shown in Figure 2.6. The average AoI is given by

$$\Delta = \lim_{N\to\infty} \frac{\sum_{k=1}^{N} Q_k}{\sum_{k=1}^{N} N_k} = \frac{\mathrm{E}[Q_k]}{\mathrm{E}[N_k]}. \tag{2.22}$$

The trapezoid area is given by

$$Q_k = \frac{1}{2}\left((B_k + W_k + S_k)^2 - (B_k - N_k + W_k + S_k)^2\right)$$

$$= N_k(B_k + W_k + S_k) - \frac{N_k^2}{2}. \tag{2.23}$$

If follows from (2.22) and (2.23) that the average AoI can be rewritten as

$$\Delta = \frac{\mathrm{E}[N_k W_k] + \mathrm{E}[N_k S_k] + \mathrm{E}[N_k B_k]}{\mathrm{E}[N_k]} - \frac{\mathrm{E}\left[N_k^2\right]}{2\,\mathrm{E}[N_k]}, \tag{2.24}$$

where

$$\mathrm{E}[N_k S_k] = \sum_{n=1}^{\infty} n\ \mathrm{E}[S_k | N_k = n]\, P_N(n). \tag{2.25}$$

For optimal fixed-to-variable prefix codes, the average service time given the source length $\mathrm{E}[S_k|N_k = n]$ can be bounded by

$$\frac{n\,H(X)}{R} \ \leq\ \mathrm{E}[S_k|N_k = n] \ \leq\ \frac{n\,H(X)+1}{R}. \tag{2.26}$$

Thus $\mathrm{E}[N_k S_k]$ is bounded by

$$\frac{H(X)\,\mathrm{E}\left[N^2\right]}{R} \leq \mathrm{E}[N_k S_k] \leq \frac{H(X)\,\mathrm{E}\left[N^2\right]}{R} + \frac{\mathrm{E}[N]}{R}. \tag{2.27}$$

For backlog-adaptive source coding, both the encoder backlog B_k and previous blocklength N_{k-1} are known to the decoder. Let's define the blocklength selection process as a function $N_k = f(B_k, N_{k-1})$. The encoder and decoder are synchronized during the compression process as long as the function f is determined beforehand and kept unchanged. Here we consider the simplest case where the encoder has no control of the blocklength but uses the entire backlog every time the channel becomes idle. That is, $N_k = B_k$, which we refer to as the passive blocklength adaptation scheme. For this scheme, we revisit the just-in-time parsing policy and analyze an alternative lazy parsing policy.

One significant limitation of the passive scheme is the potential persistence of the backlog. If the encoder occasionally outputs a very long codeword to the channel for the previous block, the encoder will then encode a large source block afterward. This may not be favorable since the corresponding codeword for a large source block is expected to be long, which may cause the decoder to wait for the long source phrase and the age accumulates and persists.

It Section 2.2, we show by experiment that there exists an optimal choice for the blocklength N given a channel rate R when fixed blocklength coding scheme is used. We are interested in the equivalent problem here, i.e., whether there exists an optimal strategy for the blocklength selection f. This leads us to the examination of a constrained maximum blocklength selection scheme in Section 2.3.2, in which we set a maximum threshold τ for the blocklength such that the encoder will only compress a length τ source phrase even if the backlog is large.

2.3.1 Passive Blocklength Adaptation

2.3.1.1 Lazy Parsing Policy

We first consider a source sequence parsing policy that allows the encoder to wait for an idle time after the delivery of every codeword. Every time the channel becomes available, the encoder holds the backlog until the next source symbol arrives, and then groups the new symbol with the backlog as the new source block to be encoded. We refer to this scheme as the lazy parsing policy. In this case, the k^{th} source blocklength N_k depends on the previous codeword length L_{k-1} by

$$N_k = B_k = \left\lceil \frac{L_{k-1}(N_{k-1})}{R} \right\rceil. \tag{2.28}$$

We denote $L(n) = L_k(N_k = n)$ as the length of codeword using the source blocklength $N_k = n$, and the conditional CDF of $L(n)$ is

$$G_{L|N}(l|n) = \Pr\left[L \leq l | N = n\right]. \tag{2.29}$$

Thus we can use a Markov chain to describe the blocklength process N_k. The transition probability from blocklength i to j is

$$p_{ij} = G_{L|N}(jR|i) - G_{L|N}((j-1)R|i), \quad i, j \in \mathbb{Z}^+. \tag{2.30}$$

Note that p_{ij} depends only on the distribution of the codelength L_k and the channel rate R. The distribution of code length $G_{L|N}$ for all N is known to both the encoder and decoder since the codebook is chosen in the beginning and fixed for the entire source sequence. Given a channel rate R, the steady-state probability, denoted by $\pi_n = P_N(n)$, can be obtained numerically based on (2.30). Then the average age is

$$\Delta = \frac{\mathrm{E}\left[N^2\right]}{2\,\mathrm{E}[N]} + \frac{\mathrm{E}[NS]}{\mathrm{E}[N]}, \tag{2.31}$$

where

$$\mathrm{E}[NS] = \frac{1}{R} \sum_{n=1}^{\infty} n \; \mathrm{E}[L(n)] \, \pi_n. \tag{2.32}$$

2.3.1.2 Just-in-time Parsing Policy

For just-in-time parsing, the encoder submits a new codeword right after the service of the previous block, and the k^{th} source block length N_k depends on the previous waiting time and service time by

$$N_k = B_k = \max\left(\left\lfloor W_{k-1} + \frac{L_{k-1}(N_{k-1})}{R}\right\rfloor, 1\right). \tag{2.33}$$

The average age (2.24) is written as

$$\Delta = \frac{\mathrm{E}[N_k W_k] + \mathrm{E}[N_k S_k]}{\mathrm{E}[N_k]} + \frac{\mathrm{E}\left[N_k^2\right]}{2\,\mathrm{E}[N_k]}. \tag{2.34}$$

We note that the waiting time W_k is a random variable which represents the difference between the delivery time of the most recent source block D_{k-1} and the largest integer that is smaller than D_{k-1}, i.e. $\lfloor D_{k-1} \rfloor$. Thus, $0 \leq W_k < 1$ but the distribution of

W_k is unknown. Altering W_k will affect the stationary distribution of the blocklength N_k, resulting in changes to the average age in (2.24) that are small but difficult to characterize.

Due to the complexity of analyzing W_k, here we consider an approximate model for the backlog process N_k, in which the waiting time W_k for each block in (2.33) is $W_k = 0$. We assume the arrival of every source block is now postponed to the time instance right after the completion of service for the previous block. In other words, this approximation artificially postpones the source block arriving time to ensure $W_k = 0$ for every k. Note that the interarrival time in this model is no longer the blocklength of the source but the service time of the previous block. The interarrival time is given by

$$Z_k = \max\left(S_{k-1}, 1\right) = \max\left(\frac{L_{k-1}(N_{k-1})}{R}, 1\right). \tag{2.35}$$

Thus the backlog process at the encoder, a.k.a. the blocklength process, is $B'_k = N'_k = \lfloor Z_k \rfloor$. And N'_k can be represented by a Markov chain, where the transition probability from state i to state j is

$$p'_{ij} = \begin{cases} G_{L|B}(2R|i), & j = 1, \\ G_{L|B}\left((j+1)R|i\right) - G_{L|B}(jR|i), & j > 1. \end{cases} \tag{2.36}$$

Similarly, we denote the steady-state probability by $\pi'_n = P'_N(n)$, and the average age in (2.24) becomes

$$\Delta' = \frac{\mathrm{E}\left[Z_k^2\right]}{2\,\mathrm{E}[Z_k]} + \frac{\mathrm{E}[Z_k S_k]}{\mathrm{E}[Z_k]}, \tag{2.37}$$

with $$\mathrm{E}[Z_k] = \sum_{n=1}^{\infty} \mathrm{E}[Z_k|N_k = n]\,\pi'_n, \tag{2.38}$$

$$\begin{aligned} \mathrm{E}[Z_k S_k] &= \mathrm{E}\left[Z_k\,\frac{L_k(N_k)}{R}\right] \\ &= \frac{1}{R}\sum_{n=1}^{\infty}\sum_{n'=1}^{\infty} \mathrm{E}[Z_k|n]\,\mathrm{E}\left[L(n')\right] p'_{nn'}\pi'_n. \end{aligned} \tag{2.39}$$

In (2.38) and (2.39), the conditional expected interarrival time is

$$\mathrm{E}[Z_k|N_k = n] = \frac{\mathrm{E}[L(n)]}{R} + \mathrm{E}\left[\left(1 - \frac{L(n)}{R}\right)^+\right], \tag{2.40}$$

where $(\cdot)^+ = \max(\cdot, 0)$. This approximation with postponed arrivals and zero waiting time is just an analysis technique instead of a practical system. Intuitively, it is expected to provide lower average age compared to that of the true model in (2.24). Every time a codeword is received at the decoder, the instantaneous age is reset to a smaller value than that for the JIT policy. However, the symbol arrival rate is no longer 1 symbol per time unit since once a symbol is postponed, the arrival times of all remaining source symbols are postponed. Thus the effective symbol arriving rate is

$$\lambda'_S = \mathrm{E}\left[\frac{B'_k}{Z_k}\right] = \sum_{n=1}^{\infty} \mathrm{E}\left[\frac{\lfloor Z_k(n) \rfloor}{Z_k(n)}\right]\pi'_n < 1. \tag{2.41}$$

2.3.2 Constrained Maximum Blocklength

While the previous section provided analysis for passive blocklength adaptation, we now consider a baseline control scheme with the maximum constraints on the blocklength. In this case, we have a Markov chain for the backlog B_k. We consider the analysis of the more tractable lazy parsing policy and show how the constraints on the source blocklength affect the Markov chain. Let τ denotes maximum blocklength constraint. For any backlog $B_k \geq \tau$, the length of the compressed source block will be $N_k = \tau$, i.e., the k-th blocklength is

$$N_k = \min(\tau, B_k). \tag{2.42}$$

The new encoder backlog after the delivery of block Y_k will be

$$B_{k+1} = (B_k - \tau)^+ + \left\lceil \frac{L_k(\tau)}{R} \right\rceil. \tag{2.43}$$

To describe the backlog process, we introduce the following notation

$$\begin{aligned} i_\tau^+ &= (i-\tau)^+ \\ \underline{i}_\tau &= \min(\tau, i), \end{aligned} \tag{2.44}$$

for any integer i. The transition probability from backlog $B_k = i$ to $B_{k+1} = j$ is

$$q_{ij} = G_{L|B}\left((j - i_\tau^+)R|\underline{i}_\tau\right) - G_{L|B}\left((j - i_\tau^+ - 1)R|\underline{i}_\tau\right). \tag{2.45}$$

The states from $i = 1$ to τ in this Markov chain remain the same as those in the lazy parsing scheme in (2.30), while all other states $i \geq \tau$ are duplicate states as the state τ. That is, setting a threshold τ is equivalent to pruning all the states beyond state τ and copying the state τ as $\tau+1, \tau+2, \ldots$. We denote the steady-state probability of backlog $B_k = b$ as

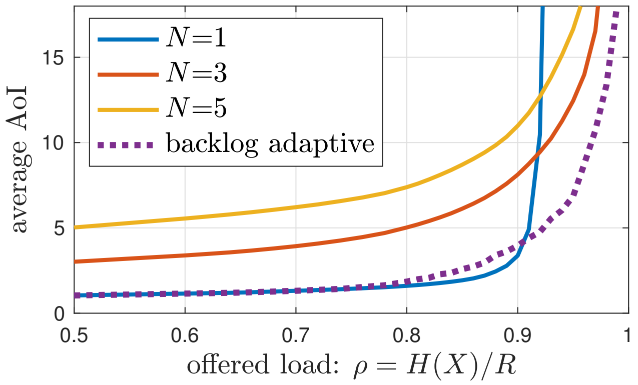
$$\pi_b = \lim_{k\to\infty} \Pr[B_k = b]. \tag{2.46}$$

The average age for constrained maximum blocklength (CMB) with lazy parsing policy is given by

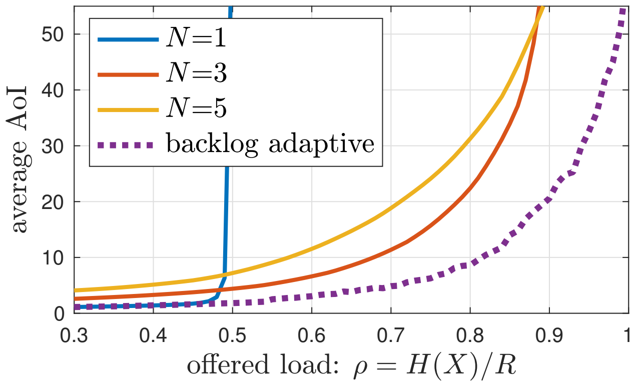
$$\Delta_\tau = \frac{\mathrm{E}[NB] + \mathrm{E}[NS]}{\mathrm{E}[N]} - \frac{\mathrm{E}\left[N^2\right]}{2\,\mathrm{E}[N]}, \tag{2.47}$$

where

$$\begin{aligned} \mathrm{E}[N] &= \sum_{b=1}^{\infty} \underline{b}_\tau \pi_b \\ \mathrm{E}[NB] &= \sum_{b=1}^{\infty} b\underline{b}_\tau \pi_b \\ \mathrm{E}[NS] &= \frac{1}{R}\sum_{b=1}^{\infty} \underline{b}_\tau L(b_\tau)\pi_b. \end{aligned} \tag{2.48}$$



(a) source probability $P = \{0.6, 0.3, 0.1\}$.



(b) source probability $P = \{0.9, 0.08, 0.02\}$.

Figure 2.7: Comparing backlog-adaptive source codes to fixed blocklength codes for two source models. For each source model, the entropy $H(X)$ is fixed but the channel rate R is varied.

2.3.3 Numerical Evaluations

Figure 2.7 compares the backlog-adaptive streaming source coding scheme to fixed blocklength N using Huffman codes. A ternary source with distribution $P_X(x_1) = 0.6, P_X(x_2) = 0.3, P_X(x_3) = 0.1$ is used in Figure 2.7a. With $H(X)$ fixed, the offered load $\rho = H(X)/R$ is controlled by varying the channel rate R. The backlog-adaptive coding scheme achieves the same minimum average age as $N = 1$, which implies that the blocklength adaptation mostly leads to the smallest blocklength $N = 1$. As the offered load ρ increases and approaches 1, the sharp transition effect occurs earlier for smaller N since the corresponding average information bits per symbol is larger. We observe that the backlog-adaptive blocklength scheme using just-in-time parsing

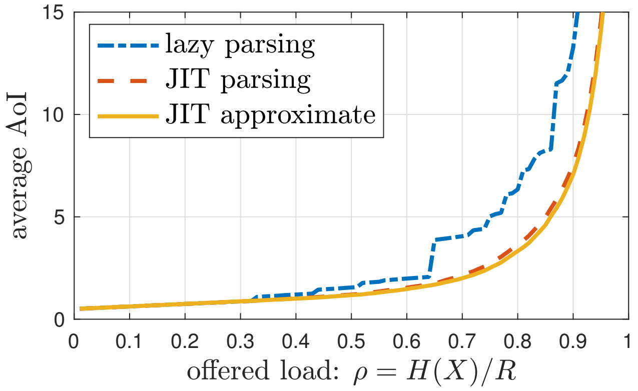


Figure 2.8: Comparing just-in-time (JIT) parsing policy, its approximation with postponed arrivals, and the lazy update policy using Shannon code.

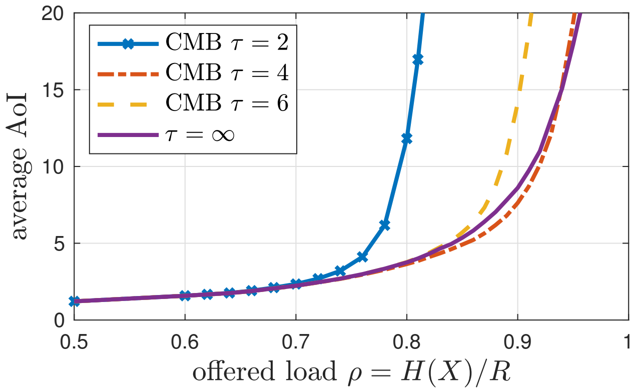
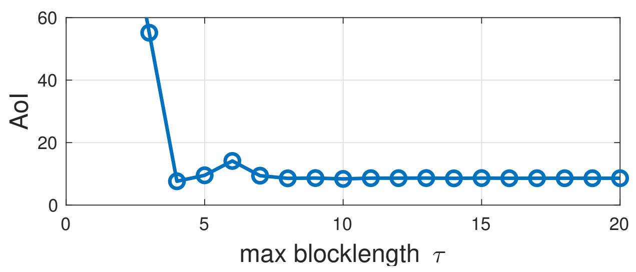


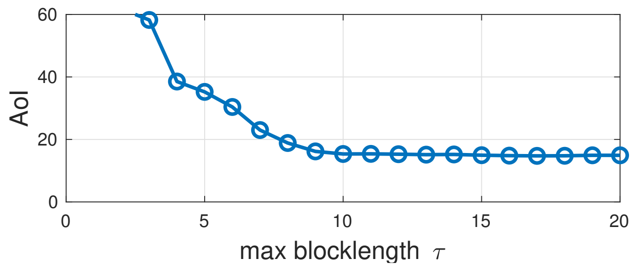
Figure 2.9: Example of average age for CMB with different τ using Shannon code.

outperforms any other fixed blocklength schemes in this example when the offered load ρ is high, i.e., $\rho > 0.9$. The requirement for large blocklength N under heavy load is automatically satisfied by the backlog-adaptive coding scheme, since a small channel rate R results in a large average blocklength $\mathrm{E}[N]$. We also observe similar behaviors of the backlog adaptive scheme in the experiment using another source distribution $P_X(x_1) = 0.9, P_X(x_2) = 0.08, P_X(x_3) = 0.02$ and the same source as shown in Figure 2.7b. We conclude that given a fixed channel rate R, the encoder backlog process provides the encoder the ability to adapt its blocklength to the optimal choice.

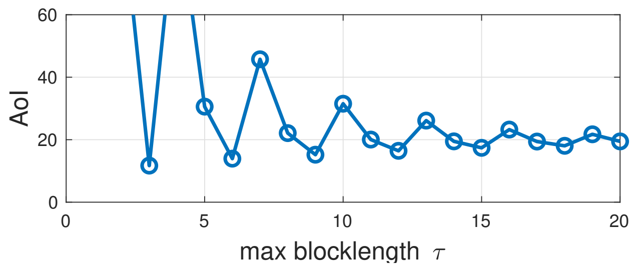
Figure 2.8 shows the comparison of average age among the just-in-time parsing policy, its approximation using postponed arrivals, and the lazy parsing policy. Here we use Shannon code to the same ternary source with $P_X(x_1) = 0.6, P_X(x_2) = 0.3, P_X(x_3) = 0.1$.



(a) $P(X) = (0.6, 0.3, 0.1)$.



(b) $P(X) = (0.75, 0.125, 0.125)$.



(c) $P(X) = (0.8, 0.1, 0.1)$.

Figure 2.10: Examples of average age for CMB as a function of τ using Shannon code. Different ternary source with offered load $\rho = H(X)/R = 0.9$.

In the low offered load ρ region, all three models achieve the same minimum average age. As ρ increases, the lazy policy leads to higher age than the JIT policy and blows up earlier. The approximation of JIT parsing with postponed arrivals provides very tight results to the actual JIT scheme in this example.

Figure 2.9 shows the average age with different maximum blocklength constraint τ in high traffic regime using the same source and backlog-adaptive Shannon code with JIT parsing policy. Note that the scheme with $\tau = \infty$ has no constraint on the maximum blocklength. We see that $\tau = 4$ mostly gives the lowest average age for a wide range of offered load, implying that constraining the maximum blocklength may possibly provide lower average age than passive blocklength adaptation given a channel rate R. However, this phenomenon depends on the source distribution and the coding scheme. Figure 2.10a depicts the average age as a function of τ at the offered load $\rho = H(X)/R = 0.9$. We observe that age varies at low ρ region and achieves the minimum age at $\tau = 4$. We suspect the reason is that the code length distribution under $N < \tau = 4$ is in favor of the average age, and thus allowing the source block to grow larger is unnecessary. The average age converges in experiment as τ increases, since the probability of getting a very large source block is very small given an offered load $\rho = 0.9$. Figure 2.10b and 2.10c demonstrate the average age for other ternary source distributions. We observe that the average age curve can be monotonic decreasing or following a certain converging pattern. This indicates that the existence of an age-optimal τ is not always guaranteed, and the average age given ρ depends on how the Markov chain in (2.45) evolves in a complex way as τ changes.

2.4 Random Symbol Arrivals

When the source symbols arrive randomly in time and the source mostly stays silent, the problem becomes more complicated due to different system requirements. One type of system requires the receiver to keep track of the state of the source at every time instance, including the case when the source doesn't generate any new symbols. We refer to this requirement as the reconstruction of the *information stream* including both the source symbols and the idleness between symbols. In this case, every idleness at the source at any time instance is also informative to the receiver, thus the encoder has to choose a coding scheme that contains both the idleness symbol and the source symbols. The other type of system cares about the source sequence consisting of the source symbols themselves but not the idleness of the source. In this case, the requirement for the receiver is the reconstruction of the *source sequence* only. Although only the source symbols are informative, the source still need to reserve a special codeword or flag to inform the receiver when there is no arriving source symbol. In both cases, the transmission of an idleness flag will affect the transmission of the source sequence, since it leads to additional overhead in the source symbol codeword.

Let's consider the streaming source coding system in Figure 2.1 again. In each time

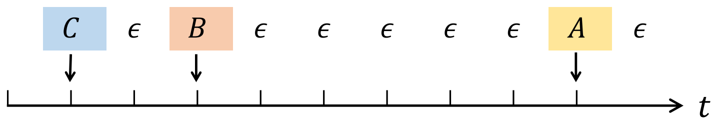


Figure 2.11: Example timeline of randomly arriving symbols generated by the source. The source symbols $X_1X_2X_3$ are shaded in color.

slot (starting from $t = 1$), the source is either idle or it generates a discrete i.i.d. symbol X_k from a finite alphabet $\mathcal{X}$ with PMF $P_X(x)$. The source is idle with probability $1-q$, and we use a special flag ϵ to indicate the idleness of the source at every symbol time instance. Let $N_X(t)$ denote the number of symbols observed by the encoder by time t. Figure 2.11 demonstrates an example timeline for the arriving symbols at the encoder input. The first three symbols $X_1 = C$, $X_2 = B$ and $X_3 = A$ arrive at times $t = 1, 3, 9$, respectively. The *source sequence* up to $t = 10$ is CBA and $N_X(t = 10) = 3$, but the *information stream* including the idle flag ϵ between any two source arrivals is actually $C\epsilon B\epsilon\epsilon\epsilon\epsilon\epsilon A\epsilon$.

The encoder then has to decide whether to 1) encode the source symbols only or 2) encode the information stream including all ϵ based on the timely information needed at the monitor. It is application dependent whether the the idleness between symbols is informative. If the monitor is an aggregator that collects and processes data generated by IoT devices, the objective of the encoder/decoder pair is more likely to be reconstructing the source sequence only. However, if the monitor would like to keep track of the source state at every time instance and reproduce the timing information about every source symbol, then the idleness ϵ is informative and needs to be encoded together with the source symbols X.

The encoded bit sequence is fed into the channel which is a bit pipe with a First-come-first-served (FCFS) buffer that outputs R bits per time unit. In this work, we limit our focus to the general case $R = 1$, meaning that the source arrival time interval matches the channel interval. Let U_k denote the arrival time of the source symbol X_k at the input of the encoder. Based on different application requirements, we have the following different age definitions and the corresponding coding objectives:

1. *Age of Information Stream (ISA):* The receiver needs to recover the entire information stream with all idle flags between source symbols. The age is defined as the age of the most recent received stream symbol at the decoder, and an idle flag ϵ is regarded as an informative update by the source.

2. *Age of Source Sequence (SSA):* the receiver needs to recover the source sequence $X_1X_2\dots X_k$ only, and the age is defined as the age of the most recent received source symbol at the decoder. The special idle flag ϵ is not regarded as an update by the source and need not to be encoded.

In case 1, we note that the idleness is as important as the source symbols. Thus, we can basically combine the idle flag symbol ϵ and original source alphabet $\mathcal{X}$ into alphabet $\mathcal{X}' = \{\epsilon\} \cup \mathcal{X}$. Furthermore, the new symbols arrive at the input of the encoder at every time instance and thus the arrival process is deterministic. In case 2, idle flags are not required to be reconstructed since they are not informative. However, the encoder has to send some special bit sequence to inform the decoder when there are no new arriving symbols to encode. One simple method is to let the encoder produce one single bit 0 and transmit it to the binary channel whenever the source is idle and the channel buffer is empty. In this case, all other codewords are added a prefix bit 1 to be distinguishable from the idleness code 0. Another method is to add ϵ to the symbols and construct an alternative codebook. We also note that this is not an issue if symbols arrive in a deterministic way. Since the decoder knows the deterministic arrival times of all symbols, the encoder can generates random gibberish bits when the channel buffer becomes idle so that the decoder can ignore those gibberish bits according to the symbol arrival time [64].

2.4.1 Age of Information Stream

We first focus on the analysis of information stream age including all the idleness symbol. Since the idleness symbol ϵ is now part of the alphabet that we need to encode, we

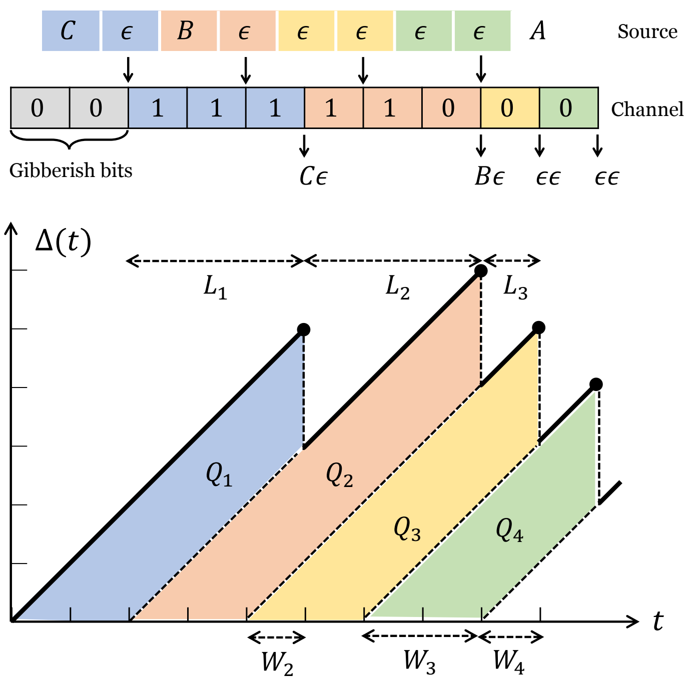


Figure 2.12: Example of the age of information stream (ISA) process with $|\mathcal{X}| = 3$ and blocklength $N = 2$.

construct the new source $X^{(\epsilon)} \in \mathcal{X}'$ with PMF

$$P_{X^{(\epsilon)}}(x^{(\epsilon)}) = \begin{cases} 1-q, & \text{if } x^{(\epsilon)} = \epsilon \\ q\,P_X(x), & \text{if } x^{(\epsilon)} \in \mathcal{X}. \end{cases} \tag{2.49}$$

The system behaves exactly the same as the system with deterministic symbol arrivals discussed and channel rate $R = 1$ in Section 2.2. Starting at time $t = 1$, discrete memoryless symbols $X^{(\epsilon)}$ arrive at every time unit sequentially, so the i-th information symbol $X_i^{(\epsilon)}$ arrives at time i.

Figure 2.12 shows a sample path of the age process at the receiver with ternary source symbols A, B, C and blocklength $N = 2$. Before the first block B_1 arrives, the encoder randomly generates two gibberish bits 00 and send it to the channel. Since the decoder knows those bits are sent before when the first block arrives, it ignores those gibberish bits and reconstructs nothing. The first symbol C and the next idle flag ϵ are grouped as the first block B_1, and then encoded into a long codeword 111. Thus, the codeword 100 corresponding to the second block $B_2 = B\epsilon$ has to be queued at the buffer first and sent after the transmission completion of the previous block B_1. Since the next two blocks are both $\epsilon\epsilon$, which is the most probable symbol[1], they are mapped to the shortest codeword 0 that takes only 1 time unit for transmission. We observe that the blocklength N is the inter-arrival time between any two blocks at the input of encoder. The age is a sawtooth function that increases linearly in time in the absence of any symbol blocks and is reset to $D_k - kN$ at time D_k when symbol block B_k is decoded at the receiver.

We note that each encoded bit takes one time unit to be transmitted by the FIFO buffer, thus the service time of the symbol block B_k with corresponding binary code length L_k is exactly the code length $S_k = L_k$. To maintain a stable queueing system, it is required that the arrival rate is strictly less than the service rate, i.e. $1/\operatorname{E}[Y] < 1/\operatorname{E}[S]$ and we have the following claim and definition of *valid* blocklength

Lemma 2.4.1. *Given a block length N, the streaming source coding system is stable if and only if* $\operatorname{E}[L] < N$.

• **Definition:** A blocklength is valid if and only if there exist a codebook with average codelength $\operatorname{E}[L] < N$ for $X^{(\epsilon)} \in \mathcal{X}'$.

We note that $N = 1$ is not a valid blocklength for any source X with more than one symbols. The encoder has to encode more than two alphabets including the idle flag ϵ, then at least one alphabet gets codeword with length more than 1 and thus $\operatorname{E}[L] > 1$.

Lemma 2.4.2. *For a symbol arrival probability q, there exists a valid blocklength N only if the original source $X \in \mathcal{X}$ has source entropy*

$$H(X) < \frac{1 - H_b(q)}{q}, \tag{2.50}$$

where $H_b(q)$ is the binary entropy function $H_b(q) = -q \log q - (1-q)\log(1-q)$.

Proof. Lemma 2.4.1 can be written as $\operatorname{E}[L]/N < 1$. Since $\operatorname{E}[L]/N > H(X^{(\epsilon)})$, it is

[1]if the source symbol arrival probability $q > 1/2$

required that $H(X^{(\epsilon)}) < 1$. The entropy of the new source is

$$\begin{aligned}H(X^{(\epsilon)}) &= -(1-q)\log(1-q) - \sum_{x\in X} qP_X(x)\log qP_X(x)\\ &= -(1-q)\log(1-q) - q\Big(\log q\sum_{x\in X}P_X(x) + \sum_{x\in X}P_X(x)\log P_X(x)\Big)\\ &= H_b(q) + qH(X).\end{aligned} \tag{2.51}$$

Thus, $H_b(q) + qH(X) < 1$ and rearranging the terms yields Lemma 2.4.2.

□

If the source entropy $H(X)$ and the source symbol arriving probability q satisfy Lemma 2.4.2, we have the following corollary based on Theorem 2.2.2 and Theorem 2.2.3.

Corollary 2.4.3. *The average age of information stream is*

$$\Delta_{IS} = \frac{\mathrm{E}\big[L^2\big] - \mathrm{E}[L] - N^2 + N}{2(N - \mathrm{E}[L])} + \sum_{r=1}^{N-1}\frac{1}{1-z_r} + \mathrm{E}[L] + \frac{N}{2}, \tag{2.52}$$

where z_r are the unique roots of the equation

$$z^N - \sum_{l=0}^{l_{\max}} P_L(l)z^l = 0 \tag{2.53}$$

that are on or within the unit circle but not equal to 1.

Corollary 2.4.4. *The average age of information stream is upper bounded by*

$$\Delta_{IS} \le \bar{\Delta}_{IS} = \frac{\mathrm{E}[L^2] - \mathrm{E}^2[L]}{2(N - \mathrm{E}[L])} + \mathrm{E}[L] + \frac{N}{2}. \tag{2.54}$$

For a given the blocklength N, we can use the the upper bound in Corollary 2.4.4 as a penalty function and search for the optimal codebook using Algorithm 1 in Section 2.2.2.

2.4.2 Age of Source Sequence

In this section, we focus on the age of the source sequence when the idleness ϵ is not informative. If the receiver is only interested in the source sequence $X_1, X_2, \ldots, X_n$ without any timing information, then the encoder does not have to encode every idle symbol ϵ between source symbols. In order to reconstruct every source symbol in a timely way, here we limit our analysis here to blocklength $N = 1$. However, since the source symbols arrive randomly, the channel buffer as shown in Figure 2.1 is sometimes empty but this is not allowed since the channel has to output either a bit 0 or 1 at every time slot. Hence, the encoder is required to send a special codeword, which differs from the codewords for the source symbols X_k, to the buffer when the buffer becomes empty. We note that this scheme is identical to encoding the idle flag ϵ as a symbol in the codebook, but we only apply the encoding and output the codeword $\varepsilon(\epsilon)$ when the channel buffer is empty. When the decoder sees the codeword $\varepsilon(\epsilon)$, it knows there is an idleness in the buffer and no new symbol arrives at the input of the encoder. The idle flag ϵ is then discarded by the decoder.

2.4.2.1 Prefix Encoding

We first focus on the scenario where the symbol arrival probability q is very small. The simplest way is to transmit a single bit "0" if the buffer is empty, and otherwise transmit a "1" followed by an encoded bit sequence for the source X. We refer to this scheme as *prefix idleness encoding.* In this scheme, the "0" bit occupies only one time slot and thus doesn't affect the next incoming source symbol. As a result, the length of every encoded sequence for source symbol X is increased by 1, because of the extra bit 1 as the header. Denote the original codelength for source symbol X as L_X, then the new length is then $L_{X_\epsilon} = L_X + 1$. For this baseline coding scheme, the system is stable if and only if $\mathrm{E}[L_X] + 1 < 1/q$. That is, for sources with entropy $H(X) \geq 1/q - 1$, there is no feasible source code for a stable system.

On the other hand, a source symbol X_k is declared at the output of decoder after the entire bit sequence corresponding to X_k is delivered to the decoder. At every time

Table 2.1: Example of codebook using prefix idleness encoding.

X_ϵ	ϵ	A	B	C
$\varepsilon(X_\epsilon)$	0	10	110	111

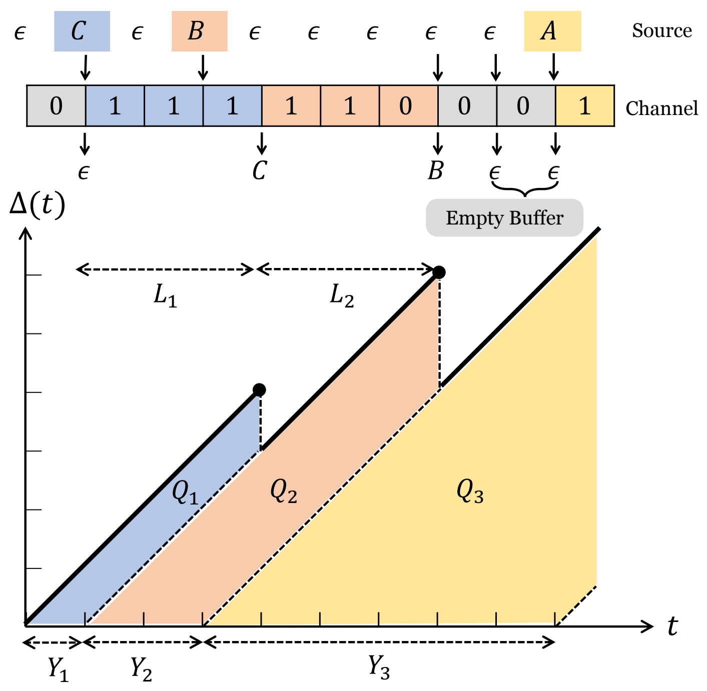


Figure 2.13: Example of instantaneous age of source sequence (SSA) at the receiver of streaming source coding with $|\mathcal{X}| = 3$. The special idle flag ϵ is encoded only if there is no new arriving symbol and the buffer becomes empty.

t, the decoder reconstructs the source sequence up to $X^{N(u(t))}$, where $u(t) < t$ is the time stamp of the most recent decoded source symbol. We note that $u(t)$ is advanced to a new time index only if a new symbol is decoded. The age of the source sequence $X^{N(u(t))}$ at the receiver at time t is then given by $\Delta(t) = t - u(t)$.

Figure 2.13 depicts an example of the FIFO buffer output process and the age process. Source symbols $X \in \{A, B, C\}$ arrive at the input of the encoder sequentially, and each symbol is encoded using the prefix-free codebook specified in Table 2.1. The first symbol $X_1 = C$ arrives at time $t = 1$, so the encoder output a 0 bit to inform the decoder that there is no arriving symbol at time $t = 0$. Right after source symbol C comes, the corresponding bit sequence 111 is fed into the FIFO buffer and output to the decoder one by one using $L_1 = 3$ time units. Thus, the age $\Delta(t)$ increases linearly from an initial value $\Delta_0 = 0$ and drops to $\Delta(4) = 3$ time units at time $t = 4$. The second symbol $X_2 = B$, which arrives at time $t = 3$, is deferred by one channel slot since the buffer is serving the codeword for the previous symbol C. Symbol B is then delivered to the decoder at time $t = 7$. The age is then reset to the waiting time plus the codeword transmission time for symbol B. Afterwards, no new source symbol arrives at times $t = 8$ and $t = 9$, so the encoder encodes two idle flags ϵ and send two 0's consecutively to the decoder. These two idle flags are then discarded by the decoder and the age $\Delta(t)$ increases linearly.

Similarly, $\Delta(t)$ is a random process that varies in time with the receiver's reconstruction of the source. We denote Y_n as the interarrival time between the $(n-1)$-th symbol and the n-th source symbol, and T_n as total time that symbol n spends in the system. Evaluating Figure 2.13 yields the average age of source sequence

$$\Delta_{SS} = \lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \int_0^{\mathcal{T}} \Delta(t)\,\mathrm{d}t = \lim_{N\to\infty} \frac{\sum_{n=1}^{N} Q_n}{\sum_{n=1}^{N} Y_n}, \tag{2.55}$$

where the polygonal area is

$$\begin{aligned} Q_n &= \frac{1}{2}\Big((Y_n + T_n)^2 - T_n^2\Big) \\ &= \frac{Y_n^2}{2} + Y_n T_n. \end{aligned} \tag{2.56}$$

For a stable system in which Y_n and T_n are i.i.d. for all integer n, the average age in (2.55) is

$$\begin{aligned} \Delta_{SS} &= \frac{\mathrm{E}[Q_n]}{\mathrm{E}[Y_n]} \\ &= \frac{\mathrm{E}[Y_n T_n]}{\mathrm{E}[Y_n]} + \frac{\mathrm{E}\big[Y^2\big]}{2\,\mathrm{E}[Y]} \\ &= \mathrm{E}[T] + \frac{\mathrm{E}\big[Y^2\big]}{2\,\mathrm{E}[Y]} + \frac{\mathrm{Cov}[Y_n, T_n]}{\mathrm{E}[Y]}. \end{aligned} \tag{2.57}$$

Here the interarrival time Y_n is geometrically distributed with Bernoulli success probability q. We also note that (2.57) applies to any distribution of Y_n and T_n.

Similarly, we denote Δ_k as the k-th peak value of the age process $\Delta(t)$. The average

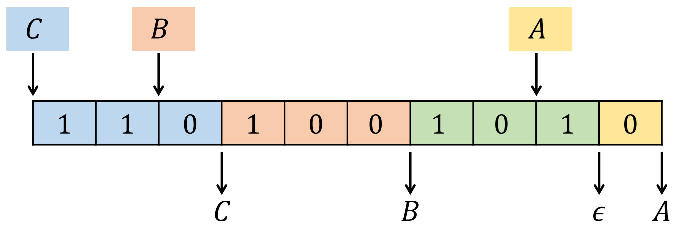


Figure 2.14: An example of the output process of the FIFO buffer in which the special codeword reserved for empty buffer signal ϵ is 101.

Table 2.2: Example of codebook using embedded idleness encoding.

X	A	B	ϵ	C	D
$\varepsilon(X)$	0	100	101	110	111

peak age at the receiver is then defined as

$$\Delta_{SS}^{P} = \lim_{K\to\infty} \frac{1}{K}\sum_{k=1}^{K}\Delta_k = \mathrm{E}[T] + \mathrm{E}[Y]. \tag{2.58}$$

We view every source symbol as a customer, and the service time of each symbol is then the number of time slots to transmit the binary codeword through the channel. T_n is then the total system response time for customer n. Since the customers arrive as a Bernoulli process, the key to calculate either AoI metric is the analysis of a queueing system with geometric interarrival time and general discrete-time service time that depends on the length of the codeword. We refer to this queueing model as a discrete-time Geo/G/1 queue.

2.4.2.2 Embedded Encoding

We note that the previous prefix encoding scheme is suitable for small system offered load ρ. However, when the system is mostly busy, the advantage of a short codeword for the idleness symbol ϵ will be forfeited since the buffer is overloaded by serving longer codewords for source messages. Therefore, it is necessary to have a more general idleness encoding scheme which includes ϵ in the codebook and assigns ϵ a longer codeword. In order to construct the new codebook, we assign the idleness symbol ϵ a probability p_ϵ. Figure 2.14 depicts an example of the FIFO buffer output process with the same arrival process as Figure 2.13. The corresponding codebook with the embedded idleness encoding is shown in Table 2.2. When the FIFO buffer becomes empty, the codeword 101 is transmitted to the decoder to indicate an idleness. In this case, the codeword corresponding to symbol A is deferred by 1 time slot since the buffer is busy sending the last bit 1 of the codeword corresponding to the idleness symbol ϵ.

If we treat every source symbol as a customer, the customer arrival process is identical to that of the prefix encoding scheme. The only difference here is the additional waiting time if the system is idle due to the transmission of idleness symbol ϵ. The transmission of ϵ can be viewed as the vacation of the service when the channel becomes idle, in which the service vacation length is exactly the length of the codeword for ϵ. Since ϵ will be encoded and sent again if there is no new symbols in the encoder buffer, it is equivalent to the case where the channel takes another vacation if the there is no customer waiting in the channel buffer. Thus, the AoI analysis in (2.57) and (2.58) still holds, and the corresponding target is the queueing analysis for Geo/G/1 queue with service vacation. We also note prefix encoding is a special case of the embedded encoding model with service vacation length being exactly 1.

2.4.2.3 Analysis for Geo/G/1 Vacation Queue

For geometric interarrival time Y with PMF $\Pr[Y = k] = (1-q)^{k-1}q$, the first and second moments are $\mathrm{E}[Y] = 1/q$ and $\mathrm{E}\big[Y^2\big] = (2-q)/q^2$. And the average system response time is the sum of average service time and average waiting time, i.e. $\mathrm{E}[T] = \mathrm{E}[S] + \mathrm{E}[W]$. Then the average age in (2.57) is rewritten as

$$\Delta = \mathrm{E}[W] + \mathrm{E}[S] + \frac{2-q}{2q} + q\,\mathrm{Cov}[Y_n, T_n]. \tag{2.59}$$

For queueing system with service vacations, we denote the original waiting time in Geo/G/1 queue as W_0 and the additional waiting time due to the service vacation as W_1. It was shown in [65, Theorem 3] that the stationary waiting time W, can be decomposed into the sum of two independent random variables $W = W_0 + W_1$. Therefore, the average waiting time is

$$\mathrm{E}[W] = \mathrm{E}[W_0] + \mathrm{E}[W_1]. \tag{2.60}$$

And the PGF of the waiting time is

$$\hat{W}(z) = \hat{W}_0(z)\hat{W}_1(z). \tag{2.61}$$

First we define the offered load as $\rho = q\,\mathrm{E}[S]$. For discrete-time Geo/G/1 queue, the PGF of the original waiting time $\hat{W}_0(z)$ and the average waiting time $\mathrm{E}[W_0]$ can be obtained from a distributional form of Little's Law and the PGF of the mean queue length as follows [66, 67]

$$\hat{W}_0(z) = \frac{(1-\rho)(1-z)}{(1-z) - q\left(1 - \hat{S}(z)\right)}, \tag{2.62}$$

$$\mathrm{E}[W_0] = \frac{\mathrm{E}\left[S^2\right] - \mathrm{E}[S]}{2(1/q - \mathrm{E}[S])}, \tag{2.63}$$

where $\hat{S}(z)$ is the PGF of the service time.

In our system model, the service facility, including the encoder and the bit pipe channel, goes into vacation for a fixed amount of time, which is denoted by V. At each vacation completion time instance, the service facility checks if there is any new customers, which is new symbols in our case in the encoder buffer. If there is any new customer, the service will be resumed immediately, meaning that the encoder converts a new symbol into bits and submits them to the bit pipe. Otherwise, the service facility takes another vacation with period V. The PGF of the additional waiting time $\hat{W}_1(z)$ due to service vacation is [65]

$$\hat{W}_1(z) = \frac{1 - z^V}{V(1-z)}. \tag{2.64}$$

Thus, the average waiting time $\mathrm{E}[W_1]$ is given by

$$\mathrm{E}[W_1] = \lim_{z \to 1} \frac{\mathrm{d}}{\mathrm{d}z} \hat{W}_1(z) = \frac{V-1}{2}. \tag{2.65}$$

We note that $\mathrm{E}[W_1] = 0$ when $V = 1$, meaning there will be no waiting due to vacation if the vacation takes exactly one time unit, which is identical to no vacation.

Substituting (2.63) and (2.65) back to (2.60), and substituting (2.62) and (2.64) to (2.61), we obtain the average composite waiting time and the corresponding PGF as follows

$$\mathrm{E}[W] = \frac{\mathrm{E}\big[S^2\big] - \mathrm{E}[S]}{2(1/q - \mathrm{E}[S])} + \frac{V-1}{2} \tag{2.66}$$

$$\hat{W}(z) = \frac{(1-\rho)(1-z^V)}{V\Big((1-z) - q\Big(1 - \hat{S}(z)\Big)\Big)}. \tag{2.67}$$

In [16], the covariance $\mathrm{Cov}[Y_n, T_n]$ is proven to be non-positive for G/G/1 queue, confirming the intuition that the interarrival time before customer n and the system time of customer n should be negatively correlated. We extend the result in [16, eqn. 86] and have the following lemma.

Lemma 2.4.5. *The covariance between the system time T_n and the interarrival time Y_n for Geo/G/1 queue is*

$$\mathrm{Cov}[Y_n, T_n] = -\frac{1-q}{q^2} + \frac{1-q}{q^2}\,\hat{T}(1-q) + \frac{1-q}{q}\,\hat{T}^{(1)}(1-q). \tag{2.68}$$

Proof of Lemma 2.4.5 is provided in the appendix on page 64. Here we note that $\hat{T}(1-q) = \mathrm{E}\big[(1-q)^T\big]$ and

$$(1-q)\,\hat{T}^{(1)}(1-q) = \mathrm{E}\big[T(1-q)^T\big]. \tag{2.69}$$

Since $1 - q \in [0, 1]$, $\mathrm{E}\big[T(1-q)^T\big] \le \mathrm{E}[T]\,\mathrm{E}\big[(1-q)^T\big]$ and we have the following upper bound on the covariance.

Lemma 2.4.6. *The covariance* $\mathrm{Cov}[Y_n, T_n]$ *for Geo/G/1 queue is upper bounded by*

$$\mathrm{Cov}[Y_n, T_n] \le -\frac{1-q}{q^2} + \left(\frac{1-q}{q^2} + \frac{1-q}{q}\,\mathrm{E}[T]\right)\hat{T}(1-q). \tag{2.70}$$

Since the system response time $T = S + W$, the PGF of system time $\hat{T}(z)$ is

$$\hat{T}(z) = \hat{S}(z)\hat{W}(z). \tag{2.71}$$

And similarly the first order differentiation $\hat{T}^{(1)}(z)$ can be obtained by

$$\begin{aligned}\hat{T}^{(1)}(z) &= \frac{\mathrm{d}}{\mathrm{d}z}\hat{T}(z)\\ &= \frac{\mathrm{d}\hat{S}(z)}{\mathrm{d}z}\hat{W}(z) + \frac{\mathrm{d}\hat{W}(z)}{\mathrm{d}z}\hat{S}(z)\\ &= \hat{S}^{(1)}(z)\hat{W}(z) + \hat{W}^{(1)}(z)\hat{S}(z).\end{aligned} \tag{2.72}$$

To keep it simple, we denote

$$\bar{q} = 1 - q. \tag{2.73}$$

Let $z = 1 - q$ in (2.67), we have

$$\hat{W}(\bar{q}) = \frac{(1-\rho)\,(1-\bar{q}^V)}{qV\hat{S}(\bar{q})}, \tag{2.74}$$

$$\hat{T}(\bar{q}) = \hat{W}(\bar{q})\hat{S}(\bar{q}) = \frac{(1-\rho)\,(1-\bar{q}^V)}{qV}. \tag{2.75}$$

For $V = 1$, we note that $\hat{T}(\bar{q})|_{V=1} = 1 - \rho$, indicating that $\hat{T}(\bar{q})$ is exactly the expected idle time of the queue if there is no vacation. Since the offered load is given by

$$\rho = q\,\mathrm{E}[S], \tag{2.76}$$

the first-order derivatives are

$$\begin{aligned}\hat{W}^{(1)}(\bar{q}) &= \frac{\mathrm{d}\hat{W}(z)}{\mathrm{d}z}\,|_{z=\bar{q}}\\ &= \frac{-(1-q\,\mathrm{E}[S])\left(1-\bar{q}^V\right)\left(qS^{(1)}(\bar{q})-1\right)}{V\left(q\hat{S}(\bar{q})\right)^2} - \frac{-(1-q\,\mathrm{E}[S])\bar{q}^V}{q\hat{S}(\bar{q})},\end{aligned} \tag{2.77}$$

$$\hat{T}^{(1)}(\bar{q}) = \hat{S}^{(1)}(\bar{q})\hat{W}(\bar{q}) + \hat{W}^{(1)}(\bar{q})\hat{S}(\bar{q})$$

$$= \frac{1 - q\,\mathrm{E}[S]}{q}\left(\frac{1-\bar{q}^{V}}{Vq\hat{S}(\bar{q})} - \bar{q}^{V-1}\right) \tag{2.78}$$

Substituting (2.75) and (2.78) back to Lemma 2.4.5, and then putting Lemma 2.4.5 and (2.66) back to (2.59) gives the following main theorem.

Theorem 2.4.7. *The average AoI for discrete-time Geo/G/1 queue with deterministic service vacations is given by*

$$\begin{aligned}\Delta = {} & \frac{\mathrm{E}\big[S^2\big] - \mathrm{E}[S]}{2(1/q - \mathrm{E}[S])} + \mathrm{E}[S] + \frac{V}{2} \\ & + \frac{\bar{q}(1 - q\,\mathrm{E}[S])\left(1-\bar{q}^{V}\right)}{Vq^2}\left(1 + \frac{1}{\hat{S}(\bar{q})}\right) - \frac{\bar{q}(1-q\,\mathrm{E}[S])}{q}.\end{aligned} \tag{2.79}$$

For queueing system without service vacation, i.e. $V = 1$, we have the follows

Theorem 2.4.8. *The average AoI for discrete-time Geo/G/1 queue without vacation is given by*

$$\Delta = \frac{\mathrm{E}\big[S^2\big] - \mathrm{E}[S]}{2(1/q - \mathrm{E}[S])} + \mathrm{E}[S] + \frac{1}{2} + \frac{\bar{q}(1-q\,\mathrm{E}[S])}{q^2\hat{S}(\bar{q})}. \tag{2.80}$$

Here we remark that the calculation of average age requires the complete knowledge of the service distribution $F_S(x)$. We can instead bound the average age using Lemma 2.4.6 and (2.75) to have the following results

Corollary 2.4.9. *The average AoI for discrete-time Geo/G/1 queue without vacation is upper bounded by*

$$\Delta \leq \mathrm{E}[T] + \frac{2-q}{2q} + \bar{q}\Big(\mathrm{E}[T]\big(1 - q\,\mathrm{E}[S]\big) - \mathrm{E}[S]\Big), \tag{2.81}$$

where the average system time is

$$\mathrm{E}[T] = \frac{\mathrm{E}\big[S^2\big] - \mathrm{E}[S]}{2(1/q - \mathrm{E}[S])} + \mathrm{E}[S]. \tag{2.82}$$

The bound in Corollary 2.4.9 depends only on the first and second moments of the service distribution S. Alternatively, we can bound the average age by using the general bounds on the covariance term $\mathrm{Cov}[Y_n, T_n]$ in [16, Lemma 18].

Corollary 2.4.10 (Alternative Bounds from G/G/1 Queue)**.** *The average age for discrete-time Geo/G/1 queue without vacation is bounded by*

$$\Delta \leq \frac{\mathrm{E}\left[S^2\right] - \mathrm{E}[S]}{2(1/q - \mathrm{E}[S])} + \mathrm{E}[S] + \frac{2-q}{2q}, \tag{2.83}$$

$$\Delta \geq \left(\frac{\mathrm{E}\left[S^2\right] - \mathrm{E}[S]}{2(1/q - \mathrm{E}[S])} + \mathrm{E}[S] \right) \bar{q}^{1/q} + \frac{2-q}{2q}. \tag{2.84}$$

Note that the upper bound (2.83) is obtained by setting $\mathrm{Cov}[Y_n, T_n] = 0$, i.e. removing the correlation between Y_n and T_n. We will see that the upper bound in (2.83) is surprisingly tight for most cases. Furthermore, the upper bound in (2.81) is very close to (2.83) in some numerical results.

Theorem 2.4.11. *The average peak AoI for discrete-time Geo/G/1 queue with deterministic service vacations is given by*

$$\begin{aligned} \Delta^P &= \mathrm{E}[T] + \mathrm{E}[Y] \\ &= \frac{\mathrm{E}\left[S^2\right] - \mathrm{E}[S]}{2(1/q - \mathrm{E}[S])} + \frac{V-1}{2} + \mathrm{E}[S] + \frac{1}{q}. \end{aligned} \tag{2.85}$$

2.4.2.4 Optimizing Prefix Encoding

In the prefix encoding scheme, the idle flag ϵ is encoded as a single bit 0. Thus, the vacation period is $V = 1$ and the service facility is available at every time instance. Let's assume the encoder and decoder choose a codebook for the source symbols X beforehand. Since every source symbol is now encoded into a sequence with prefix 1, the service time of a source symbol with index n is

$$S_n = L_X(X_n) + 1, \tag{2.86}$$

where $L_X(\cdot)$ is the length of the original codeword without the extra prefix 1. Substituting (2.86) back to Theorem 2.4.8 gives the following results

Theorem 2.4.12. *The average age of the source sequence using prefix idleness encoding*

Algorithm 2 AoI-aware Embedded Encoding

```
Require: SEARCH$(P, f)$, $P_X$, $f$
1: $\varepsilon \leftarrow$ SEARCH$(P_X,\ f)$
2: $p_\epsilon = 1 - q\,\mathrm{E}[L(\varepsilon)]$
3: for $x$ in $X$ do
4:     $P_X(x) \leftarrow (1 - p_\epsilon)P_X(x)$
5: return SEARCH$(P_X \cap p_\epsilon,\ f)$
```

is given by

$$\Delta_{SS,P} = \frac{\mathrm{E}\big[L^2\big] + \mathrm{E}[L]}{2(1/q - \mathrm{E}[L] - 1)} + \mathrm{E}[L] + \frac{3}{2} - \frac{q\bar{q}\,\mathrm{E}[L]}{q^2\bar{q}\hat{L}(\bar{q})}, \tag{2.87}$$

where we denote $L = L_X$ and $\hat{L}(\cdot)$ is the PGF of the codelength L_X.

In order to obtain the optimal codebook with prefix encoding scheme for SSA, we can reuse the convex hull search algorithm in Section 2.2.2 with the upper bound in (2.83) as the penalty function. Since the service time $S_n = L_n + 1$ in SSA, rewriting $x = \mathrm{E}[L_n]$ and $y = \mathrm{E}\big[L_n^2\big]$ in (2.83) yields

$$f_2(x, y) = \frac{y + x}{2(1/q - x - 1)} + x + 1 + \frac{2 - q}{2q}, \tag{2.88}$$

where $0 \leq x \leq 1/q - 1$ to maintain a stable system. We prove in the appendix that (2.88) is convex and thus the convex searching algorithm is applicable.

2.4.2.5 Optimizing Embedded Encoding

For any codebook containing the idleness symbol ϵ, the service time of a source symbol is exactly the length of the codeword for the symbol, i.e. $S_n = L_X(X_n)$. And the service vacation length is the codelength of the idleness symbol, $V = L_\epsilon$. Rewriting Theorem 2.4.8, we have the following theorem.

Theorem 2.4.13. *The average age of the source sequence using embedded encoding scheme is*

$$\Delta_{SS,E} = \frac{\mathrm{E}\big[L^2\big] - \mathrm{E}[L]}{2(1/q - \mathrm{E}[L])} + \mathrm{E}[L] + \frac{L_\epsilon}{2}$$

$$+\frac{\bar{q}(1-q\,\mathrm{E}[L])\left(1-\bar{q}^{L_\epsilon}\right)}{L_\epsilon q^2}\left(1+\frac{1}{\hat{L}(\bar{q})}\right)-\frac{\bar{q}(1-q\,\mathrm{E}[L])}{q}, \tag{2.89}$$

where we denote $L = L_X$ and $\hat{L}(\cdot)$ is the PGF of the source symbol codelength L_X.

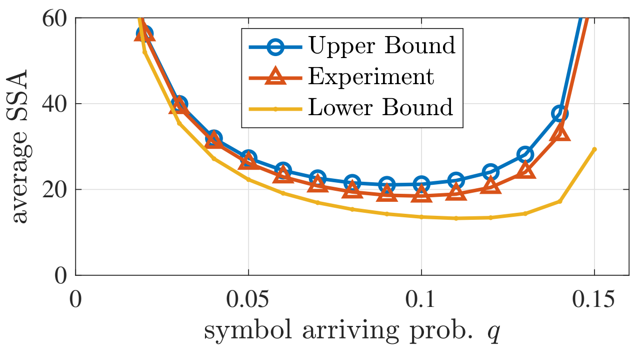
We note that Theorem 2.4.13 provides the average age for a given codebook, which encodes ϵ as a codeword with length L_ϵ and all other source symbols X as codewords with length L_X. Although we obtain the exact expression for the average age, it is not straightforward to obtain the age-optimal codebook design based on theorem 2.4.13, since the construction of a codebook determines the distribution of source codelength P_L as well as the idleness codelength L_ϵ.

Here we let I_ε to be the fraction in time that the buffer is empty when a codebook ε is used to compress the source, and p_ϵ to be the probability of idleness ϵ used to construct the codebook. For a given source probability P_X and convex penalty function $f(\mathrm{E}[L], \mathrm{E}\left[L^2\right])$, we can always exploits Algorithm 1 in Section 2.2.2 to obtain the optimal codebook design. However, the probability for idleness p_ϵ in this case is in fact unknown and subject to our choice. Intuitively, one should expect p_ϵ to be the actual frequency of sending the idleness symbol ϵ in the streaming source coding system, which is correlated to I_ε, the fraction of idle time. Hence, we first use the following heuristic to obtain the p_ϵ. First, we assume that the channel is not strictly binary. A special channel symbol ϕ will be sent whenever the encoder buffer is empty. In this case, the encoder and decoder pair doesn't need to include the idleness symbol ϵ in the codebook design, and thus the system behaves as a Geo/G/1 queue without vacation and the service time of each symbol X_n is exactly the length of the codeword $S_n = L_X(X_n)$. Here we can use the upper bound on the average age in (2.83) and have the following penalty function

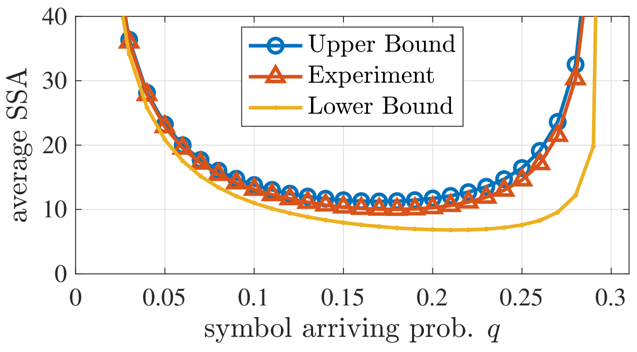
$$f_3(x, y) = \frac{y-x}{2(1/q-x)} + x + \frac{2-q}{2q}. \tag{2.90}$$

Alternatively, we can also use the exact expression of the average peak age in Theorem 2.4.11 with $V = 1$ as the penalty function

$$f_4(x, y) = \frac{y-x}{2(1/q-x)} + x + \frac{1}{q}. \tag{2.91}$$



(a) $s = 1$



(b) $s = 2$

Figure 2.15: Comparing experimental results and the bounds for SSA with Zipf source using prefix idleness encoding.

In this case, the AoI-optimal codes can be obtained by Algorithm 1, which is denoted as a subroutine Search() here. We note that Search() takes a set of probabilities P and a penalty function f as the inputs. Then the fraction of time for the buffer being empty I_ε, which is $1 - \rho = 1 - q\,\mathrm{E}[L]$ in this case, is then used as p_ϵ for for the embedded idleness encoding. We then scale the probabilities of all original source symbols P_X by $1 - p_\epsilon$, and use the subroutine Search() again to obtain a new codebook including the idleness symbol ϵ. The detailed procedure for the predictive scheme is shown in Algorithm 2.

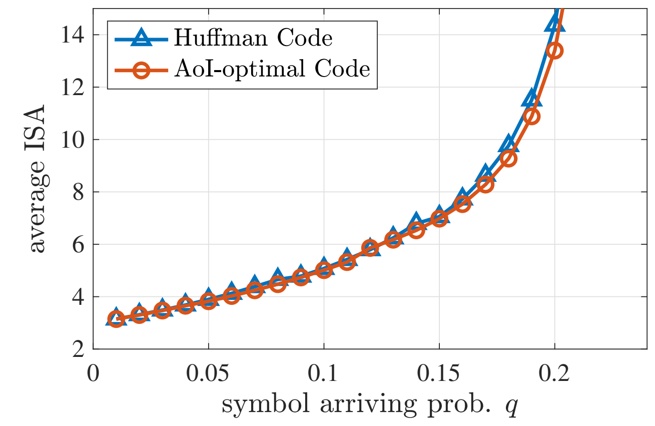
2.4.3 Numerical Evaluations

In this section, we examine the streaming source coding system by numerical experiments and evaluate both the SSA and ISA of the system. Figure 2.15 depicts the comparison among the experimental SSA, the upper bound in (2.83) and the lower bound in (2.84) using prefix idleness encoding. In this experiment, the source X follows the Zipf distribution with PMF

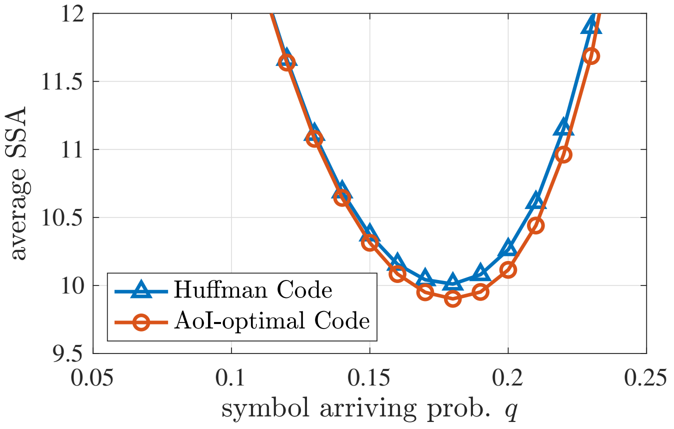
$$P_X(x) = \frac{1/x^s}{\sum_1^n 1/x^s},$$

where we choose the exponent $s = 1$ in Figure 2.15a and $s = 2$ in Figure 2.15b. We observe the average age has an inverted bell shape. When the symbol arrival probability q is small, the receiver has to wait a long time to get an informative source symbol, and the age is dominated by the slow symbol arrival rate. On the other hand, the system is overloaded if q is very large and the backlog of queued symbols grows without bounds. The upper bounds in both examples are very close to the experimental results, while the lower bound becomes looser as q increases.

Figure 2.16 depicts the comparison between the AoI-optimal code and the Huffman code for both ISA and SSA. In Figure 2.16a, we assume that the source X has 3 symbols with probabilities $P(A) = 0.8, P(B) = 0.15, P(C) = 0.05$. The AoI-optimal code is obtained based on the penalty function in (2.19). We evaluate the ISA for the AoI-optimal code and the Huffman code using blocklength $N = 4$, and vary the symbol arriving probability q. We observe that when q is small, the AoI-optimal code is exactly the Huffman code, which implies the system is mostly idle and the age is dominated by the average code length. As q increases, the age optimal code starts to provide lower average age than the Huffman code, meaning that the optimal coding strategy is to balance the average codelength and the second moment of the codelength. However, the difference between two coding schemes is actually in most of the time. This observation is similar to what we had for deterministic symbol arrivals in Section 2.2, which may be due to the limited variance in prefix codes. Figure 2.16b shows the evaluation of the SSA for source X following the Zipf distribution. where we set the alphabet size $|\mathcal{X}| = 100$ and the exponent $s = 2$. The AoI-optimal code is obtained based on the penalty function in (2.88). Similarly, the AoI-optimal code outperforms the Huffman code for large q as the average age starts to depend on higher moments of the codeword length.



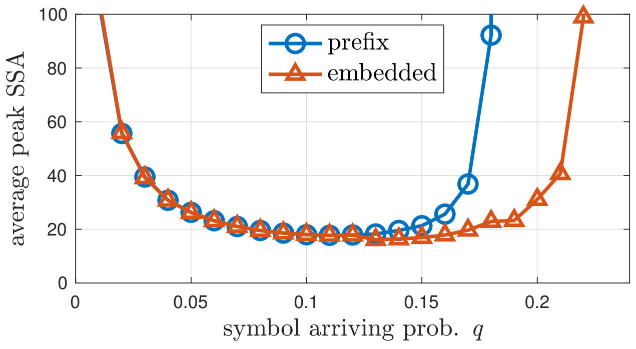
(a) ISA evaluation using blocklength $N = 4$.



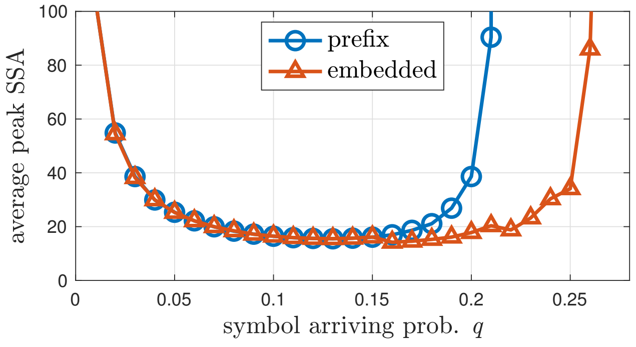
(b) SSA evaluation.

Figure 2.16: Comparison between AoI-optimal codes and Huffman codes.

Figure 2.17a and 2.17b depict average peak SSA for the two different empty buffer encoding schemes by varying the symbol arriving rate q between 0 and $1/H(X)$. In Figure 2.17a, the source X has 20 symbols and all the symbols are uniformly distributed with $P_X(x) = 1/20$. When the source arrival rate q is small, both scheme yield large average peak age, and the embedded scheme is identical to the prefix scheme as expected since the system is mostly idle. In this case, the optimal encoding scheme is to assign the shortest codeword to the null symbol ϵ. As q increases, the average peak age first decreases and then begins to rise since the system becomes unstable when the average length of message codeword $\mathrm{E}[L] > 1/q$. The curve corresponding to the embedded scheme blows up later than that of the prefix scheme when the system load becomes large. This is mainly because the embedded scheme assigns a longer codeword to the



(a) average peak SSA versus arriving probability q for uniform X.



(b) average peak SSA versus arriving probability q for Zipf X.

Figure 2.17: Comparison between prefix idleness encoding and embedded encoding for average peak SSA.

idleness symbol ϵ and this shortens the average length of the message codeword $\mathrm{E}[L_X]$. Figure 2.17b shows a similar experiment for source X following the Zipf distribution, where we set $n = |\mathcal{X}| = 20$ and the exponent $s = 1$. Similarly, the embedded scheme is identical to the prefix scheme when q is small, and it leads to lower age when the system load is larger.

2.5 Appendix

Proof of Lemma 2.4.5

In [16], the covariance of interarrival time Y_n and the system response time T_n is given by

$$\mathrm{Cov}[Y_n, T_n] = \int_0^\infty \mathrm{E}[(y - Y)(Y - \mathrm{E}[Y]) \,|\, Y \le y] Y(y) \mathrm{d}T(y). \tag{2.92}$$

Since both Y_n and T_n are discrete random variables, we rewrite the covariance as

$$\mathrm{Cov}[Y_n, T_n] = \sum_{y \in T} \mathrm{E}[(y - Y)(Y - \mathrm{E}[Y]) \,|\, Y \le y] \Pr[Y \le y] \Pr[T = y]. \tag{2.93}$$

We denote $f(y) = \mathrm{E}[(y - Y)(Y - \mathrm{E}[Y]) \,|\, Y \le y] \Pr[Y \le y]$ and thus $\mathrm{Cov}[Y_n, T_n] = \sum_{y \in T} f(y) \Pr[T = y]$. Then we can decompose $f(y)$ into

$$f(y) = f_1(y) + f_2(y) + f_3(y), \tag{2.94}$$

where

$$f_1(y) = (y + \mathrm{E}[Y]) \, \mathrm{E}[Y|Y \le y] \Pr[Y \le y], \tag{2.95}$$

$$f_2(y) = -\,\mathrm{E}\big[Y^2|Y \le y\big] \Pr[Y \le y], \tag{2.96}$$

$$f_3(y) = -y \, \mathrm{E}[Y] \Pr[Y \le y]. \tag{2.97}$$

For geometric Y, (2.95) can be rewritten as

$$\begin{aligned}
f_1(y) &= \left(y + \frac{1}{q}\right) \mathrm{E}[Y|Y \le y] \Pr[Y \le y] & (2.98)\\
&= \left(y + \frac{1}{q}\right) (\mathrm{E}[Y] - \mathrm{E}[Y|Y > y] \Pr[Y > y]) & (2.99)\\
&= \left(y + \frac{1}{q}\right) \left(\frac{1}{q} - \left(y + \frac{1}{q}\right)(1-q)^y\right) & (2.100)\\
&= \left(\frac{y}{q} + \frac{1}{q^2}\right) - \left(y + \frac{1}{q}\right)(1-q)^y. & (2.101)
\end{aligned}$$

Note that we use total probability to get (2.99) and the memoryless property in (2.100). Similarly, (2.96) and (2.97) are

$$\begin{aligned}
f_2(y) &= -\,\mathrm{E}\big[Y^2|Y \le y\big] \Pr[Y \le y] \\
&= -\big(\mathrm{E}[Y] - \mathrm{E}\big[Y^2|Y > y\big] \Pr[Y > y]\big) \\
&= -\left(\frac{2-q}{q^2} - (1-q)^y \sum_{x=y+1}^{\infty} x^2 \, \frac{(1-q)^{x-1} q}{(1-q)^y}\right)
\end{aligned}$$

$$
\begin{aligned}
&= -\frac{2-q}{q^2} + (1-q)^y \sum_{x=y+1}^{\infty} x^2 \, (1-q)^{x-y-1} q \\
&= -\frac{2-q}{q^2} + (1-q)^y \sum_{x'=1}^{\infty} (x'+y)^2 \, (1-q)^{x'-1} q \\
&= -\frac{2-q}{q^2} + (1-q)^y \sum_{x'=1}^{\infty} (x'+2yx'+y^2)^2 \, (1-q)^{x'-1} q \\
&= -\frac{2-q}{q^2} + (1-q)^y \big(\mathrm{E}\big[Y^2\big] + 2y \, \mathrm{E}[Y] + y^2\big) \\
&= -\frac{2-q}{q^2} + (1-q)^y \left(\frac{2-q}{q^2} + \frac{2y}{q} + y^2\right). \qquad (2.102)
\end{aligned}
$$

$$
\begin{aligned}
f_3(y) &= -y \, \mathrm{E}[Y] \Pr[Y \le y] \\
&= -\frac{y}{q} + \frac{y}{q}(1-q)^y. \qquad (2.103)
\end{aligned}
$$

Substituting (2.101), (2.102) and (2.103) back to (2.94) yields

$$
f(y) = -\frac{1-q}{q^2} + \left(\frac{1-q}{q^2} + \frac{y}{q}\right)(1-q)^y. \qquad (2.104)
$$

Thus, the covariance is

$$
\begin{aligned}
\mathrm{Cov}[Y_n, T_n] &= \sum_{y \in T} f(y) \Pr[T = y] \\
&= \sum_{y \in T} \left(-\frac{1-q}{q^2} + \left(\frac{1-q}{q^2} + \frac{y}{q}\right)(1-q)^y\right) \Pr[T = y] \\
&= -\frac{1-q}{q^2} + \frac{1-q}{q^2} \sum_{y \in T} \Big((1-q)^y \Pr[T = y]\Big) + \frac{1}{q} \sum_{y \in T} \Big(y(1-q)^y \Pr[T = y]\Big) \\
&= -\frac{1-q}{q^2} + \frac{1-q}{q^2} \, \mathrm{E}\big[(1-q)^T\big] + \frac{1}{q} \, \mathrm{E}\big[T(1-q)^T\big]. \qquad (2.105)
\end{aligned}
$$

We observe that $\mathrm{E}\big[(1-q)^T\big]$ is the PGF of the system time $\mathrm{E}\big[z^T\big]$ where $z = 1-q$, and

$$
\begin{aligned}
\mathrm{E}\big[T(1-q)^T\big] &= \sum_{y \in T} y(1-q)^y \Pr[T = y] \\
&= (1-q) \sum_{y \in T} y(1-q)^{y-1} \Pr[T = y] \\
&= (1-q) \sum_{y \in T} \frac{\mathrm{d}}{\mathrm{d}z} z^y |_{z=1-q} \Pr[T = y]
\end{aligned}
$$

$$\begin{aligned}
&= (1-q)\frac{\mathrm{d}}{\mathrm{d}z}\,\mathrm{E}\big[z^{T}\big]\big|_{z=1-q} \\
&= (1-q)\,T^{(1)}(1-q) \qquad (2.106)
\end{aligned}$$

Proof: $\bar{\Delta}_{SS,P}$ is a valid objective function.

For a specific equal age contour, we need to show that the contour is convex and non-increasing for any valid Δ. We denote $x = \mathrm{E}[L]$, $y = \mathrm{E}\big[L^2\big]$ and the upper bound on $z = \bar{\Delta}_{SS}$ in (2.83). For any given z, (2.83) can be written as the quadratic function

$$y(x) = 2x^2 - 2zx + \left(\frac{2}{q} - 2\right)z - \frac{2-q}{q^2} - 1, \qquad (2.107)$$

in which the discriminant is given by another quadratic function

$$\begin{aligned}
\delta(z) &= (2z)^2 - 8\left(\left(\frac{2}{q} - 2\right)z - \frac{2-q}{q^2} - 1\right) \\
&= 4\left(z^2 - \left(\frac{4}{q} - 4\right)z + \frac{2(2-q)}{q^2} + 2\right). \qquad (2.108)
\end{aligned}$$

We need to show that $\delta(z) \geq 0$ for all z, which is equivalent of showing the discriminant of $\delta(z)$ is always negative as follows

$$\begin{aligned}
\delta' &= \left(\frac{4}{q} - 4\right)^2 - 4\left(\frac{2(2-q)}{q^2} + 2\right) \\
&= 8 - \frac{24}{q}. \qquad (2.109)
\end{aligned}$$

Since $q \in (0,1)$, $\delta' < 0$ and thus $\delta(z) > 0$ for all z. That implies $y(x) = 0$ has two distinct roots and the contour $y(x)$ is convex and non-increasing.

Coin Collector's Problem and Package-Merge Algorithm.

In this section, we summarize the procedures used in [50, 52] for the subroutine MIN-LINEAR(α, β) in Algorithm 1. The package-merge algorithm is a greedy algorithm for finding an optimal solution for the coin collector's problem. It was first introduced in [50] to obtain an optimal length-limited Huffman code for a given distribution with alphabet

Algorithm 3 Package-Merge Algorithm

```
1: $S \leftarrow \emptyset$
2: $\forall d$, $L_d \leftarrow$ list of items with width $2^d$
3: while $t > 0$ do
4:     $minwidth \leftarrow$ the smallest term in the diadic expansion of $t$
5:     if $\mathcal{I} = \emptyset$ then
6:         return No Solution
7:     else
8:         $d \leftarrow \arg\min_d L_d$
9:         $r \leftarrow 2^d$
10:        if $r > minwidth$ then
11:            return No Solution
12:        else if $r > minwidth$ then
13:            Delete the minimum weight item from $L_d$ and insert it into $S$
14:            $t \leftarrow t - minwidth$
15:        $P_{d+1} \leftarrow$ PACKAGE($L_d$)
16:        discard $L_d$
17:        $L_{d+1} \leftarrow$ MERGE($P_{d+1}, L_{d+1}$)
18: return $S$
```

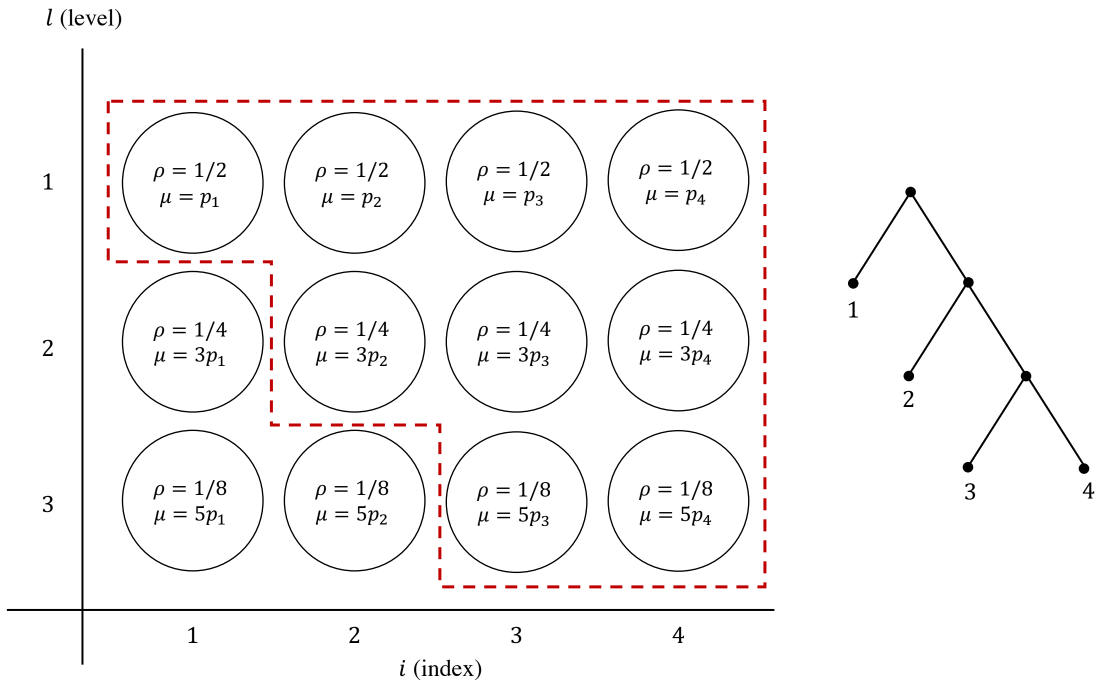
size n, in which no codeword is longer than L. In [52], this algorithm is revisited and extended to the problem of finding optimal code for quasi-arithmetic penalties of the codelength. An instance $(\mathcal{I}, t)$ of the coin collector's problem of size n is defined as the following:

1. A set $\mathcal{I}$ of m items, and each item has a *width* and *weight*. The width of each item is a (possibly negative) integral power of 2, and each weight is a real number.

2. A non-negative real number t, which is the *total width*.

A solution to such an instance is defined to be a subset $\mathcal{S}$ of $\mathcal{I}$ whose widths sum to t, and an optimal solution is a solution of minimum total weight. We can think of each item as a coin, whose width is the face value and weight is the actual numismatic value. The coin collector has run out of money and needs to use some of his collected coin to buy something of cost t. The target is to select a subset of coins from his collection of minimum numismatic value whose total face value is exactly t. Here we focus on the binary version of the problem, in which the face values will only be powers of 2.

We first let L_d be the list of items of width 2^d, and each list is sorted in the order of increasing weight. We also define the *dyadic expansion* of the total weight t as its



l (level)
1
2
3
$\rho = 1/2$ $\mu = p_1$
$\rho = 1/2$ $\mu = p_2$
$\rho = 1/2$ $\mu = p_3$
$\rho = 1/2$ $\mu = p_4$
$\rho = 1/4$ $\mu = 3p_1$
$\rho = 1/4$ $\mu = 3p_2$
$\rho = 1/4$ $\mu = 3p_3$
$\rho = 1/4$ $\mu = 3p_4$
$\rho = 1/8$ $\mu = 5p_1$
$\rho = 1/8$ $\mu = 5p_2$
$\rho = 1/8$ $\mu = 5p_3$
$\rho = 1/8$ $\mu = 5p_4$
1
2
3
4
i (index)
1
2
3
4

Figure 2.18: An example of the nodeset notation (on the left) and its equivalent tree notation (on the right). Each node (i, l) has width $\rho = 2^{-l}$ and weight $\mu = p_i(2l-1)$.

representation of powers of 2. For instance, 3.75 can be expressed as $2^1 + 2^0 + 2^{-1} + 2^{-2}$. Here we use the implementation of the package-merge algorithm from [50], which is illustrated in Algorithm 3. This implementation has $O(m)$ time if the items are presorted according to the weight.

In the Package() step, we form a new list P_{d+1} by combining the items in consecutive pairs from L_d, which is already sorted by the weights of the items. That is, the k-th item of P_{d+1} is the package formed by combining items $2k-1$ and $2k$ of L_d. If the length of L_d is an odd number, the item with the largest weight is simply discarded. The Merge() step is the merging of two sorted lists.

In order to reduce the codebook construction problem to the coin collector's problem, we first introduce the following *nodeset* notation for a codebook, which can also be easily converted to the tree representation. For a source with n symbols, we first define a node (i, l) to be an abstract representation of the item, in which $i \in [1, n]$ is the index of the node, and $l \in [1, l_{max}]$ is the level of the node. Any set of nodes is a *nodeset*. For a prefix-free code T that can be represented as a tree,

$$nodeset(T) = \{(i,l) | 1 \leq l \leq l_i\}, \tag{2.110}$$

where l_i is the depth of the i-th leaf of the tree notation for T as shown in Figure 2.18. The total weight or width of a codebook T is simply the sum of the weights or widths of all the nodes in $nodeset(T)$. We can see that the problem of finding an optimal codebook for a function $g(L) = \alpha\,\mathrm{E}[L] + \beta\,\mathrm{E}[L^2]$, can be translated to the problem of finding the nodeset with minimum total weight $g(L) = \alpha\,\mathrm{E}[L] + \beta\,\mathrm{E}[L^2]$. In order to set the total weight of $nodeset(T)$ to be $g(L)$, we let the width and weight for each node to be $\rho(i,l)$ and $\mu(i,l)$, which are given by

$$\begin{aligned} \rho(i,l) &= 2^{-l} \\ \mu(i,l) &= \alpha p_i + \beta p_i \left(l^2 - (l-1)^2\right). \end{aligned} \tag{2.111}$$

Figure 2.18 demonstrates an example with objective function $g(L) = \mathrm{E}[L^2]$. In this case, it follows from (2.111) that $\mu(i,l) = p_i(2l-1)$ for each node. For any prefix-free codebook T, it was also proved that the total width of $nodeset(T)$ is exactly $n-1$, which is also illustrated in the example in Figure 2.18. This enables us to directly obtain the optimal codebook for $g(L)$ by Algorithm 3 with width and weight in (2.111). For details about the optimality of the algorithm and proofs of this formulation of problem reduction, we suggest the reader to see the original work [50, 52].

Chapter 3

Distributing Timely Updates

3.1 Motivations

In Chapter 2, we mainly focus on the characterization of the update information itself, and try to develop the suitable source coding to convert those information into channel sequences given the channel capacity. In this case, the transmission time of an update is a function of the size of the update packet, since the encoded bits are transmitted to the destination sequentially through a pipe. This assumption holds when the channel connecting the source and the destination is lossless with constant data rate, mostly in low layer communications in optical fibers or wired links. However, if the source and destination are far apart in different network domains, and the update messages have to be encapsulated into higher layer packets and routed in the network, the transmission time of each update is very likely to be random. For example, the randomness in delay can be due to the queueing delay of a packet for network routing and congestion. Alternatively, the randomness may come from multiple trials of transmission of incremental encoded information to overcome packet loss [68]. Therefore, a different challenge arises when the source or transmitter is no longer capable of controlling the transmission time of an update.

Status updating through networks with random delays has been studied in different contexts. A rich class of literature examines the system where the source has the knowledge of the network service rate but not the delay of every update packet [3,4,11,17]. In this type of the system, the source or transmitter optimize the updating rate injected into the networking system based on the type of the network. Under many circumstances, the networking facility allowing new packet preemption leads to lower age of information at the destination since the service facility prioritizes fresher information over older ones. On the other hand, one can assume the delivery time of every update can be known to the sources via independent feedback channels, so that the sources can better manage the updating time based on the delivery acknowledgement [5, 21].

In next-generation wireless networks with pervasive computing and dense sensor networks, delay-sensitive data may be popular and simultaneously requested by large numbers of receivers [69–72]. For example, the information updates of an autonomous car will be broadcast to nearby vehicles and passengers equipped with portable sensing devices to construct a shared view of the local environment and maintain safety [73, 74]. In the future smart city, massively connected IoT sensors and devices will be deployed to gather the information about real-time traffic, air quality and etc. These real-time data will then be collected and distributed to multiple data centers for real-time traffic control and environmental monitoring [75, 76].

In order to support ubiquitous real-time applications, lots of attention has been paid to the concept and deployment of the network edge computing, or fog computing [77–79]. The network edge is more favorable than the core cloud network, since it provides closer storage and computing proximity for end users, usually within one or two network hops from the users and up to milliseconds delay. This benefits applications with ultra-low latency requirements, such as VR/AR gaming and autonomous driving, by offering computational offloading capability as well as content caching and sharing at the edge. Thus, the timely content updates generated by an application at the source should be distributed to the edge of the network redundantly, so that the nearby end users can connect to those edge storage nodes and fetch the updates in a timely way. We note that allocating content redundantly over multiple storage nodes has been well studied in cloud storage and the content distribution network (CDN) [80–83]. It has been shown that replicating data or applying erasure coding on the stored data not only increases the reliability of the data, but also benefits the accessibility by shorten the content download time through parallelism. Differently, here we utilize data redundancy in space to achieve better information freshness.

In this chapter, we consider a multicast strategy for the distribution of timely updates.

For every update, the source initiates a multicast transmission to the network and tries to deliver the update to a group of edge nodes. The transmission of each update takes random time in the network, due to the possible queueing delay in the network or the usage of retransmission and error correction to recover lost data. Since the total number of edge storage nodes can be very large and waiting for all the nodes to receive every update may be too time consuming, we consider a proactive updating system in which the source can stop the transmission of the current update at any time and start to transmit a newly generated update in order to prevent the information at the storage nodes from becoming stale. We note that each update packet can be relatively large, so that each packet may be divided into smaller data chunks for transmission in the network. Thus, stopping the transmission of the current update is simply stopping the transmission of the remaining small data chunks. If each update is a small packet, this can be done in practice by setting a time-to-live (TTL) value for each update packet when it was generated, so that the update packet will be dropped by the IP network when the TTL expires [84]. Alternatively, the transmission of the old update packets can also be paused or stopped in some network protocols that support prioritized packet preemption, such as IEEE 802.1 and 802.3 for time-sensitive data transmission in the Ethernet [85, 86]. Therefore, the source has the flexibility to control how long to wait for the multicast transmission of an update, and each update is likely to be delivered to only a subset of nodes due to early termination. Intuitively, if the source waits for a long time for each update transmission, each node is more likely to get an update, but the update message is also more likely to be outdated. On the other hand, if the source preempts the current update transmission earlier, each node is less likely to get an update, but the update information is fresher once it gets delivered.

Here we summarize and expand our results in single-hop multicast network with shifted exponential service time [87], multicast network with prioritized users [88], and content updates in Dynamo-type distributed storage [89]. In this chapter, we examine a more general update replication and distribution system as shown in Figure 3.1. A source generates and distributes timely updates to n nodes at the network edge through multicast network with independent and identical distributed (i.i.d.) random delays.

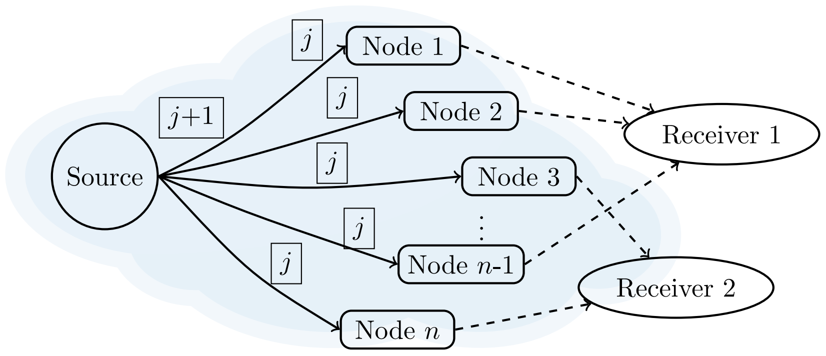


Figure 3.1: Multicast systems for timely updates: the source sequentially transmit content updates to multiple storage nodes at the network edge with random delays. The next update $j+1$ is generated right after the source terminates the transmission of update j. Each receiver has access to a random subset of r nodes and selects the freshest update from them.

The source controls when to stop the transmission of the current update and start the an update. We refer to the scheme to stop the current update as the *restart policy*, and the time duration for every update as the *service interval*. Since the receiving mobile device is usually with constrained network resources due to power limitation, the end user may only have access to a limited number of nearby edge nodes, and these nodes are usually selected in random due to the varying wireless channel condition and the real-time location of the mobile device [90–92]. Here we assume the receiver randomly connects to r nodes at the edge and fetches the most recent update from those r nodes at any time. Consequently, not every timely update may be successfully delivered to the receiver, and thus there is a probability for the receiver to obtain stale messages from the access nodes. We refer to r as the *access number*, which is the number of random nodes the receiver has access to. Our objective is to provide the answer to the following three questions:

1. Given that an access number r, what is the optimal restart strategy at the source that minimizes the age metric at the receiver?

2. Assuming the source can select the optimized restart strategy given an access number r, how many access nodes r does the receiver need to maintain a good enough age?

3. If the source chooses an optimized updating strategy based on a specific r_1 but the receiver has a different access number r_2, what will be the age performance at the receiver?

To start from the first question, we analyze the average age of the system for any given receiver access number r. Given the distribution of the network delay, we derive the optimal restart strategy analytically. For the second and third questions, we perform numerical evaluations based on our age optimized updating scheme.

3.2 Related Work

This work is mostly motivated by prior results about updates through erasure channels [32]. Since every data chunk can be erased with some probability, each update packet is transmitted with redundancy in time, either through maximum distance separable (MDS) codes with fixed redundancy, such as Reed-Solomon codes [93], or rateless codes with infinite incremental redundancy [94]. Transmission with fixed redundancy in time is equivalent to setting a fixed stopping time for every update transmission. Similarly, using rateless codes with update delivery feedback is the same as waiting for every update to be delivered to the destination. It was shown in [32] that the benefit of having channel feedback is minor in terms of the age, which matches one of our observations in this work. We also note that the model in [32] can be treated as a special case of this work with discrete-time erasure channel. The model we consider is also relevant to Hybrid-ARQ, a reliable transmission scheme that combines the conventional ARQ with error correction at the physical layer [95], and cross-layer coded multicast where an additional rateless code is applied at the packet level [68].

In [96] and [97], the age analysis is generalized to multicast networks with two-hop and multi-hop, respectively. The problem of randomly arriving updates at the multicast network is also addressed. In [25], it was shown that a greedy scheduling policy that prioritize the highest age packet provides optimal weighted sum of age over all users for wireless broadcast networks. In [23], update messages are replicated and sent to the receiver through multiple servers; given a general packet arrival process and memoryless packet service times, it was shown that Last-Generated First-Serve scheduling policy is age-optimal. In [98], a pull-based updating system is considered, in which the arriving source updates are sent to multiple servers and the interested user fetches the update by sending replicated requests to all the servers. Similar to our counting strategy in this work, it was shown there exists an optimal number of responses k from n servers for the user to wait for.

Data allocation with redundancy has captured lots of attention in the distributed cloud storage community. Most of the work on data allocation and access in distributed storage systems is concerned with either the recovery probability of the data [80, 81], or the accessibility, as measured by the download latency of the service rate [82, 83]. It has been shown that the optimized allocation scheme is usually non-intuitive and complicated, even for a given access model, and the optimization is challenging due to the combinatorial nature of the problem. Although erasure coding was shown to provide superior data reliability with less redundancy compared to replication, whether a particular coding scheme incur additional load to the system is unclear, especially in the unexplored network edge infrastructure side [99]. Therefore, the data redundancy method needs to be carefully chosen in these distributed computing systems. In this work, we mainly focus on the case where each update is replicated and stored as one copy in every edge node. Applying erasure coding on timely updates across multiple edge nodes will significantly reduce the storage overhead, but it requires an extensive study on the a joint coding-allocation design due to the multi-version problem across different nodes as shown in [100, 101].

3.3 Time-average Age Analysis

We consider a system with a single source generating timely updates at-will and broadcasting time-stamped updates to n storage nodes at the network edge through n links with independent random delays, as shown in Figure 3.1. Each update is time-stamped when it was generated. For every update j, it takes time X_{ij} to reach node i. We dnote X_{ij} as the service time, and assume X_{ij} to be i.i.d. for any link i and update

j. When most recently received update at time t at node i is time-stamped at time $u(t)$, the instantaneous age of information, or simply the *age*, is the random process $\Delta_i(t) = t - u(t)$. The evolution of the age process at a node can be described by the following two scenarios:

1. If the information at the node is not updated, i.e. $u(t)$ is fixed, the age process $\Delta(t)$ grows linearly in time t.

2. When an update successfully reaches the node at time t, $u(t)$ is advanced to the timestamp of the new update; let's call it $u'(t) \geq u(t)$. The age process $\Delta(t)$ has a discrete jump from $t - u(t)$ to $t - u'(t)$.

In this work, we consider a proactive updating system in which the source decides when to stop the transmission of the current update and start the transmission of a fresh update. For update j, the source waits for a time interval Y_j and then preempts the current update with a new update $j+1$ with a fresh timestamp. We refer to the policy to determine the service interval Y_j for all j as the *restart policy.* For example, Y_j can be a fixed value so that the source waits for a constant amount of time for each update j. Alternatively, the source can waits for the successful delivery of the current update j to a subset of k nodes. This makes Y_j a random variable determined by how long the transmission to those k nodes takes. The source starts the transmission of update $j+1$ immediately following the completion time of update j.

• **Definition:** A restart policy is **stationary** if the service intervals Y_j are i.i.d..

In order to obtain a fresh update at the network edge, a receiver has access to a subset of r edge nodes in the system randomly picket at the start of experiment, and retrieves the freshest content among these r nodes. We denote $\mathcal{R}$ as the subset of nodes the receiver is attached to. Under this model, the age at the client at time t is defined as the minimum age over all the nodes, i.e.

$$\Delta(t) = \min_{i \in \mathcal{R}} \Delta_i(t). \tag{3.1}$$

The time average age process at the receiver is given by

$$\Delta = \lim_{\tau \to \infty} \frac{1}{\tau} \int_{t=0}^{\tau} \Delta(t). \tag{3.2}$$

Since the service times X_{ij} are i.i.d. for each node i and content update j, the age processes at each node $\Delta_i(t)$ are statistically identical. If the receiver has access to only one node thus $r = 1$, the age process at the receiver is identical to the age at the node i, $\Delta(t) = \Delta_i(t)$. Since the service time X_{ij} is random, for any update j and a node i, there are two possibilities for the age process $\Delta_i(t)$ at the node i:

1. If $X_{ij} > Y_j$, i.e. the service time is greater than the allowed service interval, update j will not be delivered to node i and the age at that node $\Delta_i(t)$ increases linearly over time for the entire service interval.

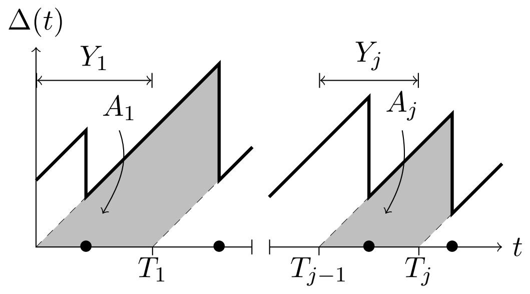
2. If $X_{ij} \leq Y_j$, i.e. the service time is shorter than the service interval, then update j will be delivered to node i. The instantaneous age $\Delta_i(t)$ will be reset to X_{ij} at the time of update delivery.

Let T_j be the time the source generates and starts to transmit the update j. Since the receiver obtains the freshest update among the r nodes in $\mathcal{R}$, whether an update j is delivered to the receiver or not depends on whether the update j is delivered to at least one of the r nodes in $\mathcal{R}$. Hence, for any update j, there are also two possibilities for the age process at the receiver $\Delta(t)$:

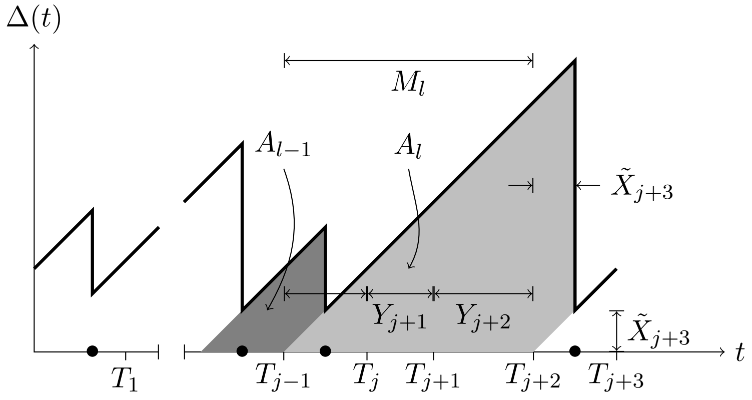
1. If $\min_{i \in \mathcal{R}} X_{ij} > Y_j$, none of the r nodes successfully receive a node and the receiver age $\Delta(t)$ increases linearly over time for the entire service interval.

2. If $\min_{i \in \mathcal{R}} X_{ij} \leq Y_j$, at least one of the r nodes receives update j within the service interval Y_j. The instantaneous age $\Delta_i(t)$ will be reset to X_{ij} at the time of earliest update delivery. That is,

$$\Delta_i(t)|_{t=T_j+\min_{i \in \mathcal{R}} X_{ij}} = \min_{i \in \mathcal{R}} X_{ij}. \tag{3.3}$$



(a) $p = 1$.



(b) $0 < p < 1$: successful updates occur in intervals 1, $j-1$, j, and $j+3$.

Figure 3.2: Sample path of the age $\Delta(t)$ with success probability p. Update delivery instances are marked by $\bullet$

After this delivery, the age increases linearly in time and none of the later deliveries matters since all the updates have the same generation timestamp.

We denote ψ_j to be the binary index of whether the update j is delivered to the receiver, thus

$$\psi_j = \begin{cases} 1, & \text{if } \min_{i\in\mathcal{R}} X_{ij} \le Y_j \\ 0, & \text{if } \min_{i\in\mathcal{R}} X_{ij} > Y_j. \end{cases} \tag{3.4}$$

For any stationary restart policy, the service interval Y_j are i.i.d. for all j. Since Y_j and any of $\{X_{ij} : i \in \mathbb{Z}^+\}$ are independent, $\{\psi_j : j \in \mathbb{Z}^+\}$ is a Bernoulli process with success probability $p = \Pr[\min_{i\in\mathcal{R}} X_{ij} \le Y_j]$.

If the system is designed such that every update j is delivered to the receiver, the success probability $p = 1$. This can be achieved by forcing the source to always wait for the delivery of an update to all the nodes. Alternatively, the system can choose a restart policy that yields $p < 1$, such as setting a constant time threshold τ for all updates j. In this case, an update j may or may not be delivered to the receiver, depending on the set of random service times $\{X_{ij} : i = 1, 2, \ldots, n\}$. Suppose an update is delivered to the receiver during service interval j and the next successful delivery occurs at interval $j + M$. Since $\{\psi_j\}$ is a Bernoulli process, M is a geometric r.v. with probability mass function (PMF)

$$P_M(m) = (1-p)^{m-1}p, \qquad m \geq 1.$$

Thus M has first and second moments

$$\mathrm{E}[M] = \frac{1}{p}, \qquad \mathrm{E}\big[M^2\big] = \frac{2-p}{p^2}. \tag{3.5}$$

Figure 3.2 depicts the sample paths of age with $p = 1$ and $0 < p < 1$, respectively. We represent the area under the age sawtooth as the concatenation of the polygons $A_1, \ldots, A_l$ as shown in Figs. 3.2a and 3.2b. In Figure 3.2b, the update j is delivered to the receiver $\mathcal{R}$ in the service interval j, and the receiver waits for $M_l = 3$ service intervals until the next successful update. Note that the restart policy with $p = 1$ can be viewed as a special case with deterministic $M_l = 1$ since the update secceeds at every service interval. We represent the length in time between two service intervals with successful updates as

$$W_l = \sum_{j'=j}^{j+M_l-1} Y_{j'}. \tag{3.6}$$

From Figure 3.2b, the average age is the average of the swatooth age diagram, which is

$$\begin{aligned}\Delta &= \lim_{L\to\infty} \frac{\sum_{l=1}^{L} A_l}{\sum_{l=1}^{L} W_l} \\ &= \lim_{L\to\infty} \frac{\frac{1}{L}\sum_{l=1}^{L} A_l}{\frac{1}{L}\sum_{l=1}^{L} W_l} = \frac{\mathrm{E}[A]}{\mathrm{E}[W]}.\end{aligned} \tag{3.7}$$

We also denote the random variable $\tilde{X}_j$ as the service time of a successful update delivered to the receiver, and $\tilde{X}_j$ has CDF

$$\begin{aligned} F_{\tilde{X}_j}(x) &= F_{\min_{i\in\mathcal{R}} X_{ij}|\psi_j=1}(x) \\ &= F_{\min_{i\in\mathcal{R}} X_{ij}|\min_{i\in\mathcal{R}} X_{ij}\le Y_j}(x). \end{aligned} \tag{3.8}$$

Evaluating Figure 3.2b, we have

$$A_l = \frac{1}{2}\Big(W_l + \tilde{X}_{j+M_l}\Big)^2 - \frac{1}{2}\tilde{X}^2_{j+M_l} = \frac{W_l^2}{2} + \tilde{X}_{j+M_l}W_l. \tag{3.9}$$

Thus, the expected area is

$$\mathrm{E}[A] = \frac{1}{2}\,\mathrm{E}\big[W^2\big] + \mathrm{E}\Big[\tilde{X}_{j+M_l}W_l\Big]. \tag{3.10}$$

Lemma 3.3.1. *The random variables $\tilde{X}_{j+M_l}$ and W_l are uncorrelated and*

$$\mathrm{E}\Big[\tilde{X}_{j+M_l}W_l\Big] = \mathrm{E}\Big[\tilde{X}\Big]\,\mathrm{E}[W]. \tag{3.11}$$

Proof. It follows from (3.6) that

$$\begin{aligned} \mathrm{E}\Big[\tilde{X}_{j+M_l}W_l\Big] &= \mathrm{E}\left[\tilde{X}_{j+M_l}\sum_{j'=j}^{j+M_l-1} Y_{j'}\right] \\ &= \mathrm{E}\Big[\tilde{X}_{j+m}\left(Y_j + Y_{j+1} + \ldots + Y_{j+m-1}\right)\Big]\,\Pr[M_l = m] \\ &= \mathrm{E}\Big[\tilde{X}\Big]\,\mathrm{E}[Y_j + Y_{j+1} + \ldots + Y_{j+m-1}]\,\Pr[M_l = m] \\ &= \mathrm{E}\Big[\tilde{X}\Big]\,\mathrm{E}[W] \end{aligned} \tag{3.12}$$

Here we use the fact that $\tilde{X}_m$ is independent of the service intervals $Y_0, \ldots, Y_{m-1}$ and is i.i.d. across all m. □

Substituting (3.10) and Lemma 3.3.1 into (3.7) yields the average age

$$\Delta = \frac{1}{\mathrm{E}[W]}\Big(\frac{1}{2}\,\mathrm{E}\big[W^2\big] + \mathrm{E}\Big[\tilde{X}\Big]\,\mathrm{E}[W]\Big)$$

$$= \frac{\mathrm{E}\left[W^2\right]}{2\,\mathrm{E}[W]} + \mathrm{E}\left[\tilde{X}\right]. \tag{3.13}$$

We remark that W_l is a random sum of random variables Y_j, in which the number of terms M_l and the sequence $\{Y_j\}$ may be dependent in (3.6). In order to obtain the first and second moment of W_l in (3.6), we first define Y_S as the length of a service interval given that the update is successfully delivered to the receiver, and Y_F as the length of a service interval given that the update is failed to be delivered. Thus, Y_S and Y_F have PDFs

$$f_{Y_S}(y) = f_{Y|\psi=1}(y) \tag{3.14}$$

$$f_{Y_F}(y) = f_{Y|\psi=0}(y). \tag{3.15}$$

Therefore, $\mathrm{E}[Y] = p\,\mathrm{E}[Y_S] + (1-p)\,\mathrm{E}[Y_F]$.

Lemma 3.3.2. *W has first and second moments*

$$\mathrm{E}[W] = \mathrm{E}[M]\,\mathrm{E}[Y], \tag{3.16}$$

$$\begin{aligned}\mathrm{E}\left[W^2\right] = {} & (\mathrm{E}[M]-1)\,\mathrm{Var}[Y_F] + \mathrm{Var}[Y_S] \\ & + \left(\mathrm{E}\left[M^2\right] - 2\,\mathrm{E}[M] + 1\right)(\mathrm{E}[Y_F])^2 \\ & + (\mathrm{E}[Y_S])^2 + 2(\mathrm{E}[M]-1)\,\mathrm{E}[Y_F]\,\mathrm{E}[Y_S]. \end{aligned} \tag{3.17}$$

Here we note that (3.16) is the general form of the Wald's identity, which holds even when M and Y are dependent. If M and Y are independent, we note that (3.17) is reduced to the Blackwell-Girshick equation [102]. Proof of the lemma appears in the Appendix. Substituting (3.5) into Lemma 3.3.2, and then substituting Lemma 3.3.2 back to (3.13) leads to the following theorem.

Theorem 3.3.3. *The average age at the receiver is*

$$\begin{aligned}\Delta = {} & \frac{1}{\mathrm{E}[Y]}\Big((1-p)\,\mathrm{Var}[Y_F] + p\,\mathrm{Var}[Y_S] + \frac{(1-p)(2-p)}{p}(\mathrm{E}[Y_F])^2 + p(\mathrm{E}[Y_S])^2 \\ & + 2(1-p)\,\mathrm{E}[Y_F]\,\mathrm{E}[Y_S]\Big) + \mathrm{E}\left[\tilde{X}\right].\end{aligned}$$

Theorem 3.3.3 holds for general stationary restart policy with i.i.d. service interval Y_j. If the system employs a restart policy such that the service interval Y_j is independent of the event of success delivery ψ_j, we have $f_{Y_F}(y) = f_{Y_S}(y) = f_Y(y)$. This also implies M_l and the sequence $\{Y_j\}$ in (3.6) are independent.

Corollary 3.3.4. *If the restart policy yields the service interval such that $f_{Y_F}(y) = f_{Y_S}(y) = f_Y(y)$, then the average age at the receiver is*

$$\Delta = \frac{2-p}{2p}\,\mathrm{E}[Y] + \frac{1}{2}\frac{\mathrm{Var}[Y]}{\mathrm{E}[Y]} + \mathrm{E}\big[\tilde{X}\big].$$

Proof. This can be shown by letting $\mathrm{E}[Y_F] = \mathrm{E}[Y_S] = \mathrm{E}[Y]$ and $\mathrm{Var}[Y_F] = \mathrm{Var}[Y_S] = \mathrm{Var}[Y]$ in Theorem 3.3.3. □

An example policy for Corollary 3.3.4 to hold is setting a fixed threshold τ for every update j, as discussed in Section 3.4.2. In this case, $Y_j = \tau$ and thus it's independent of whether an update is successfully delivered to the receiver as described by ψ_j. Another example is the policy that lets the source restart as soon as there are w nodes receives the current update for every update j. We show that the service interval Y_j only depends on w and the access number r in Section 3.4.1.

In contrast, we consider another policy in which Y_j is a random variable and the source draws every service interval Y_j according to some distribution $F_Y(y)$. For example, the source chooses Y_j to be either 1 or 10 equally likely for every j. For this randomized policy, Y_j and ψ_j are not independent and thus Corollary 3.3.4 does not apply. Given the condition that update j is delivered to the receiver, we know the service interval Y is more likely to be larger, and the expected value of Y_S is larger than Y. Similarly, the service interval is expected to be smaller given the update j is not delivered, implying that Y_F is expected to be smaller. In this case, we need to use Theorem 3.3.3 to obtain the average age expression.

3.4 Multicast with Homogeneous Nodes

The first system we investigate consists of only homogeneous nodes. That is, we assume the source treats all the edge nodes equally, and the deliveries of an update to every node is independent of the others.

3.4.1 Earliest-count Restart Policy with Feedback

Here we examine the Earliest-count (EC) restart policy in which the source counts the earliest number of nodes each update reaches and restarts once that number reaches the pre-determined threshold. Let's assume every node i can acknowledge the source through an independent instantaneous channel once the transmission of an update j is completed. For every update j, the source terminates the multicast transmission only if the source receives the earliest w acknowledgements from the n nodes in the system, where we refer to $w \in \{1, 2, \ldots, n\}$ as the *counter*. If one node receives the update earlier than any of the other nodes, it has to wait for an idle period until that content reaches $w - 1$ other nodes. Otherwise, if update j is not delivered to a node i, the node waits for the entire service interval Y_j until the source starts the next write. Since the service times $X_{1j}, X_{2j}, \ldots, X_{nj}$ are i.i.d., the service interval Y_j is the w-th smallest variable in $X_{1j}, X_{2j}, \ldots, X_{nj}$.

3.4.1.1 Notation for Order Statistics

We denote the k-th order statistic of a set of n i.i.d. random variables $X_1, \ldots, X_n$, i.e., the k-th smallest variable, as $X_{k:n}$. For example, $X_{1:n} = \min_k X_k$. The mean, variance and m-th moment of $X_{k:n}$ is denoted by $\mu_{k:n}$, $\sigma^2_{k:n}$ and $\mu^{(m)}_{k:n}$, respectively. The order statistics for some distributions are known as follows [103].

Shifted Exponential: For shifted exponential distributed X with cumulative distributed function (CDF)

$$F_X(x) \;=\; 1 - e^{-\lambda(x-c)}, \quad x \geq c,$$

the mean and variance of the order statistic $X_{k:n}$ are given by

$$\mu_{k:n} = c + \frac{1}{\lambda}(H_n - H_{n-k}) \tag{3.18a}$$

$$\sigma^2_{k:n} = \frac{1}{\lambda^2}\left(H_{n^2} - H_{(n-k)^2}\right) \tag{3.18b}$$

where H_n and H_{n^2} are the generalized harmonic numbers defined as

$$H_n = \sum_{j=1}^{n} \frac{1}{j}$$

$$H_{n^2} = \sum_{j=1}^{n} \frac{1}{j^2}. \tag{3.19}$$

Thus the second moment of the order statistic is

$$\mu^{(2)}_{k:n} = c^2 + \frac{2c}{\lambda}(H_n - H_{n-k}) + \frac{1}{\lambda^2}\left((H_n - H_{n-k})^2 + H_{n^2} - H_{(n-k)^2}\right). \tag{3.20}$$

Pareto: For Pareto distributed X with CDF

$$F_X(x) = 1 - \left(\frac{b}{x}\right)^{\nu}, \tag{3.21}$$

the m-th moment of the order statistics is given by

$$\mu^{(m)}_{k:n} = b^m \frac{\Gamma(n+1)\Gamma(n-k+1-\frac{m}{\nu})}{\Gamma(n-k+1)\Gamma(n+1-\frac{m}{\nu})}, \tag{3.22}$$

where $\Gamma(n) = (n-1)!$ is the gamma function.

3.4.1.2 Age Calculation

For Earliest-count policy, we note that the service interval is the w-th order statistics denoted by

$$Y_j = X_{w:n}. \tag{3.23}$$

In this case, the delivery of an update to the receiver depends on the choices of w and r. We first denote $\mathcal{W}_j$ as the set of w nodes receiving an update during the service interval j; $\mathcal{W}_j$ is randomly chosen among all n nodes at every service interval j, since the service times X_{ij} are i.i.d. for all i and j. For $r+w>n$, there is at least one overlapping node between the set $\mathcal{R}$ and the set $\mathcal{W}_j$, and thus every update will be eventually delivered to at least one of the nodes in $\mathcal{R}$, and thus to the receiver. In this case, the indicator $\psi_j = 1$ for all j, and the age process evolves as shown in Figure 3.2a. For $r+w \le n$, an update j may not be delivered to the receiver since the set $\mathcal{W}_j$ and set $\mathcal{R}$ can be disjoint. This occurs with probability $\binom{n-w}{r}/\binom{n}{r}$ since $\mathcal{W}_j$ is randomly chosen for each j. Let's denote the probability of an update failure at the receiver as $q = 1-p$, then

$$q_{(w,r)} = \Pr\{\mathcal{R} \cap \mathcal{W} = \emptyset\} = \begin{cases} 0, & r+w>n \\ \binom{n-w}{r}/\binom{n}{r}, & r+w \le n. \end{cases} \tag{3.24}$$

For a given total number of nodes n, the failure probability q only depends on w and r. Here we note that the indicator ψ_j is determined by whether the two subsets w_j and r overlap, and Y_j is determined by the w-th smallest variable in $X_{1j}, X_{2j}, \ldots, X_{nj}$ regardless of the random variable ψ_j.

Lemma 3.4.1. *The service interval Y_j is independent of ψ_j.*

Proof. We first denote $\mathbf{X}_j = [X_{1j}, \ldots, X_{nj}]$ as the vector of service times at the j-th service interval. Since the subset $\mathcal{R}$ is randomly chosen,

$$\begin{aligned} &\Pr[\psi_j = 0 | \mathbf{X}_j = \mathbf{x}_j] = q_{(w,r)} \\ &\Pr[\psi_j = 1 | \mathbf{X}_j = \mathbf{x}_j] = 1 - q_{(w,r)}. \end{aligned} \tag{3.25}$$

Thus, ψ_j is independent of $\mathbf{X}_j$, and for any set $\mathcal{A}$, we have

$$\Pr[\psi_j = 0, \mathbf{X}_j \in \mathcal{A}] = \Pr[\psi_j = 0] \Pr[\mathbf{X}_j \in \mathcal{A}]. \tag{3.26}$$

For Earliest-count policy, $Y_j = X_{w:n}$ and thus we can rewrite it as a function $Y_j = g_w(\mathbf{X}_j)$

and let $\mathcal{A}(y) = \{\mathbf{x}_j \mid g_w(\mathbf{x}_j) \le y\}$. Therefore,

$$\begin{aligned}
\Pr[\psi_j = 0, Y_j \le y] &= \Pr[\psi_j = 0, \mathbf{X}_j \in \mathcal{A}(y)] \\
&= \Pr[\psi_j = 0] \Pr[\mathbf{X}_j \in \mathcal{A}(y)] \\
&= \Pr[\psi_j = 0] \Pr[Y_j \le y]. \qquad (3.27)
\end{aligned}$$

And we can repeat the step for the complement case $\psi_j = 1$. Thus, ψ_j and Y_j are independent. □

Therefore, the average age can be obtained through Corollary 3.3.4.

Theorem 3.4.2. *For the Earliest-count restart policy with w, the average age at the receiver with access to r nodes is*

1. *for $w + r > n$,*

$$\Delta_{(w,r)} = \sum_{k=1}^{w} \mu_{k:n} \frac{\binom{n-k}{r-1}}{\binom{n}{r}} + \frac{1}{2} \frac{\mu^{(2)}_{w:n}}{\mu_{w:n}};$$

2. *for $w + r \le n$,*

$$\Delta_{(w,r)} = \sum_{k=1}^{w} \mu_{k:n} \frac{\binom{n-k}{r-1}}{\binom{n}{r} - \binom{n-w}{r}} + \frac{1}{2} \frac{\binom{n}{r} + \binom{n-w}{r}}{\binom{n}{r} - \binom{n-w}{r}} \mu_{w:n} + \frac{1}{2} \frac{\sigma^2_{w:n}}{\mu_{w:n}}.$$

Proof. Substituting $p = 1 - q$ into Corollary 3.3.4 yields

$$\Delta = \frac{1+q}{2(1-q)} \mathrm{E}[Y] + \frac{1}{2} \frac{\mathrm{Var}[Y]}{\mathrm{E}[Y]} + \mathrm{E}\left[\tilde{X}\right]. \qquad (3.28)$$

It follows from (3.24) that

$$\frac{1+q}{2(1-q)} = \begin{cases} \frac{1}{2}, & r + w > n \\ \frac{1}{2} \frac{\binom{n}{r} + \binom{n-w}{r}}{\binom{n}{r} - \binom{n-w}{r}}, & r + w \le n. \end{cases} \qquad (3.29)$$

Thus, for $w+r>n$, (3.28) can be written as

$$\begin{aligned}\Delta &= \frac{1}{2}\,\mathrm{E}[Y] + \frac{1}{2}\frac{\mathrm{Var}[Y]}{\mathrm{E}[Y]} + \mathrm{E}[\tilde{X}] \\ &= \frac{1}{2}\frac{\mathrm{E}\left[Y^2\right]}{\mathrm{E}[Y]} + \mathrm{E}[\tilde{X}]. \qquad (3.30)\end{aligned}$$

Denote the node $\underline{i}_{\mathcal{R}}$ as the node with least service time in the set of receiving nodes $\mathcal{R}$, i.e.,

$$\underline{i}_{\mathcal{R}} = \arg\min_{i\in\mathcal{R}} X_i.$$

We note that the subset $\{X_i \,|\, i\in\mathcal{R}\}$ is constructed by choosing r nodes from the set $\{X_{1:n},\dots,X_{n:n}\}$. The node $\underline{i}_{\mathcal{R}}$ has the smallest service time means the remaining $r-1$ service times in $\mathcal{R}$ are greater than k-th smallest among all n nodes Thus, the remaining $r-1$ service times are randomly chosen from the subset $\{X_{k+1:n},\dots,X_{n:n}\}$. Thus, the probability of the node $\underline{i}_{\mathcal{R}}$ being the k-th smallest among all n nodes is

$$\Pr[\underline{i}_{\mathcal{R}} = i_k] = \binom{n-k}{r-1}\Big/\binom{n}{r}, \qquad (3.31)$$

In addition, we rewrite the set of nodes with successful updates as

$$\mathcal{W} = \{i_1, i_2, \dots, i_w\}, \qquad (3.32)$$

where $X_{i_k} = X_{k:n}$ is the k-th smallest service time.

For $w+r>n$, the average service time for a successful update delivered to the client is

$$\begin{aligned}\mathrm{E}\left[\tilde{X}\right] &= \mathrm{E}\left[X_{\underline{i}_{\mathcal{R}}}\right] \qquad (3.33a)\\ &= \sum_{k=1}^{n-r+1} \mathrm{E}\left[X_{\underline{i}_{\mathcal{R}}} \,|\, \underline{i}_{\mathcal{R}} = i_k\right] \Pr\left[\underline{i}_{\mathcal{R}} = i_k\right], \qquad (3.33b)\\ &= \sum_{k=1}^{n-r+1} \mathrm{E}[X_{k:n}] \Pr[\underline{i}_{\mathcal{R}} = i_k], \qquad (3.33c)\end{aligned}$$

$$= \sum_{k=1}^{n-r+1} \mu_{k:n} \frac{\binom{n-k}{r-1}}{\binom{n}{r}}. \tag{3.33d}$$

In (3.33a), the expectation of $\tilde{X}$ is defined as the expectation of the minimum of all the service times X_i in the receiving set $\mathcal{R}$, given that this node $\underline{i}_{\mathcal{R}}$ is also in the set $\mathcal{W}$. When $w + r > n$, the node $\underline{i}_{\mathcal{R}}$ is always in $\mathcal{W}$, and we note that the order of $\underline{i}_{\mathcal{R}}$, which is denoted by k, can be at most $n - r + 1$ since there are $r - 1$ nodes in r that with larger service time. (3.33b) is obtained by averaging over the conditional expectation of all possible order statistics $X_{k:n}$, and (3.33d) follows from (3.31).

For $w + r \leq n$, the node $\underline{i}_{\mathcal{R}}$ is not necessarily in the successful subset $\mathcal{W}$, and thus the order of $\underline{i}_{\mathcal{R}}$, which is denoted by k, can be at most w. Hence,

$$\mathrm{E}[\tilde{X}] = \mathrm{E}\big[X_{\underline{i}_{\mathcal{R}}} \,|\, \underline{i}_{\mathcal{R}} \in \mathcal{W}\big] \tag{3.34a}$$

$$= \sum_{k=1}^{w} \mathrm{E}[X_{k:n}] \Pr\{\underline{i}_{\mathcal{R}} = i_k \mid \underline{i}_{\mathcal{R}} \in \mathcal{W}\}, \tag{3.34b}$$

$$= \sum_{k=1}^{w} \mu_{k:n} \frac{\Pr[\underline{i}_{\mathcal{R}} = i_k]}{1 - q}, \tag{3.34c}$$

$$= \sum_{k=1}^{w} \mu_{k:n} \frac{\binom{n-k}{r-1}}{\binom{n}{r} - \binom{n-w}{r}}. \tag{3.34d}$$

In (3.34d), we apply (3.31) and $q = \binom{n-w}{r}/\binom{n}{r}$ in (3.24) for the case $r + w \leq n$. Note that the length of a service interval is $Y = X_{w:n}$. To get Theorem 3.4.2, we substitute (3.33d) back to (3.30) for $w + r > n$, and substitute (3.29) and (3.34d) back to (3.28) for $w + r \leq n$.

□

For a special case where the receiver has access to only one node, we have the following corollary by setting $r = 1$ in Theorem 3.4.3.

Corollary 3.4.3. *For the Earliest-count restart policy with w, the average age at the receiver with access to only $r = 1$ nodes is*

$$\Delta_{(w,r=1)} = \frac{1}{w} \sum_{k=1}^{w} \mu_{k:n} + \frac{2n - w}{2w} \mu_{w:n} + \frac{\sigma_{w:n}^2}{2\mu_{w:n}}.$$

If the source waits for all the nodes for every update, i.e. $w = n$, we denote the expected service time as $\mu = \mathrm{E}[X]$ and Corollary 3.4.3 becomes

$$\Delta_{(w=n,r=1)} = \mu + \frac{1}{2}\frac{\mu_{n:n}^{(2)}}{\mu_{n:n}}. \tag{3.35}$$

In both Theorem 3.4.2 and Corollary 3.4.3, the average age depends on the first and second moment of the order statistics of the i.i.d. service time X. Assuming r and the distribution of X is known to the source, our objective is to select the optimal stopping counter w for the source such that the average age is minimized.

3.4.1.3 Shifted Exponential Model

We first consider the case where the service time follows a shifted exponential distribution with CDF

$$F_X(x) \;=\; 1 - e^{-\lambda(x-c)}, \quad x \geq c. \tag{3.36}$$

The constant time shift $c > 0$ captures the delay produced by the update generation and assembly process. For example, c could represent the time to fetch temperature data from a large set of sensors in the network, or the time to compress a video frame into a size-constrained packet in remote video surveillance. On the other hand, c can also represent a propagation delay on top of an exponential network delay if the source and database are geographically separated.

Theorem 3.4.4. *Let n be large and $w < n$, we denote $\beta = 1 - w/n$ and $\theta = r/n$. For shifted exponential (λ, c) service time X and a given r, the average age at the receiver can be approximated as:*

1. *for $w + r > n$,*

$$\Delta_{(w,r)} \approx \hat{\Delta}_1(\beta) = \left(\frac{1}{\lambda r} + c\right)(1-\theta^r) - \frac{1}{\lambda}\theta^r \log\frac{1}{\theta} + \frac{1}{2\lambda}\log\frac{1}{\beta} + \frac{c}{2}.$$

2. for $w + r \leq n$,

$$\Delta_{(w,r)} \approx \hat{\Delta}_2(\beta) = \frac{1}{\lambda r} + \frac{1}{2\lambda}\log\frac{1}{\beta} + c + \frac{c(1+\beta^r)}{2(1-\beta^r)}.$$

Theorem 3.4.4 relies on the asymptotic approximation of the Harmonic function in (3.18a) by $H_k \approx \log k + \gamma$, where $\gamma \approx 0.577$ is the Euler-Mascheroni constant. For a given β, we observe that the difference between the approximations in Theorem 3.4.4 is given by

$$\delta(\beta) = \hat{\Delta}_2(\beta) - \hat{\Delta}_1(\beta) = \frac{\theta^r}{\lambda}\log\frac{1}{\theta} + \theta^r(c + \frac{1}{\lambda r}) + \frac{c\beta^r}{1-\beta^r}. \tag{3.37}$$

Since $\beta \in (0,1)$ and $\theta \in (0,1)$, we observe $\delta(\beta) > 0$ and thus $\hat{\Delta}_2(\beta) > \hat{\Delta}_1(\beta)$ for any β. Next, we focus on the case where $w + r > n$ and $\beta = 1 - w/n < r/n = \theta$. We show that the function $\delta(\beta) \to 0$ for any β and a given r, as $n \to \infty$ and thus $\hat{\Delta}_2(\beta)$ serves as a good approximation within this region as well.

Since $r \geq 1$ and $0 < \beta < 1$, we have the following upper bound

$$\delta(\beta) \leq \frac{\theta}{\lambda}\log\frac{1}{\theta} + \theta(c + \frac{1}{\lambda r}) + \frac{c\beta}{1-\beta}, \tag{3.38}$$

with equality holding at $r = 1$. We then let $\delta_1 = \frac{\theta}{\lambda}\log\frac{1}{\theta}$ and

$$\begin{aligned} \lim_{n\to\infty} \delta_1 &= \lim_{n\to\infty} \frac{r}{n\lambda}\log\frac{n}{r} \\ &= 0. \end{aligned} \tag{3.39}$$

Similarly, we let $\delta_2(\beta) = \theta(c + \frac{1}{\lambda r}) + \frac{c\beta}{1-\beta}$ and $\delta_2(\beta)$ is monotonically increasing in β. Thus,

$$\begin{aligned} \lim_{n\to\infty} \delta_2(\beta) &\leq \lim_{n\to\infty} \delta_2(\beta = \theta) \\ &= \lim_{n\to\infty} \frac{r}{n}(c + \frac{1}{\lambda r}) + \frac{cr/n}{1 - r/n} \\ &= 0. \end{aligned} \tag{3.40}$$

Therefore,

$$\lim_{n\to\infty} \delta(\beta) \leq \lim_{n\to\infty} \delta_1 + \delta_2(\beta) = 0. \tag{3.41}$$

We note that (3.41) holds for any $\beta \in (0, r/n)$. This implies that the difference between the approximations $\hat{\Delta}_1(\beta)$ and $\hat{\Delta}_2(\beta)$ becomes negligible for large n. Thus $\hat{\Delta}_2(\beta)$ can be also used as an approximation for the region $w + r > n$.

Corollary 3.4.5. *The optimal β^* that minimizes the approximation $\hat{\Delta}_2(\beta)$ in Theorem 3.4.4 for positive λ and c is*

$$\beta^* = \left((\lambda cr + 1) - \sqrt{(\lambda cr + 1)^2 - 1}\right)^{1/r}.$$

The detailed proofs of Theorem 3.4.4 and Corollary 3.4.5 are provided in the appendix. Based on the optimal β^* obtained in Corollary 3.4.5, the optimal choice of stopping counter w^* can be obtained by solving

$$w^* = n\,(1 - \beta^*) \tag{3.42}$$

Since w^* may not be an integer, we will need to round it to the closest integer. For a given r, we observe in Corollary 3.4.5 that the optimal η^* depends on the shifted exponential parameters λ and c but not the total number of nodes n. This implies the source should always wait for a constant $\alpha^* = 1 - \beta^*$ fraction of all the nodes for every service interval. Based on this observation, we have the following results on the length of service interval.

Corollary 3.4.6. *For shifted exponential (λ, c) service times X and a given r, the expected service interval using the optimal restart counter w^* is approximated by*

$$\mathrm{E}[Y] \approx c - \frac{1}{\lambda r} \log\left((\lambda cr + 1) - \sqrt{(\lambda cr + 1)^2 - 1}\right). \tag{3.43}$$

Corollary 3.4.6 follows from substituting Corollary 3.4.5 back to the approximation of expected service interval $\mathrm{E}[Y]$. See appendix for the proof. We also observe that the expected stopping time for every update is independent of the total number of nodes n. For Earliest-counter restart policy with instantaneous feedback, we demonstrate the optimal strategy for the source is to wait for a constant fraction of nodes independent of n, which also gives a constant expected waiting time. This motivates us to investigate an alternative threshold-type restart policy which doesn't require the feedback of update delivery.

3.4.1.4 Pareto Model

Pareto distribution is a simple heavy-tailed distribution that has been widely used to model content transfer delay, computing delay and network traffic with potential stragglers [104–106]. For Pareto distributed random variable X with CDF

$$F(x) = 1 - (b/x)^{\nu}, \tag{3.44}$$

the scale parameter b marks the non-zero offset of the distribution, and the shape parameter ν describes the power law of the tail of the distribution. The tail of the Pareto distribution decays faster for larger b and ν. We first approximate the order statistic of Pareto r.v. X as follows.

Lemma 3.4.7. *Let n be large and $k < n$, we denote $\alpha = k/n$. The expected k-th order statistics in (3.22) with $m = 1$ can be approximated by*

$$\mu_{k:n} \approx b(1-\alpha)^{-\frac{1}{\nu}}. \tag{3.45}$$

Proof. Asymptotically as the number of nodes $n \to \infty$, we can appoximate the Gamma function by Stirling's formula as

$$\Gamma(n+1) \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^{n}. \tag{3.46}$$

Thus, the expected order statistic in (3.22) with $m = 1$ can be approximated by

$$
\begin{aligned}
\mu_{k:n} &\approx b\,\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \sqrt{2\pi(n-k-\frac{1}{\nu})}\left(\frac{(n-k-\frac{1}{\nu})}{e}\right)^{n-k-\frac{1}{\nu}}}{\sqrt{2\pi(n-k)}\left(\frac{n-k}{e}\right)^{n-k}\sqrt{2\pi(n-\frac{1}{\nu})}\left(\frac{(n-\frac{1}{\nu})}{e}\right)^{n-\frac{1}{\nu}}} && (3.47)\\
&= b\,\frac{\sqrt{n(n-k-\frac{1}{\nu})}\,n^n(n-k-\frac{1}{\nu})^{n-k-\frac{1}{\nu}}}{\sqrt{(n-k)(n-\frac{1}{\nu})}\,(n-k)^{n-k}(n-\frac{1}{\nu})^{n-\frac{1}{\nu}}} && (3.48)\\
&\approx b\,\frac{\sqrt{n(n-k)}}{\sqrt{(n-k)n}}\,\frac{n^n(n-k)^{n-k-\frac{1}{\nu}}}{(n-k)^{n-k}n^{n-\frac{1}{\nu}}} && (3.49)\\
&= b\,\frac{n^{\frac{1}{\nu}}}{(n-k)^{\frac{1}{\nu}}} && (3.50)\\
&= b(1-\alpha)^{-\frac{1}{\nu}}. && (3.51)
\end{aligned}
$$

□

We have the following theorem from Theorem 3.4.2 and the above approximation.

Theorem 3.4.8. *Let n be large and $w < n$, we denote $\beta = 1 - w/n$ and $\theta = r/n$. For Pareto (b, ν) service time X and a given r, the average age at the receiver can be approximated as:*

1. *for $w + r > n$,*

$$\Delta_{(w,r)} \approx \hat{\Delta}_1(\beta) = \frac{br}{r-\frac{1}{\nu}}\left(1-\theta^{r-\frac{1}{\nu}}\right) + \frac{b}{2}\beta^{-\frac{1}{\nu}}.$$

2. *for $w + r \le n$,*

$$\Delta_{(w,r)} \approx \hat{\Delta}_2(\beta) = \frac{br}{(r-\frac{1}{\nu})(1-\beta^r)}\left(1-\beta^{r-\frac{1}{\nu}}\right) + \frac{b\,(1+\beta^r)}{2\,(1-\beta^r)}\beta^{-\frac{1}{\nu}}.$$

See the Appendix for the proof. When the receiver has access to only a single node in the system, the approximation is given by the following corollary by letting $r = 1$ in the approximation 2) in Theorem 3.4.8.

Corollary 3.4.9. *Let n be large and $w < n$, we denote $\alpha = w/n$. For Pareto (b, ν) service time X, the average age at the receiver with access to $r = 1$ node is approximated* *by*

$$\Delta_{(w,r=1)} \approx \hat{\Delta}(\alpha) = \frac{b\nu}{(\nu-1)\alpha} + \frac{b(\nu+1)\alpha - 2b}{2(\nu-1)\alpha}(1-\alpha)^{-\frac{1}{\nu}}.$$

For a given β, we observe that the difference between the approximations in Theorem 3.4.8 is given by

$$\delta(\beta) = \frac{br}{r-\frac{1}{\nu}}\left(\frac{1-\beta^{r-\frac{1}{\nu}}}{1-\beta^r} - \left(1-\theta^{r-\frac{1}{\nu}}\right)\right) + \frac{b\beta^{r-\frac{1}{\nu}}}{1-\beta^r}. \tag{3.52}$$

Since $\nu > 1$ and $r > 1$, $r - 1/\nu > r - 1 > 0$. Therefore, $\delta(\beta) > 0$ for $\beta \in (0,1)$. Similar to the discussion for shifted exponential service time, we focus on the case where $w + r > n$ and $\beta = 1 - w/n < r/n = \theta$. We show that the function $\delta(\beta) \to 0$ for any β and a given r, as $n \to \infty$.

Since $\beta < r/n$, we let

$$\begin{aligned}
\delta_1(\beta) &= \frac{br}{r-\frac{1}{\nu}}\left(\frac{1-\beta^{r-\frac{1}{\nu}}}{1-\beta^r} - \left(1-\theta^{r-\frac{1}{\nu}}\right)\right)\\
&\le \frac{br}{r-\frac{1}{\nu}}\left(1 - \left(1-\theta^{r-\frac{1}{\nu}}\right)\right)\\
&= \frac{br\theta^{r-\frac{1}{\nu}}}{r-\frac{1}{\nu}}\\
&\le \frac{br\left(\frac{r}{n}\right)^r}{r-\frac{1}{\nu}}\\
&= \frac{br^{r+1}}{\left(r-\frac{1}{\nu}\right)n^r}.
\end{aligned} \tag{3.53}$$

Similarly, we have

$$\begin{aligned}
\delta_2(\beta) &= \frac{b\beta^{r-\frac{1}{\nu}}}{1-\beta^r}\\
&\le \frac{b\left(\frac{r}{n}\right)^{r-\frac{1}{\nu}}}{1-\left(\frac{r}{n}\right)^r}\\
&\le \frac{br^{r-\frac{1}{\nu}}n^{\frac{1}{\nu}}}{n^r - r^r}.
\end{aligned} \tag{3.54}$$

Therefore,

$$\begin{aligned}\lim_{n\to\infty}\delta(\beta) &= \lim_{n\to\infty}\delta_1(\beta)+\delta_2(\beta)\\ &\leq \lim_{n\to\infty}\frac{br^{r+1}}{\left(r-\frac{1}{\nu}\right)n^r}+\frac{br^{r-\frac{1}{\nu}}n^{\frac{1}{\nu}}}{n^r-r^r}\\ &= 0. \end{aligned} \tag{3.55}$$

We note that (3.55) holds for any $\beta \in (0, r/n)$. This implies that the difference between the approximations $\hat{\Delta}_1(\beta)$ and $\hat{\Delta}_2(\beta)$ is negligible and thus $\hat{\Delta}_2(\beta)$ also serves as an approximation for the region $w + r > n$.

Corollary 3.4.10. *The optimal $\beta^* \in (0, 1)$ that minimizes the approximation $\hat{\Delta}_2(\beta)$ in Theorem 3.4.8 is the root β of the following equation*

$$(1+r\nu)\,\beta^{2r} - 2r^2\nu^2\beta^{r+\frac{1}{\nu}} + r\nu - 1 = 0. \tag{3.56}$$

If the receiver has access to only a single node, we let $r = 1$ in Corollary 3.4.10 and have the following corollary.

Corollary 3.4.11. *The optimal $\alpha^* \in (0, 1)$ that minimizes the approximation in Corollary 3.4.9 is the root of the following equation*

$$(1+\nu)\,\alpha^2 - \alpha - \nu(1-\alpha)^{\frac{1}{\nu}} = 0. \tag{3.57}$$

3.4.1.5 Connection with Dynamo-style Storage Updates

Here we note that our system model with Earliest-count restart policy is also applicable to Dynamo-style storage systems. In modern distributed storage systems, data is often replicated across multiple machines or data centers to support fault tolerance due to hardware failures, power outages or catastrophe. Replication of data also provides higher availability by allowing simultaneous read from multiple servers. In order to overcome the asynchrony in distributed storage systems, *quorum*-based algorithms [107–110] are well developed and widely used in practice to ensure write and read consistency of replicated data. In a quorum system, either a write or read request of the data goes to a set of different servers or locations. More specifically, the data source or writer sends replicas of the data to all the nodes but only waits for the response from a subset of nodes, which is known as the *write quorum* $\mathcal{W}$. Similarly, the client or reader fetches the data from a possibly different subset of nodes, which is called the *read quorum* $\mathcal{R}$. In order to guarantee strict consistency that every read operation returns the most recent written content, a traditional quorum system requires the write quorum $\mathcal{W}$ and read quorum $\mathcal{R}$ to overlap in at least one element. This is known as a *strict quorum*. When a quorum is randomly selected by the writer and reader, a strict quorum requires that the write quorum size w and read quorum size r satisfy $w + r > n$, where n is the number of servers/nodes in the system.

However, as the write or read operation to a set of nodes experiences varying random delays, consistency of the storage system comes at the price of delay. It has been shown that a non-strict or partial quorum of reduced size is widely used in practice because of the latency benefit despite a minor loss in consistency [111]. Amazon's Dynamo database [38], and a variety of subsequent database implementations such as Apache Cassandra [112], use a non-strict quorum as the data replication mechanism in order to maintain a balance between consistency and latency. Since strict consistency is not guaranteed in partial quorum systems, the level of consistency is quantified by data *staleness*. The definition of data staleness falls into two categories: 1) staleness in time [113] [114] and 2) staleness in data version [111]. In [113], a read is considered stale if the value returned was written more than δ time units before the most recent write, where δ is a pre-determined threshold. In a slightly different time-based staleness definition [114], the data is considered fresh if it was generated no more than δ time units ago or it's the most recent written data in the system. On the other hand, [111] measures the staleness by how many versions the value returned by a read lags behind the most recent write. This work described the impact of the quorum size on the trade-off between staleness of data, which is defined as how recent the data version is, and the waiting time for data availability in a probabilistic way.

Our system model as shown in Figure 3.1 with the Earliest-count restart policy represents a special case of the content updating problem in a Dynamo-style storage system. In this case, the writer generates a new version of the data as soon as the writing operation of the previous version is completed. Our system corresponds to the case in which the reading latency is much smaller than the writing latency, and the reader gets the most recent version of the content by reading from a random r nodes in the system. One interesting question we can ask is: at any time t, how old is the content if an interested client reads the data and the reading latency is negligible. Therefore, AoI can be considered as an alternative staleness metric from a strictly time-based perspective. It differs from other staleness metrics in distributed storage systems, since we only take the age of the most recent written content into account.

If the write delays are i.i.d. for each node i and content update j, the $\Delta_i(t)$ processes at each node are statistically identical. The age processes for different read quorum $\mathcal{R}$ are also statistically identical since $\mathcal{R}$ is randomly chosen by the system. Consequently, the expected age of the content returned by a read operation at time t is statistically identical to the time-averaged age of $\Delta(t)$ given by Theorem 3.4.2.

3.4.2 Threshold Restart Policy

The previous counting policy in section 3.4.1 replies on the instantaneous and noiseless delivery feedback from every node. However, this assumption may not be practical when the source is updating a massive amount of nodes, since the feedback can be delayed and it can be costly to collect all of the individual feedback messages. Therefore, this section investigates the stopping scheme when feedback is not available to the source in this section. Since the service times are i.i.d., an intuitive restart policy is to let the source wait for a fixed stopping time τ for every update j. Thus the service interval is constant, $Y_j = \tau$ for all j. Since the service times X_i are i.i.d. for all i, the failure probability is

$$\begin{aligned} q &= \Pr[\min_{i\in\mathcal{R}} X_i > \tau] \\ &= (\Pr[X > \tau])^r \end{aligned}$$

$$= (1 - F_X(\tau))^r\,, \tag{3.58}$$

and the success probability is

$$p = 1 - q = 1 - (1 - F_X(\tau))^r\,. \tag{3.59}$$

Theorem 3.4.12. *For threshold restart policy with τ, the average age at the receiver with access to r nodes is*

$$\Delta = \frac{\tau}{2} + \frac{1}{1 - \big(1 - F_X(\tau)\big)^r} \int_{x=0}^{\tau} \big(1 - F_X(x)\big)^r \mathrm{d}x. \tag{3.60}$$

The proof is provided in the Appendix. For given r and stopping threshold τ, we show in the next two example distribution models that using the CDF $F_X(x)$ to compute the average age as in Theorem 3.4.12 is rather convenient.

3.4.2.1 Shifted Exponential Model

Theorem 3.4.13. *For shifted exponential (λ, c) service time X and a given r, the average age at the receiver is*

$$\Delta_{(\tau,r)} = \frac{\tau}{2} + \frac{1}{\lambda r} + \frac{c}{1 - e^{-\lambda r(\tau - c)}}. \tag{3.61}$$

Proof. We first rewrite $q = \big(1 - F_X(\tau)\big)^r$ for simplicity. For shifted exponential X and $\tau \geq c$,

$$q = \Big(e^{\lambda(\tau - c)}\Big)^r = e^{\lambda r(\tau - c)}. \tag{3.62}$$

It follows from Theorem 3.4.12 that

$$\begin{aligned}
\Delta_{(\tau,r)} &= \frac{\tau}{2} + \frac{1}{1-q}\int_0^{\tau} (1 - F_X(x))^r \mathrm{d}x \\
&= \frac{\tau}{2} + \frac{1}{1-q}\Big(\int_0^{c} 1\,\mathrm{d}x + \int_c^{\tau} e^{-\lambda r(x-c)}\mathrm{d}x\Big) \\
&= \frac{\tau}{2} + \frac{c}{1-q} + \frac{1}{1-q}\frac{1 - e^{\lambda r(\tau - c)}}{\lambda r}
\end{aligned}$$

$$= \frac{\tau}{2} + \frac{c}{1-q} + \frac{1}{\lambda r}. \tag{3.63}$$

Substituting (3.62) back completes the proof. □

Corollary 3.4.14. *For shifted exponential* (λ, c) *service time* X *and a given* r*, the average age at the receiver is upper bounded by*

$$\Delta_{(\tau,r)} \le \bar{\Delta}_{(\tau,r)} = c + \frac{\tau}{2} + \frac{\tau}{\lambda r(\tau - c)}. \tag{3.64}$$

Proof. For $x < 1$, we have the upper bound $e^x \le \frac{1}{1-x}$. Since $\lambda > 0$, $r \ge 1$ and $\tau > c$, the exponent $-\lambda r(\tau - c) < 0$ and it follows from Theorem 3.4.13 that

$$\begin{aligned}
\Delta_{(\tau,r)} &\le \frac{\tau}{2} + \frac{1}{\lambda r} + \frac{c}{1 - \frac{1}{1+\lambda r(\tau-c)}} \\
&= \frac{\tau}{2} + \frac{1}{\lambda r} + \frac{c\,(1 + \lambda r(\tau - c))}{\lambda r(\tau - c)} \\
&= \frac{\tau}{2} + \frac{\tau - c + c + c\lambda r(\tau - c)}{\lambda r(\tau - c)} \\
&= c + \frac{\tau}{2} + \frac{\tau}{\lambda r(\tau - c)}.
\end{aligned} \tag{3.65}$$

□

For a given threshold τ, we observe that the upper bound in Corollary 3.4.14 becomes

$$\bar{\Delta}_{(\tau,r)} = c + \frac{\tau}{2} + O\left(\frac{1}{r}\right), \tag{3.66}$$

which decreases in the order of $O(1/r)$ and is lower bounded by $c + \tau/2$ for large r. That means, having a larger access number r provides at least a decaying average age in the order of $O(1/r)$, even if the source chooses a constant τ which may not be optimal for the value of r. However, the advantage of having more edge nodes access is huge in the beginning, but it gets smaller as r increases.

For a given r, we observe the average age in Theorem 3.4.13 is a convex function of the threshold τ. Simply letting $\mathrm{d}\Delta/\mathrm{d}\tau$, we have the optimal threshold as the following corollary.

Corollary 3.4.15. *For shifted exponential (λ, c) service time X, the age optimal stopping threshold τ^* is given by*

$$\tau^* = c - \frac{1}{\lambda r} \log\Big((1+\lambda cr) - \sqrt{(1+\lambda cr)^2 - 1}\Big). \tag{3.67}$$

Corollary 3.4.16. *For shifted exponential (λ, c) service time X, the minimized average age using optimal stopping threshold τ^* is given by*

$$\Delta^*_{(r)} = \Delta_{(\tau^*, r)} = \frac{c}{2} + \frac{c}{1-\phi(r)} + \frac{1}{\lambda r}\Big(1 - \log\big(\phi(r)\big)\Big), \tag{3.68}$$

where $\phi(r) = (1+\lambda cr) - \sqrt{(1+\lambda cr)^2 - 1}$.

We observe that the optimal threshold in Corollary 3.4.15 is identical to the expected length of the service interval for optimal Earliest-count policy in Corollary 3.4.6. Although instantaneous delivery feedback is exploited in the Earliest-count policy to obtain the optimal waiting counter w^*, the expected waiting time for every update is the same as the optimal waiting time τ^* if feedback is not available. Moreover, τ^* depends only on the distribution of the i.i.d. service time X. This implies that delivery feedback is in fact not so powerful if the distribution of X is known. We'll numerically compare the resulting average age for the two different restart policy in section 3.4.3.

3.4.2.2 Pareto Model

Theorem 3.4.17. *For Pareto (b, ν) service time X and a given r, the average age at the receiver is*

$$\Delta_{(\tau, r)} = \frac{\tau}{2} + \frac{b\nu r - b^{\nu r}\tau^{-\nu r+1}}{(\nu r - 1)(1 - b^{\nu r}\tau^{-\nu r})}. \tag{3.69}$$

Proof. We first rewrite $q = \big(1 - F_X(\tau)\big)^r$ for simplicity. For Pareto X and $\tau \geq b$,

$$q = \left(\frac{b}{\tau}\right)^{\nu r} = b^{\nu r}\tau^{-\nu r}. \tag{3.70}$$

It follows from Theorem 3.4.12 that

$$\begin{aligned}
\Delta_{(\tau,r)} &= \frac{\tau}{2} + \frac{1}{1-q}\int_0^\tau (1-F_X(x))^r \mathrm{d}x \\
&= \frac{\tau}{2} + \frac{1}{1-q}\Big(\int_0^b 1\,\mathrm{d}x + \int_b^\tau \left(\frac{b}{x}\right)^{\nu r}\mathrm{d}x\Big) \\
&= \frac{\tau}{2} + \frac{b}{1-q} + \frac{1}{1-q}\left(\frac{\tau b^{\nu r}\tau^{-\nu r}}{1-\nu r} - \frac{b}{1-\nu r}\right) \\
&= \frac{\tau}{2} + \frac{b}{1-q} + \frac{1}{1-q}\frac{\tau q - b}{1-\nu r} \\
&= \frac{\tau}{2} + \frac{b\nu r - q\tau}{(\nu r-1)(1-q)}. && (3.71)
\end{aligned}$$

Substituting (3.70) into (3.71) completes the proof. □

Corollary 3.4.18. *For Pareto (b,ν) service time X and a given r, the average age at the receiver is upper bounded by*

$$\Delta_{(\tau,r)} \le \bar{\Delta}_{(\tau,r)} = \frac{\tau}{2} + \frac{b}{1-(b/\tau)^{\nu r}} + \frac{\tau}{\nu r - 1}. \tag{3.72}$$

Proof. Since $b \le \tau$, it follows from (3.71) that

$$\begin{aligned}
\Delta_{(\tau,r)} &= \frac{\tau}{2} + \frac{b\nu r - q\tau}{(\nu r-1)(1-q)} \\
&= \frac{\tau}{2} + \frac{b\nu r - b + b - q\tau}{(\nu r-1)(1-q)} \\
&\le \frac{\tau}{2} + \frac{(b\nu r - b) + (\tau - q\tau)}{(\nu r-1)(1-q)} \\
&= \frac{\tau}{2} + \frac{b}{1-q} + \frac{\tau}{\nu r - 1}. && (3.73)
\end{aligned}$$

Substituting (3.70) back to (3.73) completes the proof. □

Rewriting Corollary 3.4.18 in series expansion yields

$$\begin{aligned}
\bar{\Delta}_{(\tau,r)} &= \frac{\tau}{2} + \frac{b}{1-\left(1+\nu r\log(b/\tau) + \frac{1}{2}\nu r^2\log^2(b/\tau) + \dots\right)} + \frac{\tau}{\nu r - 1} \\
&= \frac{\tau}{2} + \frac{b}{-\nu r\log(b/\tau) - \frac{1}{2}\nu r^2 \log^2(b/\tau) - \dots} + \frac{\tau}{\nu r - 1} \\
&\le \frac{\tau}{2} + \frac{b}{-\nu r\log(b/\tau)} + \frac{\tau}{\nu r - 1}
\end{aligned}$$

$$= \frac{\tau}{2} + O\left(\frac{1}{r}\right) \tag{3.74}$$

Similar to the shifted exponential model, we observe that the upper bound with Pareto model also decreases in the order of $O(1/r)$.

For a given r, we take the second derivative of Theorem 3.4.13 and yields

$$\begin{aligned}\frac{\mathrm{d}^2\Delta}{\mathrm{d}\tau^2} = &\frac{\nu^2 r^2 b^{\nu r+1}\tau^{\nu r-2}\big(\nu r(b^{\nu r}+\tau^{\nu r})+\tau^{\nu r}-b^{\nu r}\big)}{(\nu r-1)\left(\tau^{\nu r}-b^{\nu r}\right)^3}\\ &+\frac{\nu r b^{\nu r}\tau^{\nu r-1}\big((\nu r+1)b^{\nu r}+(\nu r-1)\tau^{\nu r}\big)}{\left(\tau^{\nu r}-b^{\nu r}\right)^3}.\end{aligned} \tag{3.75}$$

Since $\nu > 1$, $r > 1$ and $0 < b < \tau$, we note that $\mathrm{d}^2\Delta/\mathrm{d}\tau^2 > 0$. Therefore, the average age in Theorem 3.4.13 is a convex function of the threshold τ. Setting $\mathrm{d}\Delta/\mathrm{d}\tau = 0$, we obtain the optimal threshold in the following corollary.

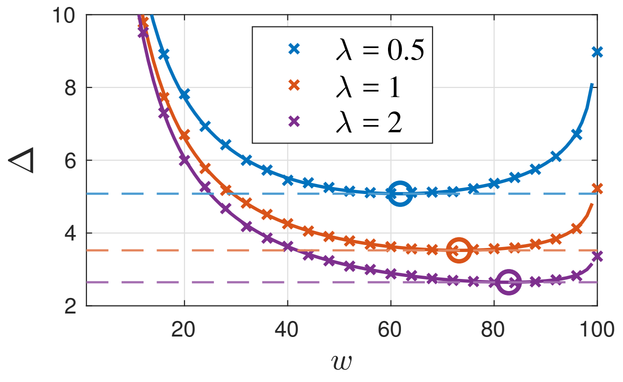
Corollary 3.4.19. *For Pareto (b, ν) service time X, the age optimal stopping threshold τ^* is the root of the following equation*

$$(\nu r-1)\tau^{2\nu r} - 2\nu^2 r^2 b^{\nu r+1}\tau^{\nu r-1} + (\nu r+1)b^{2\nu r} = 0. \tag{3.76}$$

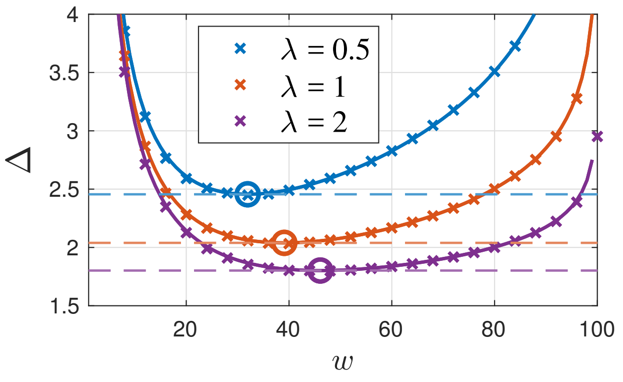
For threshold restart policy, we observe that the access number r is always coupled with the exponent, which is λ in the shifted exponential model as shown in Theorem 3.4.17, and ν in Pareto model as shown in Theorem 3.4.17. Since the exponent captures how fast the tail of the distribution decays, scaling the exponent by r is equivalent to having a service time whose tail decays r times faster. This is mainly because the receiver has access to r nodes and always selects the freshest update from all r nodes.

3.4.3 Numerical Evaluation

In this section, we numerically evaluate how the restart policy affects the average age metric. We start from the evaluations of our approximations of the average age and the corresponding optimization for both the Earliest-count policy and the threshold policy. In Figure 3.3, we demonstrate the average age with shifted exponential service time using both policies. We set the total number of nodes as $n = 100$, and the access number



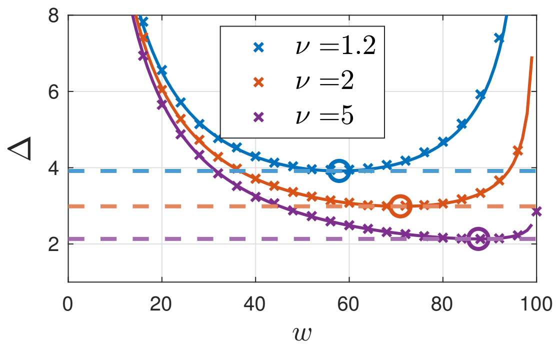
(a) access number $r = 1$, $n = 100$.



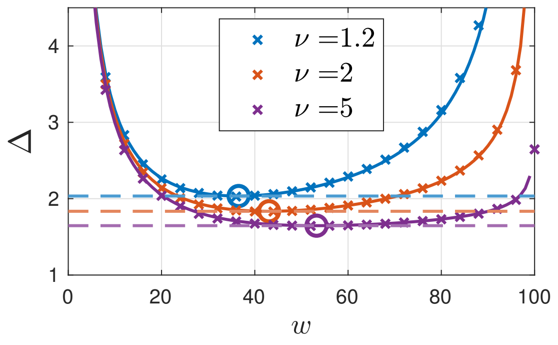
(b) access number $r = 5$, $n = 100$.

Figure 3.3: Comparison of the average age between Earliest-count (EC) and threshold policies using shifted exponential service time with $c = 1$. $\times$ marks the average age as a function of w for EC policy. Solid line marks the approximate average age, and $\circ$ marks the minimized approximate age $\hat{\Delta}$. Dashed line marks the average age using the optimal threshold restart policy.

$r = 1$ in Figure 3.3a, and $r = 5$ in Figure 3.3b. For the Earliest-count policy, we plot the average age Δ versus different counter w using symbol $\times$. Each $\times$ symbol corresponds to the experimental time-average age for a given exponential rate λ. The analytical approximation in Theorem 3.4.4 is drawn using solid line, and the near-optimal w^* obtained by Corollary 3.4.10 is marked with $\circ$. We observe the approximation is barely distinguishable from the real experimental average age, and the near-optimal w^* is also very close to the true optimal point that minimizes the average age as shown in both figures. By looking at the three curves in a single figure, the optimal w^* increases as the exponential rate λ increases, meaning that the source should waits longer for each multicast transmission if the average service time is longer. Comparing across Figure



(a) access number $r = 1$, $n = 100$.



(b) access number $r = 5$, $n = 100$.

Figure 3.4: Comparison of the average age between Earliest-count (EC) and threshold policies using Pareto service time with $b = 1$. $\times$ marks the average age as a function of w for EC policy. Solid line marks the approximate average age, and $\circ$ marks the minimized approximate age $\hat{\Delta}$. Dash line marks the average age using the optimal threshold restart policy.

3.3a and Figure 3.3b with different access number r, we observe that the average age decreases as r increases. More importantly, the optimal w^* moves to the left significantly. For example, the receiver has access to a single node in Figure 3.3a. In this case, in order to minimize the average age at the receiver, the source should consider the transmission of the current update to be completed once the update is delivered to around 60 out of 100 nodes when $\lambda = 0.5$. However, if the receiver has access to $r = 5$ nodes as shown in Figure 3.3b, it is best for the source to only wait for the deliveries to around 30 nodes for each multicast transmission.

For the threshold policy, we plot the average age using the optimized threshold τ^* in Theorem 3.4.15 as a horizontal dash line together with the Earliest-count policy with the same color in $r = 1$ in Figure 3.3a and 3.3b. As the bottom of the bell shape curve

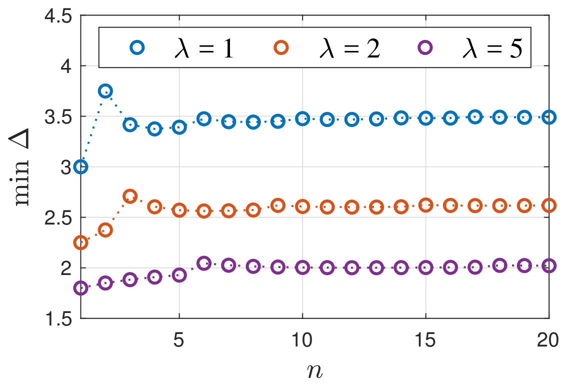
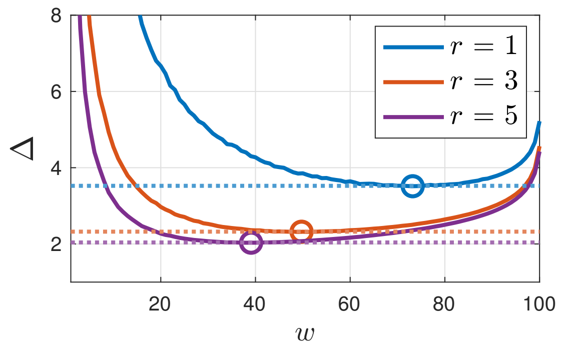


Figure 3.5: Minimum average age obtained by optimizing counter w^* for different node numbers n in the system using shifted exponential service model with $c = 1$ and access number $r = 1$.

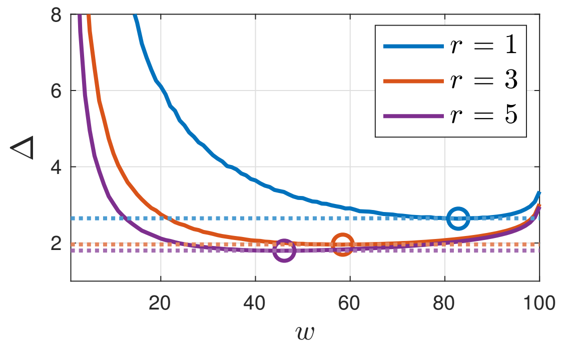
for the Earliest-count policy touches the horizontal dash line for the threshold policy in every case, we conclude that both policy lead to non-distinguishable average age performance if both policies are carefully optimized. That is, instead of counting the number of deliveries based on the instantaneous feedback, the source can achieve similar age performance by simply setting a deadline in time for each multicast transmission without the feedback.

Figure 3.4 demonstrates similar results as Figure 3.3 using Pareto service time with $b = 1$ and three different choices of ν. Although the service time is now with heavy tail, we note that most of our observations in Figure 3.3 are also applicable to Figure 3.4. It's interesting to look at the average curve corresponding $\nu = 5$ in Figure 3.4b. Although choosing $w \approx 52$ minimizes the average age, the age curve is mostly flat in the range of $[40, 70]$, with at most 0.1 units difference. If the timeliness requirement of the system is not very strict, choosing any value of w in $[40, 70]$ is good enough because the resulting average age is very close to the true minimum.

Since the numerical results in Figures 3.3 and 3.4 use a large $n = 100$, we now illustrate how the average age varies for smaller numbers of n. Figure 3.5 depicts the optimized average age for the total number of nodes $n \in [1, 20]$ using the Earliest-count policy. In this example, we choose the service time model to be shifted exponential with $c = 1$ and three different λ, and the access number $r = 1$. For every n, a corresponding age-minimized w^* is obtained by Corollary 3.4.10. When there is only one node in the



(a) $\lambda = 1$, $n = 100$.



(b) $\lambda = 2$, $n = 100$.

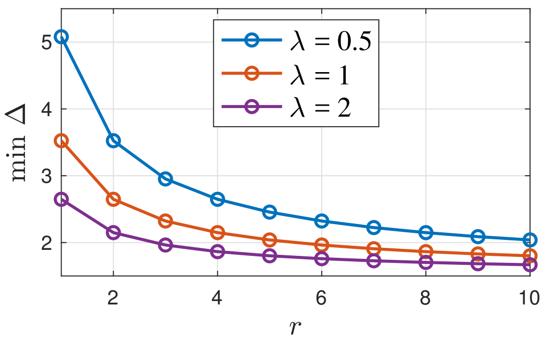
Figure 3.6: Comparison of the average age among receivers with different access number r using shifted exponential service time with $c = 1$. $\circ$ marks the minimized average age. Dash line marks the average age using the optimal threshold restart policy.

system, we observe that the average age is exactly at $3(c + 1/\lambda)/2$. In all three cases of λ, the average age fluctuates for small n since the earliest w scheme is a heuristic by waiting for an integer number of users w. As the number of users n grows, the average age converges rapidly, which matches our conclusion in Theorem 3.4.8 that the approximate average age depends only on λ and c when n is large enough and r is fixed.

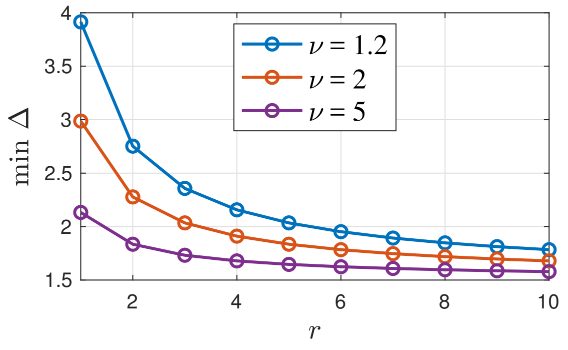
In a heterogeneous network with many receivers, the access numbers r for different receivers may not be identical. Thus, it's very likely that the source may optimize its multicast strategy based on access number r_1 but a particular receiver will have a different access number r_2. Motivated by the observation in Figure 3.3 and 3.4 that the average age can be near optimal for a wide range of w, it's interesting to evaluate the robustness of the system, for instance, how the optimal policy for a particular access number r_1 performs for other values of r_2. Figure 3.6 compares the average age as a function of w for different values of r with the shifted exponential service distribution to be. In Figure 3.6a, we observe that the optimal counter $w^* = 40$ for access number r, meaning that the source should wait for 40 out of 100 deliveries for each multicast transmission. The resulting average age is about 2 for this receiver with $r = 5$. If the source chooses this as the restart policy, then the average age of a receiver with $r = 3$ will not be optimal since its corresponding optimal $w = 50$. However, the age difference is tiny since the age curve for $r = 3$ stays almost the same for $w \in [40, 60]$. Instead, if the receiver has access to only $r = 1$ node but the source chooses $w = 40$, the resulting average age will be much larger than its minimum average age that occurs at $w = 73$. That means, if the source is not certain about the access number of the receiver, or the access number of a receiver varies randomly, choosing the optimal w for Earliest-count policy or the optimal τ for threshold policy may not be a winning choice. Instead, we observe in Figure 3.6a that choosing $w \approx 60$ can be the best solution if the access number varies among all three values of r, since the average age at $w = 60$ is very close to the minimum age for any of the three curves.

We observe in previous figures that the average age at the receiver decreases as the access number r increases due to parallelism and data replication. Figure 3.7 illustrates the optimized average age for different access number r. For each r, we use the threshold policy and obtain the minimum average age with optimized $\tau^*(r)$. In Figure 3.7a, the shifted exponential service time with different λ is used. The average age decreases quickly when r is small, but it also saturates quickly and the benefit of having larger access number r is very minor if the source optimizes the updating strategy for the given r. For example, the largest average age reduction occurs at the increase from $r = 1$ to $r = 2$, and the average age curve for $\lambda = 2$ decays very slowly when $r > 5$. Figure 3.7b demonstrates similar results for Pareto service time with different ν. From an engineering perspective, larger access number essentially means more connectivity resources and power usage for both the end-user equipment and the edge nodes. Although more edge access helps maintaining information freshness, our observation here shows that having too much access is in fact unnecessary, since the benefit of larger r degrades quickly.

Figure 3.8 demonstrates the optimized average age $\Delta^*(r)$ for every r, together with



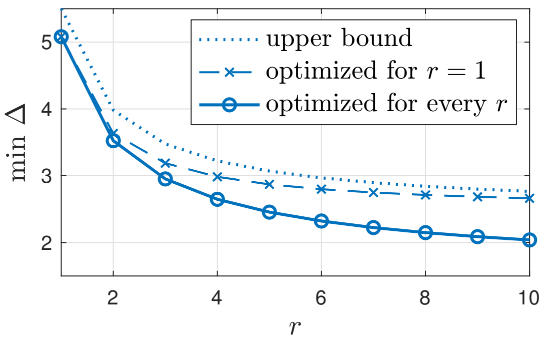
(a) shifted exponential model with $c = 1$.



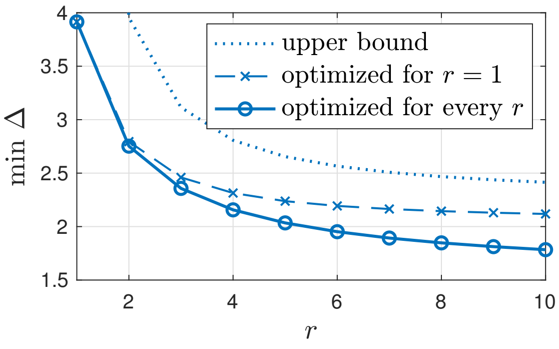
(b) Pareto model with $b = 1$.

Figure 3.7: Minimum average age obtained by optimizing the threshold τ^* as a funciton of the access number r.

another average age curve corresponding to a sub-optimal threshold τ for all r. For the sub-optimal policy, we assume the source doesn't know the access number of a receiver and assume the worst case $r = 1$ conservatively. Thus, the source chooses a threshold policy with $\tau^*(r = 1)$, which is suboptimal for $r > 1$. In Figure 3.8a, we also show the upper bound on the average age using the threshold policy that is optimized for $r = 1$, assuming the service time is shifted exponentially distributed in Corollary 3.4.14. We note that the upper bound decays in the order of $O(1/r)$. The curve for optimized average age decays slightly faster than that for the sub-optimal scheme. Figure 3.8b shows the similar comparison and the upper bound in Corollary 3.4.18 with Pareto service time.



(a) shifted exponential model with $c = 1$ and $\lambda = 1$.



(b) Pareto model with $b = 1$ and $\nu = 1.2$.

Figure 3.8: Average age as a function of the access number r using different threshold policies.

3.4.4 Appendix

Proof of Lemma 3.3.2

We note the sequence $Y_j, \ldots, Y_{j+M_l-1}$ and the number of summation terms M_l can be dependent. Since M_l is geometric, the event $M_l = m$ indicates a sequence of $m - 1$ consecutive failures followed by a success. Thus, $Y_{j'}$ is identical to Y_F for $j' =\in \{j, \ldots, j + M_l - 2\}$ and the last variable in the sequence Y_{j+M_l-1} is identical to Y_S. This implies

$$\begin{aligned} \mathrm{E}[W] &= \sum_{m=1}^{\infty} P_M(m)\,\mathrm{E}\left[\sum_{i=1}^{m} Y_i \middle| M = m\right] \\ &= \sum_{m=1}^{\infty} P_M(m)\Big((m-1)\,\mathrm{E}[Y_F] + \mathrm{E}[Y_S]\Big) \\ &= \mathrm{E}[Y_F](\mathrm{E}[M] - 1) + \mathrm{E}[Y_S]. \end{aligned} \tag{3.77}$$

Substituting (3.5) into (3.77) yields

$$\begin{aligned} \mathrm{E}[W] &= \frac{1}{p}\big((1-p)\,\mathrm{E}[Y_F] + p\,\mathrm{E}[Y_S]\big) \\ &= \mathrm{E}[M]\,\mathrm{E}[Y]. \end{aligned} \tag{3.78}$$

For the second moment, we write $\mathrm{E}\big[W^2\big]$ in total expectation as

$$\begin{aligned} \mathrm{E}\big[W^2\big] &= \sum_{m=1}^{\infty} P_M(m)\,\mathrm{E}\left[\Big(\sum_{i=1}^{m} Y_i\Big)^2 \Big| M = m\right] \\ &= \sum_{m=1}^{\infty} P_M(m)\left(\mathrm{Var}\left[\sum_{i=1}^{m} Y_i\right] + \left(\mathrm{E}\left[\sum_{i=1}^{m} Y_i\right]\right)^2\right). \end{aligned} \tag{3.79}$$

Since the random variables Y_i are independent, we let

$$\begin{aligned} \eta_1 &= \sum_{m=1}^{\infty} P_M(m)\,\mathrm{Var}\left[\sum_{i=1}^{m} Y_i\right] \\ &= \sum_{m=1}^{\infty} P_M(m)\Big((m-1)\,\mathrm{Var}[Y_F] + \mathrm{Var}[Y_S]\Big) \\ &= \mathrm{Var}[Y_F]\Big(\,\mathrm{E}[M] - 1\Big) + \mathrm{Var}[Y_S]. \end{aligned} \tag{3.80}$$

Similarly, we have

$$\begin{aligned} \eta_2 &= \sum_{m=1}^{\infty} P_M(m)\left(\mathrm{E}\left[\sum_{i=1}^{m} Y_i\right]\right)^2 \\ &= \sum_{m=1}^{\infty} P_M(m)\Big((m-1)\,\mathrm{E}[Y_F] + \mathrm{E}[Y_S]\Big)^2 \\ &= \big(\mathrm{E}\big[M^2\big] - 2\,\mathrm{E}[M] + 1\big)\,(\mathrm{E}[Y_F])^2 \\ &\qquad + 2(\mathrm{E}[M] - 1)\,\mathrm{E}[Y_F]\,\mathrm{E}[Y_S] + (\mathrm{E}[Y_S])^2. \end{aligned} \tag{3.81}$$

The claim follows by substituting (3.80) and (3.81) in (3.79).

Proof of Theorem 3.4.4

For shifted exponential r.v. X, the ratio of the first moment to the second moment of the order statistic $X_{w:n}$ is

$$\begin{aligned}\frac{\mu_{w:n}^{(2)}}{\mu_{w:n}} &= \mu_{w:n} + \frac{\sigma_{w:n}^2}{\mu_{w:n}} \\ &= \mu_{w:n} + \frac{H_{n^2} - H_{(n-w)^2}}{2\lambda^2 c + 2\lambda(H_n - H_{n-w})}. \qquad (3.82)\end{aligned}$$

As $n \to \infty$, the asymptotic expansion for the harmonic number is

$$H_n = \log n + \gamma + O\left(\frac{1}{n}\right). \qquad (3.83)$$

Similarly, if we let $w = \alpha n = O(n)$,

$$\begin{aligned}H_{n-w} &= \log(n-w) + \gamma + O\left(\frac{1}{n-w}\right) \\ &= \log(n-w) + \gamma + O\left(\frac{1}{n}\right) \qquad (3.84)\end{aligned}$$

Thus,

$$H_n - H_{n-w} = \log n - \log(n-w) + O\left(\frac{1}{n}\right). \qquad (3.85)$$

Thus, we can use the asymptotic as a tight approximation $H_n - H_{n-w} \approx \log n - \log(n-w)$ as $n \to \infty$. Note that the sequence H_{n^2} is monotonically increasing and $\lim_{n\to\infty} H_{n^2} = \pi^2/6$, thus $H_{n^2} - H_{(n-w)^2} \leq \pi^2/6$ is negligible and

$$\frac{H_{n^2} - H_{(n-w)^2}}{2\lambda^2 c + 2\lambda(H_n - H_{n-w})} = o(1). \qquad (3.86)$$

It follows from (3.86) and (3.18a) that

$$\begin{aligned}\mu_{w:n} &= c + \frac{1}{\lambda}(H_n - H_{n-w}) \\ &= c + \frac{1}{\lambda}(\log n - \log(n-w)) + O\left(\frac{1}{n}\right)\end{aligned}$$

$$= c + \frac{1}{\lambda}\left(\log\frac{n}{n-w}\right) + O\left(\frac{1}{n}\right). \tag{3.87}$$

$$\frac{\mu^{(2)}_{w:n}}{\mu_{w:n}} = \mu_{w:n} + o(1)$$

$$= c + \frac{1}{\lambda}\left(\log\frac{n}{n-w}\right) + o(1). \tag{3.88}$$

In Theorem 3.4.2, the binomial coefficient is upper bounded by $\binom{n}{k} \le \frac{n^k}{k!}$. For small k, we use the upper bound as an approximation

$$\binom{n}{k} \approx \frac{n^k}{k!}. \tag{3.89}$$

Substituting (3.86) and (3.88) into Theorem 3.4.2 we have the following results. For $w + r > n$, Theorem 3.4.2 is then rewritten as

$$\Delta = \sum_{i=1}^{n-r+1} \mu_{i:n}\frac{\binom{n-i}{r-1}}{\binom{n}{r}} + \frac{1}{2}\frac{\mu^{(2)}_{w:n}}{\mu_{w:n}} \tag{3.90a}$$

$$= \sum_{i=1}^{n-r+1} \mu_{i:n}\frac{r}{n}\left(\prod_{j=0}^{r-2}\frac{n-i-j}{n-1-j}\right) + \frac{1}{2}\mu_{w:n} + o(1) \tag{3.90b}$$

$$\approx \sum_{i=1}^{n-r+1} \mu_{i:n}\frac{(n-i)^{r-1}r}{n^r} + \frac{1}{2}\mu_{w:n} \tag{3.90c}$$

$$\approx \sum_{i=1}^{n-r+1}\left(\frac{1}{\lambda}\log\left(\frac{n}{n-i}\right) + c\right)\frac{(n-i)^{r-1}r}{n^r} + \frac{1}{2\lambda}\log\left(\frac{n}{n-w}\right) + \frac{c}{2} \tag{3.90d}$$

$$\approx r\int_{x=0}^{z=\frac{n-r}{n}}\left(\frac{1}{\lambda}\log\left(\frac{1}{1-x}\right) + c\right)(1-x)^{r-1}\mathrm{d}x + \frac{1}{2\lambda}\log\left(\frac{1}{1-\alpha}\right) + \frac{c}{2} \tag{3.90e}$$

$$= \frac{\Big(1 - (1-z)^r(1 - r\log(1-z))\Big)}{\lambda r} + c\left(1 - (1-z)^r\right) + \frac{1}{2\lambda}\log\left(\frac{1}{1-\alpha}\right) + \frac{c}{2} \tag{3.90f}$$

$$= \left(\frac{1}{\lambda r} + c\right)\left(1 - (1-z)^r\right) - \frac{1}{\lambda}(1-z)^r\log\frac{1}{(1-z)} + \frac{1}{2\lambda}\log\left(\frac{1}{1-\alpha}\right) + \frac{c}{2}. \tag{3.90g}$$

In (3.90e), we denote $\alpha = w/n$ and approximate the sum $1/n\sum_{i=1}^{w} f(i)$ by the integral $\int_{x=0}^{\alpha} f(x)\mathrm{d}x$ and this approximation is tight since

$$\left|\int_{x=0}^{\alpha=w/n} f(x)\mathrm{d}x - \frac{1}{n}\sum_{i=1}^{w} f(i)\right| \le O\left(\frac{1}{n^2}\right). \tag{3.91}$$

Similarly, for $w + r \leq n$,

$$\begin{aligned}
\Delta &= \sum_{i=1}^{w} \mu_{i:n} \frac{\binom{n-i}{r-1}}{\binom{n}{r} - \binom{n-w}{r}} + \frac{1}{2} \frac{\binom{n}{r} + \binom{n-w}{r}}{\binom{n}{r} - \binom{n-w}{r}} \mu_{w:n} + \frac{1}{2} \frac{\sigma^2_{w:n}}{\mu_{w:n}} && \text{(3.92a)} \\
&= \sum_{i=1}^{w} \mu_{i:n} \frac{r \prod_{j=0}^{r-2}(n-i-j)}{\prod_{j=0}^{r-1}(n-j) - \prod_{j=0}^{r-1}(n-w-j)} \\
&\quad + \frac{1}{2} \frac{\prod_{j=0}^{r-1}(n-j) + \prod_{j=0}^{r-1}(n-w-j)}{\prod_{j=0}^{r-1}(n-j) - \prod_{j=0}^{r-1}(n-w-j)} \mu_{w:n} + o(1) && \text{(3.92b)} \\
&\approx \sum_{i=1}^{w} \mu_{i:n} \frac{(n-i)^{r-1} r}{n^r - (n-w)^r} + \frac{n^r + (n-w)^r}{2(n^r - (n-w)^r)} \mu_{w:n} && \text{(3.92c)} \\
&\approx \sum_{i=1}^{w} \left(\frac{1}{\lambda} \log \left(\frac{n}{n-i} \right) + c \right) \frac{(n-i)^{r-1} r}{n^r - (n-w)^r} + \frac{n^r + (n-w)^r}{2(n^r - (n-w)^r)} \left(\frac{1}{\lambda} \log \left(\frac{n}{n-w} \right) + c \right) && \text{(3.92d)} \\
&\approx \int_{x=0}^{\alpha} \left(\frac{1}{\lambda} \log \left(\frac{1}{1-x} \right) + c \right) \frac{(1-x)^{r-1} r}{1 - (1-\alpha)^r} \mathrm{d}x + \frac{1 + (1-\alpha)^r}{2(1 - (1-\alpha)^r)} \left(\frac{1}{\lambda} \log \left(\frac{1}{1-\alpha} \right) + c \right) && \text{(3.92e)} \\
&= \frac{1}{\lambda c} - \frac{(1-\alpha)^r}{\lambda(1 - (1-\alpha)^r)} \log \left(\frac{1}{1-\alpha} \right) + c \\
&\quad + \frac{(1 + (1-\alpha)^r)}{2\lambda(1 - (1-\alpha)^r)} \log \left(\frac{1}{1-\alpha} \right) + \frac{c(1 + (1-\alpha)^r)}{2(1 - (1-\alpha)^r)} && \text{(3.92f)} \\
&= \frac{1}{\lambda r} + \frac{1}{2\lambda} \log \left(\frac{1}{1-\alpha} \right) + c + \frac{c(1 + (1-\alpha)^r)}{2(1 - (1-\alpha)^r)}. && \text{(3.92g)}
\end{aligned}$$

To obtain (3.92c), we use the limit in (3.86) as an approximate and substitute it back to Theorem 3.4.2. To simplify the expression we further denote $\beta = 1 - \alpha$ and $\theta = 1 - z$ to complete the proof.

Proof of Corollary 3.4.5

Since Theorem 3.4.4 contains two intervals, the optimal β^* can occur in either interval. For the first case where $w + r > n$ thus $\beta < r/n$, we taking the derivative of $\hat{\Delta}_1$ in Theorem 3.4.4 and have

$$\frac{\mathrm{d}\hat{\Delta}_1}{\mathrm{d}\beta} = -\frac{\beta^{r-1}(\lambda c r - r \log \beta)}{\lambda} - \frac{1}{2\lambda\beta}. \tag{3.93}$$

Since $0 < \beta < 1$, $\mathrm{d}\hat{\Delta}_1/\mathrm{d}\beta < 0$ and $\hat{\Delta}_1$ is a monotonic decreasing function of β. Therefore, the minimum of $\hat{\Delta}_1(\beta)$ occurs as $\beta \to r/n$, which implies $w = n - r + 1$ since w is an integer.

For the second case with $w + r \leq n$ thus $\beta \geq r/n$, we obtain the derivatives of $\hat{\Delta}_2$ as follows

$$\frac{\mathrm{d}\hat{\Delta}_2}{\mathrm{d}\beta} = -\frac{\beta^{2r} - 2(\lambda cr + 1)\beta^r + 1}{2\lambda\beta(1-\beta^r)^2} \tag{3.94}$$

$$\frac{\mathrm{d}^2\hat{\Delta}_2}{\mathrm{d}\beta^2} = \frac{2\lambda cr\beta^r\big((r+1)\beta^r + r - 1\big) + (1-\beta^r)^3}{2\lambda\beta^2(1-\beta^r)^3} \tag{3.95}$$

Since $r \geq 1$ and $\beta < 1$ in (3.95), $\mathrm{d}^2\hat{\Delta}_2/\mathrm{d}\beta^2 > 0$ and thus $\hat{\Delta}_2(\beta)$ is convex. Setting $\mathrm{d}\hat{\Delta}_1/\mathrm{d}\beta = 0$ we have

$$\beta^{2r} - 2(\lambda cr + 1)\beta^r + 1 = 0. \tag{3.96}$$

We define $\eta = \beta^r$, and the solution to (3.96) for $\eta \in (0, 1)$ is given by

$$\eta^* = (\lambda cr + 1) - \sqrt{(\lambda cr + 1)^2 - 1}. \tag{3.97}$$

Since it is required that $\beta \geq r/n$, the minimum point exists only if

$$(\lambda cr + 1) - \sqrt{(\lambda cr + 1)^2 - 1} \geq \left(\frac{r}{n}\right)^r. \tag{3.98}$$

Otherwise, the function $\hat{\Delta}_2(\beta)$ is monotonically increasing in β and the minimum of $\hat{\Delta}_2$ occurs at $w = n - r$.

Proof of Corollary 3.4.6

For the expected service time, we let $\beta = 1 - w/n$ and rewrite (3.88) as follows

$$\begin{aligned}\mathrm{E}[Y] &= c + \frac{1}{\lambda}\log\frac{1}{\beta^*}\\ &= c + \frac{1}{\lambda}\log\frac{1}{(\eta^*)^{1/r}}\end{aligned}$$

$$
\begin{aligned}
&= c - \frac{1}{\lambda r}\log \eta^* \\
&= c - \frac{1}{\lambda r}\log\Big((\lambda c r + 1) - \sqrt{(\lambda c r+1)^2 - 1}\Big). \qquad (3.99)
\end{aligned}
$$

Proof of Theorem 3.4.8

For Pareto (ν, b) r.v. X, we approximate the ratio of the first moment to the second moment of the order statistic using the same techniques as in Lemma 3.4.7. It follows from (3.22) that

$$
\begin{aligned}
\frac{\mu^{(2)}_{w:n}}{\mu_{w:n}} &= b\,\frac{\Gamma(n-k+1-\frac{2}{\nu})\Gamma(n+1-\frac{1}{\nu})}{\Gamma(n+1-\frac{2}{\nu})\Gamma(n-k+1-\frac{1}{\nu})} \\
&\approx b\,\frac{\sqrt{(n-k-\frac{2}{\nu})(n-\frac{1}{\nu})}\,(n-k-\frac{2}{\nu})^{n-k-\frac{2}{\nu}}(n-\frac{1}{\nu})^{n-\frac{1}{\nu}}}{\sqrt{(n-\frac{2}{\nu})(n-k-\frac{1}{\nu})}\,(n-k-\frac{1}{\nu})^{n-k-\frac{1}{\nu}}(n-\frac{2}{\nu})^{n-\frac{2}{\nu}}} \\
&\approx b\,\frac{\sqrt{(n-k)n}}{\sqrt{n(n-k)}}\,\frac{(n-k)^{n-k-\frac{2}{\nu}}n^{n-\frac{1}{\nu}}}{(n-k)^{n-k-\frac{1}{\nu}}n^{n-\frac{2}{\nu}}} \\
&= b\,\frac{n^{\frac{1}{\nu}}}{(n-k)^{\frac{1}{\nu}}} \\
&= b(1-\alpha)^{-\frac{1}{\nu}} = \mu_{w:n}. \qquad (3.100)
\end{aligned}
$$

That is, we can approximate the ratio $\mu^{(2)}_{w:n}/\mu_{w:n}$ simply by the first moment of the order statistic $\mu_{w:n}$ and the ratio

$$
\lim_{n\to\infty}\frac{\sigma^2_{w:n}}{\mu_{w:n}} = 0. \qquad (3.101)
$$

Let's substitute (3.100) and Lemma 3.4.7 into Theorem 3.4.2, and approximate the binomial coefficient by $\binom{n}{k} \approx \frac{n^k}{k!}$. For $w + r > n$, Theorem 3.4.2 is then rewritten as

$$
\begin{aligned}
\Delta &\approx \sum_{i=1}^{n-r+1} \mu_{i:n}\frac{(n-i)^{r-1}r}{n^r} + \frac{1}{2}\mu_{w:n} \qquad (3.102a)\\
&\approx \sum_{i=1}^{n-r+1} b\,\frac{n^{\frac{1}{\nu}}}{(n-i)^{\frac{1}{\nu}}}\frac{(n-i)^{r-1}r}{n^r} + \frac{b}{2}\,\frac{n^{\frac{1}{\nu}}}{(n-w)^{\frac{1}{\nu}}} \qquad (3.102b)\\
&\approx br\int_{x=0}^{z=\frac{n-r}{n}} (1-x)^{r-\frac{1}{\nu}-1}\mathrm{d}x + \frac{b}{2}\,\frac{n^{\frac{1}{\nu}}}{(n-w)^{\frac{1}{\nu}}} \qquad (3.102c)
\end{aligned}
$$

$$= \frac{br}{r-\frac{1}{\nu}}\left(1-(1-z)^{r-\frac{1}{\nu}}\right)+\frac{b}{2}(1-\alpha)^{-\frac{1}{\nu}}. \tag{3.102d}$$

For $w+r \leq n$,

$$\begin{aligned}
\Delta &\approx \sum_{i=1}^{w} \mu_{i:n}\frac{(n-i)^{r-1}r}{n^r-(n-w)^r}+\frac{n^r+(n-w)^r}{2(n^r-(n-w)^r)}\mu_{w:n} &(3.103a)\\
&\approx \sum_{i=1}^{w} b\,\frac{n^{\frac{1}{\nu}}}{(n-i)^{\frac{1}{\nu}}}\frac{(n-i)^{r-1}r}{n^r-(n-w)^r}+\frac{b\,(n^r+(n-w)^r)}{2(n^r-(n-w)^r)}\frac{n^{\frac{1}{\nu}}}{(n-w)^{\frac{1}{\nu}}} &(3.103b)\\
&\approx \frac{br}{(1-(1-\alpha)^r)}\int_{x=0}^{\alpha=\frac{w}{n}}\left((1-x)^{r-\frac{1}{\nu}-1}\right)\mathrm{d}x+\frac{b\,(n^r+(n-w)^r)}{2(n^r-(n-w)^r)}(1-\alpha)^{-\frac{1}{\nu}} &(3.103c)\\
&= \frac{br}{(r-\frac{1}{\nu})\,(1-(1-\alpha)^r)}\left(1-(1-\alpha)^{r-\frac{1}{\nu}}\right)+\frac{b\,(1+(1-\alpha)^r)}{2\,(1-(1-\alpha)^r)}(1-\alpha)^{-\frac{1}{\nu}}. &(3.103d)
\end{aligned}$$

To obtain (3.103a), we use the limit in (3.101) as an approximate and substitute it back to Theorem 3.4.2.

Proof of Corollary 3.4.10

Since Theorem 3.4.8 contains two intervals, the optimal β^* can occur in either interval. For the first case where $w+r>n$ thus $\beta<r/n$, we taking the derivative of $\hat{\Delta}_1$ in Theorem 3.4.8 and have

$$\frac{\mathrm{d}\hat{\Delta}_1}{\mathrm{d}\beta} = -\frac{b\beta^{-1-\frac{1}{\nu}}}{2}\left(\frac{1}{\nu}+2r\beta^r\right). \tag{3.104}$$

For $\nu>1$, $\beta\in(0,1)$, $b>0$ and $r>1$, the derivative in (3.104) is strictly negative. Thus the approximation $\hat{\Delta}$ is monotonically decreasing in β. Therefore, the minimum of $\hat{\Delta}_1(\beta)$ occurs as $\beta\to r/n$, which implies $w=n-r+1$ since w is an integer.

For the second case with $w+r\leq n$, we have $\beta\geq r/n$. We obtain the derivatives of $\hat{\Delta}_2$ as follows

$$\frac{\mathrm{d}\hat{\Delta}_2}{\mathrm{d}\beta} = \frac{b\beta^{-1-\frac{1}{\nu}}\left(\frac{1}{\nu^2}(\beta^{2r}-1)-2r^2\beta^{r+\frac{1}{\nu}}+\frac{r}{\nu}(\beta^{2r}+1)\right)}{2\,(1-\beta^r)^2\,(r-\frac{1}{\nu})}. \tag{3.105}$$

Then we let the derivative w.r.t β to be zero, i.e.,

$$\left(\frac{1}{\nu^2}+\frac{r}{\nu}\right)\beta^{2r}-2r^2\beta^{r+\frac{1}{\nu}}+\frac{r}{\nu}-\frac{1}{\nu^2}=0. \tag{3.106}$$

Multiplying both sides by ν^2 completes the proof.

Proof of Theorem 3.4.12

Here we denote $\underline{X}=\min_{i\in\mathcal{R}} X_i$ for convenience and Corollary 3.3.4 can be rewritten as

$$\Delta=\frac{\tau(1+q)}{2(1-q)}+\mathrm{E}[\underline{X}|\underline{X}\leq\tau], \tag{3.107}$$

where q is given by (3.58) and

$$\begin{aligned}\mathrm{E}[\underline{X}|\underline{X}\leq\tau]&=\int_{x=0}^{\infty}xf_{\underline{X}|\underline{X}\leq\tau}(x)\mathrm{d}x\\&=\frac{1}{\Pr[\underline{X}\leq\tau]}\int_{x=0}^{\tau}xf_{\underline{X}}(x)\mathrm{d}x\\&=\frac{1}{p}\int_{x=0}^{\tau}xf_{\underline{X}}(x)\mathrm{d}x.\end{aligned} \tag{3.108}$$

We note that $\underline{X}$ is the minimum of r i.i.d. continuous random variables X, one need to calculate the PDF of $\underline{X}$ from $f_X(x)$, which may not be straightforward in most cases. Instead, we use the following lemma to obtain the integral in (3.108).

Lemma 3.4.20.

$$\int_{x=0}^{\tau}xf_{\underline{X}}(x)\mathrm{d}x=\int_{0}^{\tau}(1-F_{\underline{X}}(x))\mathrm{d}x-\tau(1-F_{\underline{X}}(\tau)). \tag{3.109}$$

Proof. Since $1-F_{\underline{X}}(x)$ is the complementary CDF of $\underline{X}$,

$$\begin{aligned}\int_0^{\tau}(1-F_{\underline{X}}(x))\mathrm{d}x&=\int_0^{\tau}\int_x^{\infty}f_{\underline{X}}(t)\,\mathrm{d}t\,\mathrm{d}x\\&=\int_0^{\tau}\int_0^{t}f_{\underline{X}}(t)\,\mathrm{d}x\,\mathrm{d}t+\int_{\tau}^{\infty}\int_0^{\tau}f_{\underline{X}}(t)\,\mathrm{d}x\,\mathrm{d}t\end{aligned} \tag{3.110a}$$

$$
\begin{aligned}
&= \int_0^\tau t f_{\underline{X}}(t)\,\mathrm{d}t + \int_\tau^\infty \tau f_{\underline{X}}(t)\,\mathrm{d}t \\
&= \int_0^\tau x f_{\underline{X}}(x)\,\mathrm{d}x + \tau\bigl(1 - F_{\underline{X}}(\tau)\bigr). \qquad (3.110\text{b})
\end{aligned}
$$

We note that (3.110a) is obtained by the change of integration order. Rearranging terms in (3.110b) completes the proof. □

Substituting (3.58), (3.108) and Lemma 3.4.20 back to (3.107), we have

$$
\begin{aligned}
\Delta &= \frac{\tau(1+q)}{2(1-q)} + \frac{1}{1-q}\left(\int_0^\tau (1 - F_{\underline{X}}(x))\mathrm{d}x - \tau(1 - F_{\underline{X}}(\tau))\right) \\
&= \frac{\tau(1+q)}{2(1-q)} + \frac{1}{1-q}\int_0^\tau (1 - F_X(x))^r \mathrm{d}x - \frac{\tau q}{1-q} \\
&= \frac{\tau(1-q)}{2(1-q)} + \frac{1}{1-q}\int_0^\tau (1 - F_X(x))^r \mathrm{d}x \\
&= \frac{\tau}{2} + \frac{1}{1 - \bigl(1 - F_X(\tau)\bigr)^r}\int_0^\tau (1 - F_X(x))^r \mathrm{d}x. \qquad (3.111)
\end{aligned}
$$

3.5 Multicast with Prioritized Nodes

3.5.1 Priority-count Restart Policy with Feedback

In this section, we look at a prioritized restart policy in which the receiving storage nodes are categorized into two groups. The *priority* group consists of nodes that require the delivery of every update, while all other nodes without the delivery requirement are regarded as the *non-priority* group. Once a node receives an entire update message, it acknowledges the source by sending instantaneous feedback. This model arises in a variety of delay-sensitive applications, e.g. vehicle networks where the update messages are popular and simultaneously request by large numbers of users. Some receiving nodes require the history of all updates for the purpose of data aggregation and processing; thus the delivery of every update message is crucial.

The system model with prioritized counting is in Figure 3.1. We only evaluate the age performance at the storage nodes, which is equivalent to where there is a receiver with an access number $r = 1$. The priority group consists of nodes $1, \ldots, k$, and the source guarantees the delivery of every update to all of these priority nodes. Similar to

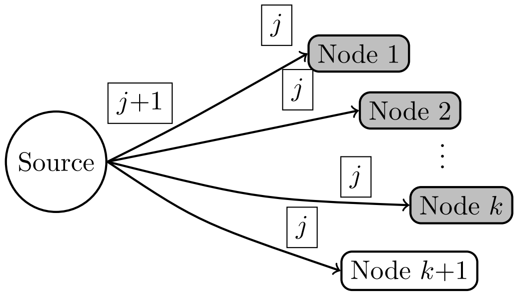


Figure 3.9: Alternative system with prioritized-count restart policy: the k nodes in the priority group are shaded. The transmission of update $j+1$ is initiated only after update j is delivered to all k nodes in the group.

the Earliest-count policy, we assume there is an instantaneous feedback channel from every node i back to the source, and node i acknowledges the source instantly as soon as the update is delivered to the node i. When all k nodes in the priority group report receiving the update j, this update is considered completed and the transmissions of this update to all other nodes are terminated. The source immediately generates the next update $j+1$ and repeats the multicast process. Since all the services are i.i.d., the service interval is i.i.d. and the policy is stationary. For every update j, let's denote that the priority nodes have service times $X_{1j}, X_{2j}, \ldots, X_{kj}$, and the non-priority node has service time $X_{k+1,j}$. The service interval is given by the maximum of the service time in the priority group, i.e.

$$Y_j = \max\{X_{1j}, X_{2j}, \ldots, X_{kj}\} = \mu_{k:k}. \tag{3.112}$$

The non-priority node, which is denoted by $k+1$, is not considered in the service interval. We are interested in the average age at both the priority node and the non-priority node, and we will show that the average age depends on the *order statistics* of the random link delay X_{ij}, which is similar to the Earliest-count policy.

3.5.2 AoI at Priority Nodes

We start by evaluating the average age at a single node in the priority group. Although the source transmits every update to the non-priority nodes, it only waits for the k nodes in the priority group. This indicates the age analysis for a priority node is equivalent to that using the Earliest-count policy which waits for all the k out of k nodes. Thus, it follows directly from (3.35) that

Theorem 3.5.1. *The average age at an individual node in the priority group is*

$$\Delta_P = \mu + \frac{\mu_{k:k}}{2} + \frac{\sigma_{k:k}^2}{2\mu_{k:k}}.$$

Note that Theorem 3.5.1 is valid for any distribution of X with finite mean and variance.

Corollary 3.5.2. *For shifted exponential* (λ, c) *service time* X*, the average age at an individual node in the priority group is lower bounded by*

$$\Delta_P \geq \frac{3c}{2} + \frac{1}{\lambda} + \frac{\log k + \gamma}{2\lambda}, \tag{3.113}$$

Proof. Substituting (3.18a) and (3.18b) into Theorem 3.5.1 gives

$$\Delta_P = \frac{3c}{2} + \frac{1}{\lambda} + \frac{H_k}{2\lambda} + \frac{H_{k^2}}{2\lambda^2 c + 2\lambda H_k}. \tag{3.114}$$

Note that $H_{k^2} = \sum_{i=1}^{k} \frac{1}{k^2}$ is monotonically increasing for $n \in \mathbb{Z}^+$ and $\lim_{k\to\infty} H_{k^2} = \pi^2/6$. Thus, given λ and c,

$$\lim_{k\to\infty} \frac{H_{k^2}}{2\lambda^2 c + 2\lambda H_k} \downarrow 0. \tag{3.115}$$

The harmonic number is given by $H_k = \log k + \gamma + O(\frac{1}{k})$, which can be lower bounded by

$$H_k \geq \log k + \gamma, \qquad \text{for } k \in \mathbb{Z}_{>0}. \tag{3.116}$$

Thus, (3.113) is given by substituting (3.115) and (3.116) into (3.114). $\square$

Corollary 3.5.2 indicates that the average age in the priority group Δ_P is independent of the number of nodes n in the system, and it behaves almost like a logarithmic function as the number of nodes k increases.

3.5.3 AoI at Non-priority Nodes

For a node in the non-priority group, the transmission of the current update is terminated right after the delivery of the update to all the k nodes in the priority group. That is, a non-priority node i fails to receive the update j if and only if the service time X_{ij} is larger than the service times of all the k nodes in the priority group. For i.i.d. service time X, the probability that X_{k+1} is the largest among all $k+1$ nodes is simply

$$q_{(k)} = 1/(k+1), \tag{3.117}$$

since we can group all the $k+1$ nodes and the rank of X_{k+1} among $k+1$ random variables is uniform from 1 to $k+1$.

We note that whether an update j is delivered to the node $k+1$, which is indicated by ψ_j, and the service interval Y_j are no more independent, and thus Corollary 3.3.4 is not applicable. Intuitively, the non-priority node successfully gets an update if $X_{k+1,j} < \max(X_{1j}, \ldots, X_{kj}) = Y_j$. Given that an update j is delivered to node $k+1$, we have extra information about the random variable Y_j since Y_j is expected to be larger. Hence, we need to use a more general average age formula in Theorem 3.3.3.

The key to evaluate Theorem 3.3.3 is the random variable Y_S and Y_F, which indicastes the length of a service interval given that the update is successfully delivered to the non-priority node $k+1$. We note that Y_S has CDF $F_{Y_S}(y) = F_{Y|Y\geq X_{k+1}}(y)$. Given $Y \geq X_{k+1}$, Y is the largest service time among $X_1, X_2, \ldots, X_{k+1}$, i.e.,

$$Y_S = X_{k+1:k+1}. \tag{3.118}$$

Similarly, Y_F has CDF $F_{Y_F}(y) = F_{Y|Y<X_{k+1}}(y)$. That means, the maximum of $X_1, X_2, \ldots, X_k$ is smaller than X_{k+1}, and Y is the k-th smallest among all $k+1$

random variables, i.e.

$$Y_F = X_{k:k+1}. \tag{3.119}$$

This also proves our conjecture above that Y_j and ψ_j are dependent and thus $Y \neq Y_F \neq Y_S$. In Theorem 3.3.3, we also note that $\tilde{X}$ indicates the service time of a non-priority node $k+1$ given that $X_{k+1} < \max(X_1, \dots, X_k)$. This condition implies X_{k+1} cannot be the largest among all $k+1$ nodes. Thus,

$$\mathrm{E}\left[\tilde{X}\right] = \mathrm{E}[X_{k+1} \,|\, X_{k+1} < X_{k+1:k+1}] = \frac{1}{k}\sum_{i=1}^{k} \mu_{i:k+1}. \tag{3.120}$$

Substituting (3.117), (3.118), (3.119) and (3.120) back into Theorem 3.3.3 yields

Theorem 3.5.3. *The average age at an individual node in the non-priority group is*

$$\Delta_E = \frac{1}{k}\sum_{i=1}^{k} \mu_{i:k+1} + \delta_1(k) + \delta_2(k),$$

where we denote

$$\begin{aligned}\delta_1(k) &= \frac{\sigma^2_{k:k+1} + k\sigma^2_{k+1:k+1}}{2(k+1)\mu_{k:k}}\\ \delta_2(k) &= \frac{\frac{k+2}{k}\mu^2_{k:k+1} + k\mu^2_{k+1:k+1} + 2\mu_{k+1:k+1}\,\mu_{k:k+1}}{2(k+1)\mu_{k:k}}.\end{aligned}$$

For exponential service times, we have the next claim follows from Theorems 3.5.1 and 3.5.3.

Theorem 3.5.4. *For exponential service time X, the average age is the same for both priority and non-priority nodes and is given by*

$$\Delta_E = \Delta_P = \frac{1}{\lambda} + \frac{H_k}{2\lambda} + \frac{H_{k^2}}{2\lambda H_k}, \tag{3.121}$$

where k is the priority group size.

Proof. For priority nodes, we obtain the average age by substituting (3.18a) and (3.18b) to Theorem 3.5.1 with $c = 0$, which directly yields (3.121). For non-priority nodes, the first term in Theorem 3.5.3 is

$$\begin{aligned}
\delta_0(k) &= \frac{1}{k}\sum_{i=1}^{k}\mu_{i:k+1} && (3.122a)\\
&= \frac{1}{k}\sum_{i=1}^{k}\frac{H_{k+1}-H_{k+1-i}}{\lambda} && (3.122b)\\
&= \frac{H_{k+1}}{\lambda} - \frac{1}{\lambda k}\sum_{i=1}^{k}H_i && (3.122c)\\
&= \frac{H_{k+1}}{\lambda} - \frac{k+1}{\lambda k}(H_{k+1}-1) && (3.122d)\\
&= \frac{1}{\lambda} + \frac{1}{\lambda k} - \frac{H_{k+1}}{\lambda k}. && (3.122e)
\end{aligned}$$

In (3.122d), we use the series identity of Harmonic numbers $\sum_{i=1}^{k} H_i = (k+1)(H_{k+1}-1)$. Similarly, substituting (3.18a) and (3.18b) into $\delta_1(k)$ and $\delta_2(k)$ gives

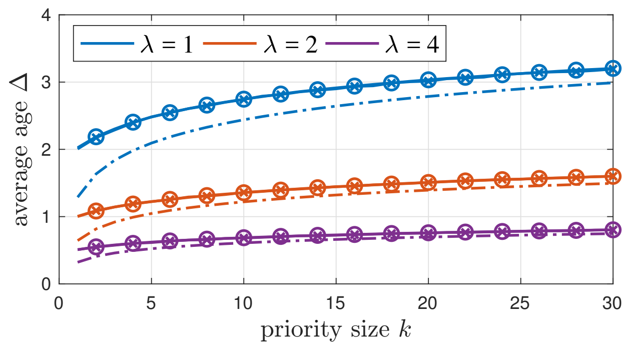
$$\begin{aligned}
\delta_1(k) &= \frac{(H_{(k+1)^2}-1) + kH_{(k+1)^2}}{2(k+1)\lambda H_k}\\
&= \frac{H_{k^2}}{2\lambda H_k} - \frac{k}{2\lambda(k+1)^2 H_k}, && (3.123)\\
\delta_2(k) &= \frac{k+2}{2k(k+1)}\frac{(H_{k+1}-1)^2}{\lambda H_k} + \frac{k}{2(k+1)}\frac{H_{k+1}^2}{\lambda H_k}\\
&\quad + \frac{1}{k+1}\frac{(H_{k+1}-1)H_{k+1}}{\lambda H_k}. && (3.124)
\end{aligned}$$

We note that $\delta_0(k)$ in (3.122e) and $\delta_2(k)$ in (3.124) only contain first order harmonic numbers, thus we combine two terms and rewrite $H_{k+1} = H_k + 1/(1+k)$, which gives

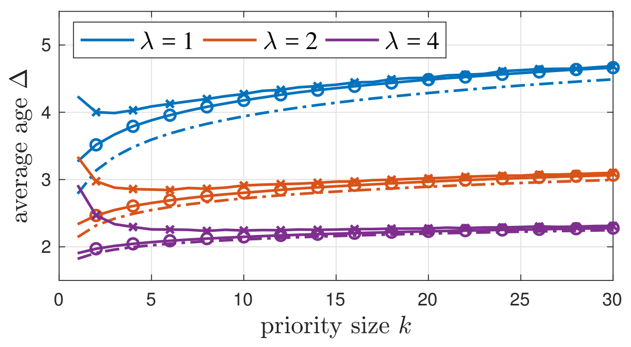
$$\delta_0(k) + \delta_2(k) = \frac{1}{\lambda} + \frac{H_k}{2\lambda} + \frac{k}{2\lambda(k+1)^2 H_k} \tag{3.125}$$

The claim is given by the sum of (3.123) and (3.125). □

Theorem 3.5.4 implies that the average age is identical for both groups regardless of whether an update is delivered to a node or not, if the service times are exponentially distributed.



(a) exponential X.



(b) shifted exponential X with $c = 1$.

Figure 3.10: Average age versus the priority group size k. circle $\circ$ marks the priority group and cross $\times$ marks the non-priority group. The lower bound for priority group is shown as dashed line.

3.5.4 Numerical Comparisons

Figures 3.10a and 3.10b compare the simulation results of the average age for the priority group Δ_P and the non-priority group Δ_E as a function of the priority group size k. In Figure 3.10a, the link delay to every node i is exponentially distributed with different λ. The average age curves for both groups overlap with each other and increase monotonically, which matches Theorem 3.5.4. The lower bound on the average age for the priority group in Corollary 3.5.2 captures the trend for varying k, and becomes tighter for sufficiently large k. Figure 3.10b shows the similar result for shifted exponential delay with $c = 1$. For small group size k, there is a significant difference between the average age for two groups. As k increases, the age for non-priority group Δ_E decreases slightly in the beginning and climbs up after a certain k. We also observe that the age difference between two groups vanishes for large enough k.

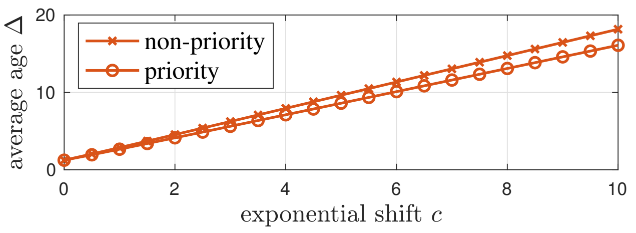


Figure 3.11: Average age versus the exponential shift c with $\lambda = 2$. The priority group size $k = 5$.

Figure 3.11 depicts the average age as a function of the shift parameter c for shifted exponential delay X. In Figure 3.11 with exponential rate $\lambda = 2$, both groups have almost linear increasing average age for different the constant shift c. The two curves start at the same point for $c = 0$, and the difference in slopes leads to a larger gap between two curves as c increases.

Chapter 4

Resource Sharing For Multi-source Updating

In the previous chapters, we focus on the age analysis for systems that consist of only one source. We assume the service facility, either the lossless bit pipe in Chapter 2 or the multicast network in Chapter 3, is dedicated to the transmission of the updates from a single source. In complex updating systems with multiple sources and receivers, the service facility is usually shared among multiple update streams. In order to resolve the resource contention among multiple sources, the service facility has to allocate its service capacity to different sources. For example, arriving updates from different sources can be regulated by queueing and processed according to certain disciplines, or they can scheduled in real-time by some controlling algorithm. In this chapter, we examine two updating systems, including remote cache updating [115] and computing jobs scheduling [116], both with resource sharing at the service facility, and demonstrate the resource allocation scheme that minimizes the corresponding age metrics.

4.1 Remote Cache Updating

Data generated at remote sources is often stored in a local cache to improve data availability and to reduce backhaul network traffic. Consider a system where a local server is connected to multiple remote sources and maintains local copies of the sources as shown in Figure 4.1. This system arises from a variety of applications. For instance, the local server could represent a web search engine that maintains time-stamped copies of millions of webpages. Alternatively, the remote sources could also be sensors in IoT devices which collect environmental or human-body data such as temperature or heart rate. In these examples, the local server is simply an aggregator that collects different types of data for further computation and analysis.

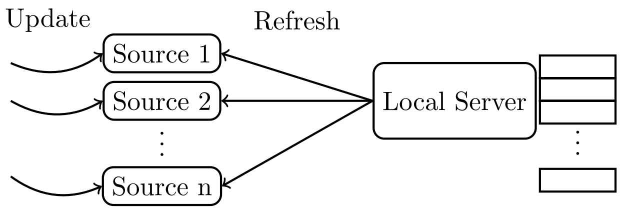


Figure 4.1: Diagram of local cache refresh with n remote sources.

In many of these applications, the data at the remote sources are dynamic and subject to random changes by external environment. The remote data item is updated randomly and replaced by new versions at the source, and these updates occur independently without being pushed to the local cache. In order to stay current about the information at the source, the local server has to refresh its cache by periodically downloading a new version of the data from the remote source. Although periodic refresh prevents the local cache from becoming outdated, the local server is usually subject to a finite refresh rate because it is tracking a large number of sources. As a result, the local server has to allocate its constrained refreshing rate to the set of sources with various update rates and popularities.

In order to measure the freshness of the local cache, various freshness metrics have been introduced and analyzed. In web crawling and news subscription problems [117–119], the *age* of the local copy is defined as the time difference between now and when the local copy became desynchronized with the remote source. We rephrase and refer to this metric as the *age of synchronization* (AoS), since it is a different timeliness metric other than the AoI in this work. The AoI metric has also been applied to the evaluation of cacheing systems. In [120], content packets arrive at a local limited-capacity cache, and the local cache decides dynamically which content will remain in the finite buffer based on the effective age, which is a function of the history of requests and the freshness of the packet. Under a different caching model in which a remote server sends dynamic content items to a local cache through a limited-capacity data link, it was shown [121] that the optimal updating rate to minimize the weighted age over all items follows a square root law for the content popularities. The work in this section is mainly motivated by [121] but also different in several ways. We assume content item is located at different server

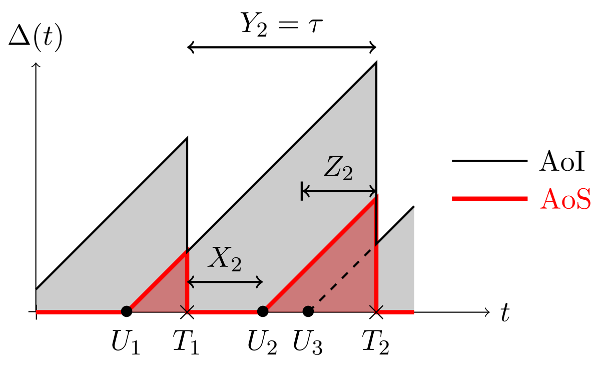


Figure 4.2: Sample paths of the AoS and AoI. $\bullet$ indicates the update at the remote source and $\times$ indicates the data update is synchronized at the local server.

sites, and each content is replaced by new versions with an update rate specified by an external random process. We examine this local cache refresh system and address the similar refresh rate allocation problem with two different freshness metrics. The goal is to ensure the freshness of the local cache over all the sources, and compare the optimal allocation policies for two different metrics. Another interesting question of our interest here is whether and when there exists an optimal policy for both metrics.

Here we consider a distributed content caching system with n remote data sources, and a local server which maintains local copies of all the sources as shown in Figure 4.1 The remote data sources receive random updates that reflect the real-world database changes. Since the local server operates independently of the remote sources, the local copy of a particular source becomes out-of-date once the source receives an update and the local copy differs from the source. To overcome the asynchrony between the server and the sources, the local server periodically fetches the most recent information from each source. More specifically, we refer to the cache update at the local server as a *synchronization* event, or *refresh* event. Here we model the random update at each source i as a Poisson process with rate λ_i, and the server refreshes the local copy i periodically every τ_i time units.

Figure 4.2 depicts a sample path of the remote source updates and the local cache synchronizations. Let's define $U_1, U_2, \ldots, U_j$ as the sequence of remote source update times, and $T_1, T_2, \ldots, T_j$ as the sequence of local refresh times. Thus, we denote $M(t)$ as the number of local refreshes up to time z. We illustrate the difference between the two freshness metrics as follows:

1. *Age of Synchronization (AoS):* Let $U^S(t)$ denote the earliest time that the remote source gets an update since the last refresh of the local copy, i.e.,

$$U^S(t) = \min\{U_j \,|\, U_j > T_{M(t)}\}. \tag{4.1}$$

For instance, $U^S(T_1)$ at time $t = T_1$ is U_1 in Figure 4.2, since U_1 is the only update time before T_1. Similarly, $U^S(T_2)$ is U_2 since U_2 is the first update time after the previous sync time T_1. The AoS of the local cache at time t is then defined as

$$\Delta^S(t) = (t - U^S(t))^+, \tag{4.2}$$

where $(\cdot)^+ = \max(\cdot, 0)$. Note that if the local cache is the same as the remote source, then $\Delta^S(t) = 0$.

2. *Age of Information (AoI):* We denote $U^I(t)$ as the time when the latest update is generated,

$$U^I(t) = \max\{U_j \,|\, U_j \le T_{M(t)}\}. \tag{4.3}$$

The AoI of the local cache at time t is $\Delta^I(t) = (t - U^I(t))^+$. Note that AoI only depends on the latest update time.

Figure 4.2 demonstrates the sample paths of two age metrics. AoS remains zero until U_1, the time instance that the local cache and the remote source become desynchronized, and then it increases linearly until the synchronization point T_1. In contrast, AoI starts accumulating from some initial value, and is reset to the desynchronized time gap $T_1 - U_1$ at time T_1. At time T_2, AoI drops to the time difference between now and the latest update $T_2 - U_3$, while AoS is reset to zero.

We emphasize that although both metrics measure the freshness of the content cache, they are different in terms of the reference object. AoS uses the remote source itself as the reference, while AoI also takes the staleness of the remote sources into account.

Based on the definition of AoI, a remote source also becomes stale compared to the external dynamic process it is capturing. That is, the local cache catches up with a remote source once a refresh event is completed.

4.1.1 Minimizing AoI and AoS

Here we consider the scenario where the server has a constrained total refresh frequency f for all the sources. Our objective is to allocate the total refresh frequency to each source such that the average age over all sources is minimized. Let's assume that the server refreshes source i periodically with frequency $f_i = 1/\tau_i$. We first start with the following discussion of the average AoS and AoI for each source i at the local server.

Lemma 4.1.1. *For a remote source i with Poisson update rate λ_i, the age of synchronization at the local server is given by*

$$\Delta_i^S = \frac{1}{2f_i} - \frac{1}{\lambda_i} + \frac{f_i}{\lambda_i^2}\left(1 - e^{-\lambda_i/f_i}\right). \tag{4.4}$$

Proof. We refer to the time interval from the previous synchronization at T_{j-1} to the next synchronization at T_j as a synchronization/refresh interval, and we define $Y_j = T_j - T_{j-1}$ as the inter-synchronization time. For each source i, the length of the synchronization interval is fixed at τ_i, i.e., $Y_j = \tau_i$ for all j. Similarly, we denote the waiting time for the first source update as X_j where $X_j = U^S(T_j) - T_{j-1}$. An example of Y_2 and X_2 is shown in Figure 4.2. In this case, the average AoS in the second synchronization interval is the average area of the red triangle starting from time $T_1 + X_2$. Let's denote the area of the age triangle for a given $X_j = x$ as

$$A(x) = \begin{cases} (\tau_i - x)^2/2, & x \le \tau_i \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

For a Poisson update process with exponential rate λ_i, the average AoS is given by

$$\Delta_i^S = \frac{\mathrm{E}[A]}{\tau_i} = \frac{1}{\tau_i}\int_{x=0}^{\infty} \mathrm{E}[A|X=x]\lambda_i e^{-\lambda_i x}\mathrm{d}x$$

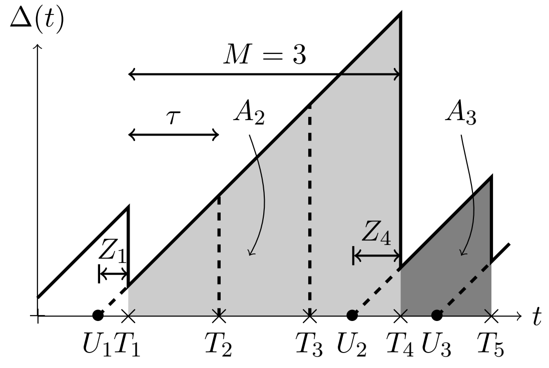


Figure 4.3: Sample paths of AoI with infrequent source updates: no update in the 2nd and 3rd intervals. • indicates the update at the remote source and × indicates the data update is synchronized at the local server.

$$= \frac{1}{\tau_i}\int_{x=0}^{\tau_i} \frac{(\tau_i - x)^2}{2}\lambda_i e^{-\lambda_i x}\mathrm{d}x$$
$$= \frac{\tau_i}{2} - \frac{1}{\lambda_i} + \frac{1 - e^{-\lambda_i \tau_i}}{\lambda_i^2 \tau_i}. \tag{4.6}$$

Substituting $f_i = 1/\tau_i$ completes the proof. Defining the load of the system as $\rho_i = \lambda_i / f_i$, we can also rewrite Lemma 4.1.1 (was also presented in [117]) as

$$\Delta_i^S = \frac{1}{\lambda_i}\left(\frac{\rho_i}{2} - 1 + \frac{1 - e^{-\rho_i}}{\rho_i}\right). \tag{4.7}$$

□

Lemma 4.1.2. *For a remote source with Poisson update rate λ_i, the age of information (AoI) at the local server is given by*

$$\Delta_i^I = \frac{1}{2f_i} + \frac{1}{\lambda_i}. \tag{4.8}$$

Proof. When the local server synchronizes the cache for source i at time T_j, the AoI is reset to the age of the most recent update. We denote the time difference from the latest update U_k to the synchronization point T_j as $Z_j = T_j - U^I(T_j)$ as shown in Figure 4.3. In this case, the AoI increases linearly in the second and third synchronization interval since there is no source update in these two intervals. We also denote random variable M as the number of intervals until the local server gets a new update from the source;

M is geometric with failure probability $p' = e^{-\lambda_i \tau_i}$. The average AoI is the average area under the sawtooth, which can be decomposed into the sum of polygonal areas as shown in Figure 4.3. Given that there are at least one updates in the j-th interval, e.g. the intervals 1 and 4, we define a new random variable $\tilde{Z}_j$ such that $P_{\tilde{Z}_j}(z) = P_{Z_j|Z_j<\tau_i}(z)$. With the evaluation of the figure, we obtain

$$\Delta_i = \lim_{l\to\infty} \frac{\sum_{k=1}^{l} A_k}{\sum_{k=1}^{l} Y_k} = \frac{\mathrm{E}[A_k]}{\mathrm{E}[Y_k]}, \tag{4.9}$$

where $\mathrm{E}[Y_k] = \mathrm{E}[M\tau_i] = \tau_i \,\mathrm{E}[M]$. And the polygonal area is given by

$$A_k = \frac{(\tilde{Z}_j + (M\tau_i + \tilde{Z}_j))M\tau_i}{2}. \tag{4.10}$$

It follows that

$$\mathrm{E}[A_k] = \tau_i \,\mathrm{E}\Big[\tilde{Z}_j\Big]\,\mathrm{E}[M] + \frac{\tau_i^2 \,\mathrm{E}\big[M^2\big]}{2}. \tag{4.11}$$

Substituting (4.11) back to (4.9) yields

$$\Delta_i = \mathrm{E}\Big[\tilde{Z}\Big] + \frac{\tau_i \,\mathrm{E}\big[M^2\big]}{2\,\mathrm{E}[M]}. \tag{4.12}$$

Note that $\tilde{Z}$ represents time difference between the latest source update and the synchronization point given that there exists an update in the synchronization interval j, i.e. $Z_j > 0$. Since the update process is Poisson, it implies

$$\begin{aligned}\mathrm{E}\Big[\tilde{Z}\Big] = \mathrm{E}[Z|Z \le \tau_i] &= \int_{z=0}^{\tau_i} z \frac{\lambda_i e^{-\lambda_i z}}{1 - e^{-\lambda_i \tau_i}} \mathrm{d}z \\ &= \frac{1}{\lambda_i} - \frac{\tau_i e^{-\lambda_i \tau_i}}{\left(1 - e^{-\lambda_i \tau_i}\right)}.\end{aligned} \tag{4.13}$$

For geometric M, we have

$$\frac{\mathrm{E}\big[M^2\big]}{\mathrm{E}[M]} = \frac{1+p'}{1-p'} = \frac{1 + e^{-\lambda_i \tau_i}}{1 - e^{-\lambda_i \tau_i}}. \tag{4.14}$$

Substituting (4.13) and (4.14) back into (4.12) gives (4.8). □

We note that Lemma 4.1.2 is similar to the relaxed problem (13) in [121]. The relaxed age in [121] can be viewed as a special case with $\lambda_i = 1$, which indicates the remote content is updated at every time unit.

We assume each source i has a popularity factor p_i at the local server. The optimization problem is then given by

$$\min_{\{f_i\}} \sum_{i=1}^{n} p_i \Delta_i, \quad \text{subject to } \sum_{i=1}^{n} f_i = f, \tag{4.15}$$

where the age metric Δ_i can be the average AoS in Lemma 4.1.1 or the average AoI in Lemma 4.1.2.

4.1.1.1 AoS minimization

Lemma 4.1.3. *The objective function for AoS $\Delta_S = \sum_{i=1}^{n} p_i \Delta_i^S$ is convex on $f_1, f_2, \ldots, f_n$.*

Proof. For the average AoS in 4.1.1, we have

$$\frac{\mathrm{d}^2 \Delta_i^S}{\mathrm{d} f_i^2} = \frac{1 - e^{-\frac{\lambda_i}{f_i}}}{f_i^3}. \tag{4.16}$$

For $f_i > 0$ and $\lambda_i > 0$, $e^{-\lambda_i/f_i} < 1$. Thus, $\frac{\mathrm{d}^2 \Delta_i^S}{\mathrm{d} f_i^2} > 0$ and Δ_i^S is convex on f_i. Since Δ^S is a linear separable function of Δ_i^S, $\Delta^S(f_1, f_2, \ldots, f_n)$ is convex. $\square$

Lemma 4.1.4. *The optimal solution $f_i^* = f_i(\alpha)$ is given by the root of the following equation*

$$g(f_i) = -\frac{1}{2f_i^2} + \frac{1 - e^{-\frac{\lambda_i}{f_i}}}{\lambda_i^2} - \frac{e^{-\frac{\lambda_i}{f_i}}}{\lambda_i f_i} = \frac{\alpha}{p_i}, \tag{4.17}$$

where the constant α is the root of the following equation

$$\sum_{i=1}^{n} f_i(\alpha) = f. \tag{4.18}$$

Proof. To solve the minimization on a convex function, we denote α as the Lagrange multiplier and form the Lagrangian as follows

$$L(f_1, f_2, \ldots, f_n) = \sum_{i=1}^{n} p_i \Delta_i^S + \alpha \sum_{i=1}^{n} f_i - \alpha f, \tag{4.19}$$

where Δ_i^S is given by Theorem 4.1.1. Setting $\partial L/\partial f_i = 0$ for every $i \in \{1, 2, \ldots, n\}$ yields (4.17). Note that the optimal f_i^* is a function of the constant α. To obtain the α, we substitute $f_i^* = f_i(\alpha)$ into the constraint $\sum_{i=1}^{n} f_i(\alpha) = f$. □

Here we note that directly solving (4.15) with AoS in Lemma 4.1.1 is complicated. Instead, we propose two approximations for Lemma 4.1.1 that lead to the following theorems.

Theorem 4.1.5 (Policy P1). *When the constraint f is sufficiently large, a near-optimal synchronization frequency f_i for source i that minimizes an approximation of the weighted average of AoS over all sources is*

$$f_i^* = \frac{f(p_i\lambda_i)^{1/3}}{\sum_{i=1}^{n}(p_i\lambda_i)^{1/3}}. \tag{4.20}$$

Proof. The Taylor series for $e^{-\rho}$ is

$$e^{-\rho} = 1 - \rho + \frac{\rho^2}{2} - \frac{\rho^3}{6} + O(\rho^4). \tag{4.21}$$

We let $\rho = \lambda_i/f_i$ and rewrite $g(f_i)$ in (4.17) in the expansion form

$$\begin{aligned}
g(f_i) &= -\frac{1}{2f_i^2} + \frac{1 - \left(1 - \frac{\lambda_i}{f_i} + \frac{\lambda_i^2}{2f_i^2} - \frac{\lambda_i^3}{6f_i^3} + \ldots\right)}{\lambda_i^2} - \frac{1 - \frac{\lambda_i}{f_i} + \frac{\lambda_i^2}{2f_i^2} - \frac{\lambda_i^3}{6f_i^3} + \ldots}{\lambda_i f_i} \\
&= -\frac{1}{2f_i^2} + \left(\frac{1}{\lambda_i f_i} - \frac{1}{2f_i^2} + \frac{\lambda_i}{6f_i^3} - \ldots\right) - \left(\frac{1}{\lambda_i f_i} - \frac{1}{f_i^2} + \frac{\lambda_i}{2f_i^3} - \frac{\lambda_i^2}{6f_i^4} + \ldots\right) \\
&= \left(\frac{\lambda_i}{6f_i^3} - \frac{\lambda_i^2}{24f_i^4} + \ldots\right) + \left(-\frac{\lambda_i}{2f_i^3} + \frac{\lambda_i^2}{6f_i^4} + \ldots\right) \\
&= -\left(\frac{3\lambda_i}{3!f_i^3} - \frac{\lambda_i}{3!f_i^3}\right) + \left(\frac{4\lambda_i^2}{4!f_i^4} - \frac{\lambda_i^2}{4!f_i^4}\right) - \ldots \\
&= \sum_{n=3}^{\infty} \frac{(-1)^n (n-1)\lambda_i^{n-2}}{n!f_i^n}
\end{aligned}$$

$$= -\frac{\lambda_i}{3f_i^3} + O\left(\frac{1}{f_i^4}\right). \tag{4.22}$$

We consider the case where every λ_i is fixed and $f_i \to \infty$ and the higher order term $O\left(\frac{1}{f_i^4}\right)$ is negligible. For each source i, we set

$$-\frac{p_i\lambda_i}{3f_i^3} + \alpha = 0. \tag{4.23}$$

Solving (4.23) under constraint in (4.1.4) yields (4.20). □

Theorem 4.1.6 (**Policy P2**). *When the constraint f is small compared to $\lambda_1, \lambda_2, \ldots, \lambda_n$, a near-optimal synchronization frequency f_i for source i that minimizes an approximation of the weighted average of AoS over all sources is*

$$f_i^* = \frac{f\sqrt{p_i}}{\sum_{i=1}^n \sqrt{p_i}}. \tag{4.24}$$

Proof. Here we consider the case where f_i is small and $\rho_i = \lambda_i/f_i$ is large for all i. Since $\lim_{\rho_i\to\infty} \frac{1-e^{-\rho_i}}{\rho_i} = 0$ in (4.7), we can approximate (4.7) by

$$\Delta_i^S \approx \frac{\tau_i}{2} - \frac{1}{\lambda_i}. \tag{4.25}$$

As in the proof for Thm. 4.1.5, we form the Lagrangian

$$L(f_1, \ldots, f_n) = \sum_{i=1}^n \left(\frac{1}{2f_i} - \frac{1}{\lambda_i}\right) - \alpha\sum_{i=1}^n f_i + \alpha f. \tag{4.26}$$

Setting $\partial L/\partial f_i = 0$ for every $i \in \{1, 2, \ldots, n\}$ yields $1/(2f_i^2) = \alpha$. Applying the constraint $\sum_{i=1}^n f_i = f$ in (4.15), it follows that the optimal refresh frequency for source i is given by (4.24). □

We observe in Theorem 4.1.5 that the AoS near-optimal refresh rate for a content i should be proportional to the cube root of both its popularity and update frequency when f is large. However, when f is small, the near-optimal policy in Theorem 4.1.6

does not depend on the update frequency at the source λ_i. Instead, the refresh rate follows a square-root law with respect to the popularity of the item p_i. When the source popularities are uniform, i.e., $p_1 = \ldots = p_n$, it was shown in [117] that the uniform refresh frequency allocation provides lower AoS than the update-proportional allocation policy that assigns the refresh frequency proportional to the source update rate. Here our near-optimal allocation policy lies between uniform and update-proportional.

4.1.1.2 AoI minimization

Theorem 4.1.7. *The optimal synchronization frequency f_i for source i that minimizes the weighted average of AoI over all sources is given by*

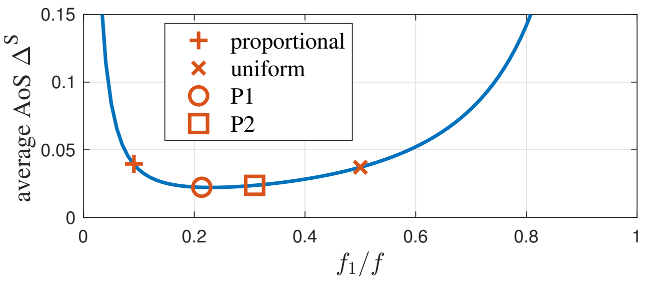
$$f_i^* = \frac{f\sqrt{p_i}}{\sum_{i=1}^{n}\sqrt{p_i}}. \tag{4.27}$$

Similar to the proof of Theorem 4.1.6, Theorem 4.1.7 is obtained by forming a Lagrangian using Lemma 4.1.2 .

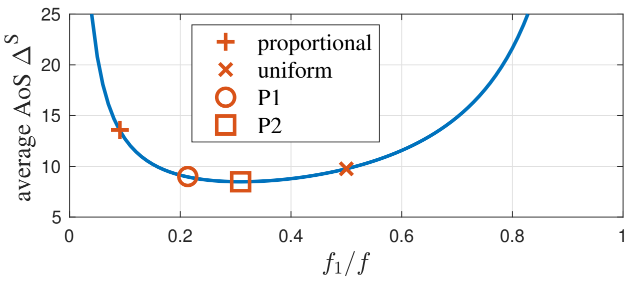
The AoI optimal refresh policy f_i^* in Theorem 4.1.7 is identical to the policy P2, which is optimal for small f in Theorem 4.1.6. We also note that this policy is identical to the optimal strategy in [121], indicating that the AoI optimal policy is independent of the refresh rates. This is mainly because AoI only captures the freshness of the most recent update at the source. Whenever the local server refreshes the cache, the AoI of the most recent update is statistically identical and exponentially distributed with rate λ_i, regardless of the refresh rate f_i, since the update process at the source is memoryless. This also implies that if the source popularities are uniform, i.e. $p_1 = p_2 = \ldots = p_n$, the optimal scheme is to assign the refresh frequency uniformly, $f_i = f/n$ for all i.

Corollary 4.1.8. *The average AoI of source i obtained through optimal policy P2 is given by*

$$\Delta_i^I = \frac{\sum_{i=1}^{n}\sqrt{p_i}}{2f\sqrt{p_i}} + \frac{1}{\lambda_i}. \tag{4.28}$$



(a) refresh constraint $f = 10$



(b) refresh constraint $f = 0.1$

Figure 4.4: AoS vs. the portion of refresh frequency f_1/f in a two sources model with unequal popularities $p_1 = 1, p_2 = 5$, and source update rates $\lambda_1 = 1, \lambda_2 = 10$.

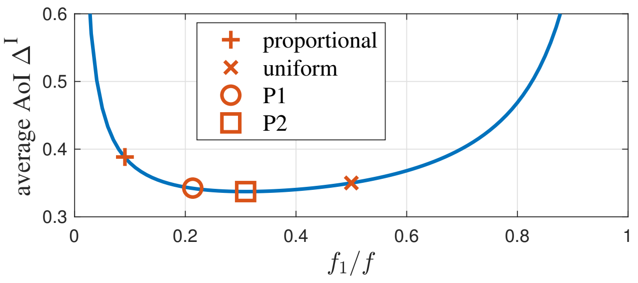


Figure 4.5: AoI vs. the portion of refresh frequency f_1/f in a two sources model with constraint $f = 10$, popularities $p_1 = 1, p_2 = 5$, and source update rates $\lambda_1 = 1, \lambda_2 = 10$.

4.1.2 Numerical Comparisons for Two Freshness Metrics

4.1.2.1 Special Case: Two Sources

To illustrate the difference between various refresh rate allocation policies, we first start with the simplest case, where the local server maintains the caches for only two sources. In Figure 4.4, the update rate at the two sources are $\lambda_1 = 1$ and $\lambda_2 = 10$, and we set the total refresh frequency f to be either (a) $f = 10$ or (b) $f = 0.1$. We assume the two sources have popularities $p_1 = 1$ and $p_2 = 5$. Figure 4.4a depicts the average AoS versus the portion of frequency, which is limited to $f = 10$, assigned to the first source

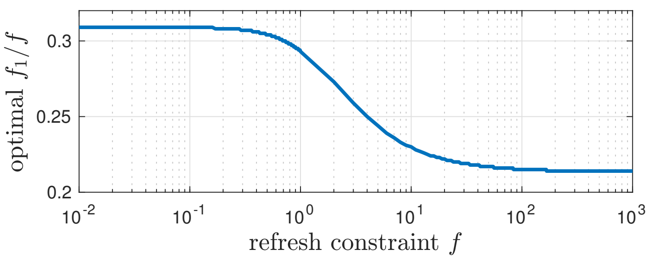


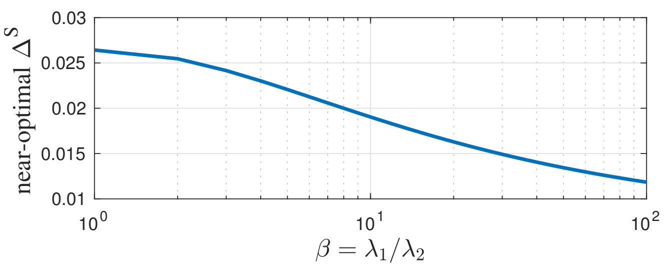
Figure 4.6: AoS-optimal allocation policy f_1/f vs. total refresh frequency constraint f. The two source update rates are given as $\lambda_1 = 1$ and $\lambda_2 = 10$.

f_1/f, in which different allocation policies are marked. We observe that policy P1 in Theorem 4.1.5 provides almost the minimum age. In this example with $\lambda_2 p_2/\lambda_1 p_1 = 50$, Theorem 4.1.5 implies that the AoS near-optimal refresh frequencies should satisfy $f_2/f_1 = 50^{1/3}$, thus $f_1 \approx 0.21f$. In Figure 4.4b, the total refresh frequency is $f = 0.1$. It is shown that the minimum age occurs at policy P2 in Theorem 4.1.6. In this case, the refresh frequencies should be $f_2/f_1 = \sqrt{5}$, i.e. $f_1 \approx 0.31f$.

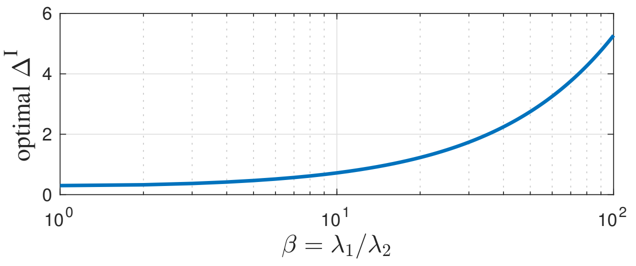
Similarly, Figure 4.5 depicts the average AoI as a function of f_1/f when $f = 10$. The AoI optimal policy, which is identical to P2, in Theorem 4.1.7 indicates that the AoI optimal refresh frequencies should also be $f_2/f_1 = \sqrt{5}$. We also note that policy P2 is independent of the source update rate λ_1 and λ_2. If the popularities are the same, $p_1 = p_2$, average AoI is minimized when $f_1 = f_2 = f/2$.

In Figure 4.6, we again fix the source update rate at $\lambda_1 = 1$ and $\lambda_2 = 10$, but vary the total refresh constraint f. The popularities are assumed to be $p_1 = 1$ and $p_2 = 5$. For every f, we obtain the optimal allocation policy f_1, f_2 by numerically solving the optimization problem in (4.15). We note that policy P1 is $f_1 \approx 0.21f$. When f is large, e.g. $f \geq 10$, policy P1 is very close to the true optimal policy. However, as f becomes smaller, e.g. $f \leq 0.01$, the AoS optimal policy converges to $f_1 \approx 0.31f$, indicating that the optimal policy follows a square root law for the popularity as policy P2, i.e., $f_1/f_2 = \sqrt{p_1/p_2} = 1/\sqrt{5}$. This also proves that when the total constraint f is small, P2 is the optimal policy for both AoS and AoI.

In Figs. 4.7a and 4.7b, we constrain the total source update rate at the two sources to be $\lambda_1 + \lambda_2 = 10$, and vary the ratio $\beta = \lambda_1/\lambda_2$ between two update rates. The



(a) Age of synchronization.



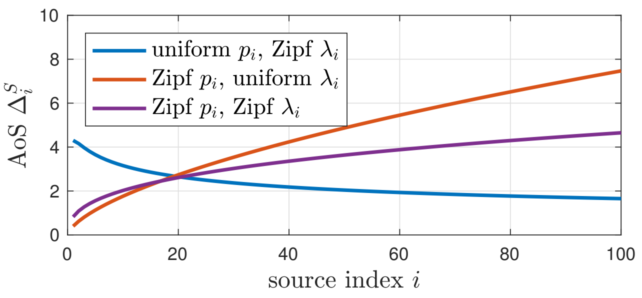
(b) Age of information.

Figure 4.7: Optimal average age vs. the ratio of two source update rates $\beta = \lambda_1/\lambda_2$ in a two sources model with equal popularity. The total source update rates are constrained by $\lambda_1 + \lambda_2 = 10$, and the local refresh rate is constrained by $f = 10$.

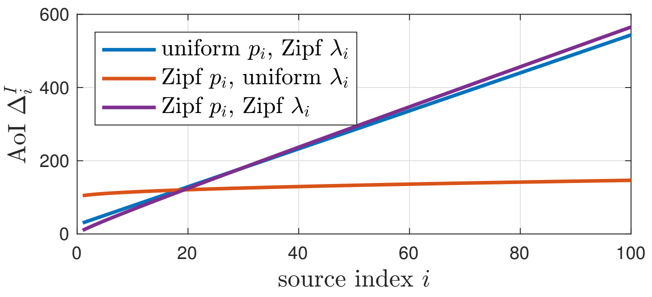
total refresh frequency is fixed at $f = 10$, and we assume the popularities are the same $p_1 = p_2$. In Figure 4.7a, we obtain the AoS near-optimal refresh policy f_1 and f_2 from Theorem 4.1.5 for each update ratio β. We observe that the average AoS is maximized when $\beta = 1$, i.e. the update rates for two sources are equal $\lambda_1 = \lambda_2$. As one source gets more frequent updates than the other, the average AoS decreases as the allocation policy puts more weight into the more frequent source. In Figure 4.7b, the AoI optimal policy f_1 and f_2 from Theorem 4.1.5 is applied for each update ratio β. In this case, $f_1 = f_2$ for all β since $p_1 = p_2$. We observe a different behavior that the average AoI increases as β increases, and the average AoI of the infrequent source blows up as $\beta \to \infty$ as presented in Lemma 4.1.2.

4.1.2.2 Many Sources

Now we consider an example with $n = 100$ sources. To demonstrate the average age at each source Δ_i instead of the weighted average over all the sources, we use uniform and Zipf distributions as examples for both the source popularity and the source update rate. We define the following distributions for p_i and note that they also apply to λ_i.



(a) Age of synchronization.



(b) Age of information.

Figure 4.8: The average age Δ_i for each source i in a system with $n = 100$ nodes, and Zipf distribution scale parameter $s = 1$.

- Uniform distribution: $p_i = 1/n$.
- Zipf distribution: $p_i = \frac{1/i^s}{\sum_1^n 1/i^s}$.

Figure 4.8a depicts the comparison of average AoS between different source update and popularity models. In this case, we set the scale parameter in Zipf distribution as $s = 1$. By comparing uniform and Zipf popularity p_i, we observe that more popular sources have lower average AoS since the allocation policy assigns higher refresh frequency to more popular sources. However, as the distribution model for source update λ_i changes from uniform to Zipf, our allocation policy leads to significantly lower average AoS for less frequent sources and higher age for more frequent sources. Figure 4.8b depicts the comparison of average AoI under the same scenario. Similarly, if we fix the update distribution model λ_i to be Zipf and compare uniform and Zipf popularity models, we observe the AoI optimal policy assigns more weight to popular sources and thus leads to lower average AoI for the popular ones.

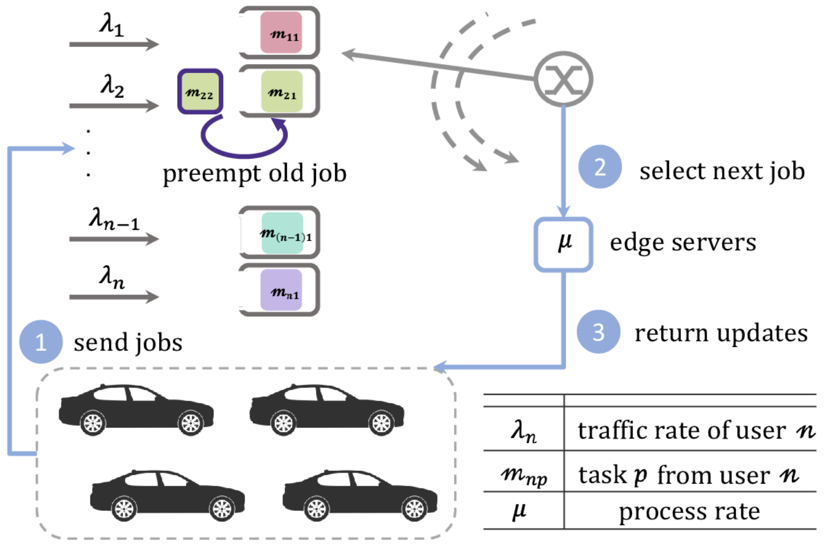


Figure 4.9: Example of edge cloud traffic scheduling system.

4.1.3 Possible Extensions

Although we have obtained nice results on the optimal rate allocation for any given timeliness metric, the results depend on the assumptions of instantaneous refresh and exponential update time. A critical one is that the refresh event completes in a very short time period, thus it's negligible in the analysis. We believe this is a proper assumption for web crawling systems, but may not be suitable for ultra-low latency communication systems for sensor networks. The time scale we're considering in a sensor network update, which is typically less than 10ms between samples, is rather small compared to website updates, thus the delay for a refreshing event may be significant and dominating.

The other possible improvement is a more refined definition of the age metrics. In this model, either AoI or AoS grows linearly in time, thus they only quantify the loss or distortion in time. It is of practical interest to look at more general penalty functions that captures the level of dissatisfaction depending on the instantaneous age.

4.2 Job Scheduling in Real-time Edge Computing

Most of the prior research on timely updates uses an "enqueue and forward" model, which assumes the source node receives randomly arriving information packets and selectively forwards them to the destination. However, in a wide range of computer vision applications, such as autonomous driving vehicles, virtual reality gaming, object tracking and facial recognition, networking delay is only part of the story when the service facility needs to further process the incoming information packets, especially when the computation overhead dominates the network transmission latency. These computer vision applications indeed demand an alternative "enqueue, process and forward" (EPF) model. For example, autonomous driving cars periodically, say every 20 ms, capture the front scenes with stereo cameras, and send them to a nearby edge cloud. The edge cloud is then required to perform heavy computer vision calculations [122–124] on those received stereo images where the output involves the estimated depth of the surrounding objects appeared in the images or 3D point cloud. Those outputs will be delivered back to the autonomous driving cars (the destination nodes) for better understanding of the traffic environment.

Timely environment updates are critical to guarantee the safety and efficiency of the driving experience. We note that the age of those updates can grow substantially as edge clouds perform computer vision calculations. Importantly, unlike central clouds with nearly unlimited computing resources, edge clouds are typically constrained by their computing capabilities and might be over-utilized when the incoming traffic is heavy. Thus, resource contention among different video streams and the randomness of the processing time may significantly contribute to making information stale.

Our first attempt is to examine an edge cloud computing system which supports the real-time processing of multiple video streams. We first start by simplifying the problem to the following baseline model. We assume the edge cloud servers are grouped as a single processing unit, which sequentially process stereo video frames from multiple users. We then can view each video frame arriving at the edge cloud server as a job, and the monitor at the user itself is receiving the processed results as the information updates. In this case, the source is self-updating itself through the closed-loop video frame processing at the edge cloud. The age of an update is then defined as the difference between current time and the generation time of that particular job at the source. Therefore, the objective is to obtain the optimal scheduling policy for job processing so that the information freshness at each user is maintained.

There have been many relevant works on the scheduling of multiple users to minimize the age of information [7, 25, 27, 125]. The scheduling of updates in an unreliable broadcast network with a base station and multiple receivers is considered in [25]. In this system, the base station accumulates updates from different sources but can only update at most one receiver at a time. A similar problem is considered in [27], in which an information update is discarded if it is not selected by the base station for transmission. Our work is motivated by the queueing model in [7], in which the job arrival times are synchronized among all sources. We note that the most relevant work to ours is [125]. It was shown by experiments that choosing the source with maximum age reduction leads to lower average age than several other schemes.

We note that the system model we consider for edge computing is very similar to the caching problem in Section 4.1. Both systems require the resource sharing at the service facility, either cache refresh or job computation, among multiple updating streams. The caching system differs from the edge computing system mainly by the size of sources. In web crawling applications, the local cache keeps track of a massive amount of sources, which makes it hard to keep track of the instantaneous age process for each source and perform real-time scheduling. Therefore, simply allocating the constrained refresh capacity to each content and periodically updating the local copy from the remote source is a more practical solution for the local cache. Intuitively, real-time scheduling of updates should outperform any stationary policy with service rate allocation, since scheduling utilizes the instantaneous state of the system.

4.2.1 Edge Computing Model and Age Penalty Function

Here we assume the edge cloud server is shared by n self-driving vehicles, and we refer to each vehicle as a user. As illustrated in Figure 4.9, each user i sequentially submits jobs to the edge cloud with rate λ_i, and each job is temporarily stored at a pre-processing buffer B_i that can hold a single user i job. In particular, an incoming job with newer generation time will replace an old job already stored in the buffer since the newer job always contains fresher information. Since each job is a video frame captured by the stereo camera, the job upload time from each user to the edge cloud is considered to be random. We denote m_{ij} as the j-th job from user i, and U_{ij} as the corresponding job upload time to the edge cloud. We assume the upload time U_{ij} is i.i.d. for all the jobs j corresponding to a user i. Once a job m_{ij} is generated at time $A_j^{(i)}$, it will be delivered to the buffer at time $A_j^{(i)} + U_{ij}$. At any time t, user i has submitted $N_i(t)$ jobs to the buffer, and the most recent job is generated at time $A_{N_i(t)}^{(i)}$, then the job stored at the buffer B_i has an instantaneous age

$$\delta_i(t) = t - A_{N_i(t)}^{(i)}, \quad i = 1, \ldots, n. \tag{4.29}$$

We refer to $\delta_i(t)$ as the *buffer age* as it equals the age of an observer who views jobs arriving at buffer B_i as updates. Under this model, the age $\delta_i(t)$ is reset to the upload time U_{ij} of the most recent job when the buffered job is replaced by a new job.

In this work, we assume the edge computing unit to be a single processor that can only handle one job at a time with processing rate S identical across all users i and jobs j. We consider only *work-conserving* policies: the server is kept busy whenever the buffers $B_1, B_2, \ldots, B_n$ are non-empty; if the buffers are all empty, then the server stays idle and waits for the next arriving job. For a work-conserving policy, once the server finishes processing the previous job but the buffers are not empty, the scheduler selects the next job from the buffers as shown in Figure 4.9.

Denote s as the processing time for job k and $\phi_k \in \{1, 2, \ldots, n\}$ as the indicator for the user corresponding to the k-th job processed by the server. We also assume that the processing time s is independent of the job arrival time. The processor starts the k-th job at time t_k and completes at time $t_k + s$. The processing result of job k, which is an information update, is then immediately sent back to the monitor at the corresponding user $\phi_k = i$. Since the processing result is much smaller than the original job size, and the downlink bandwidth is usually sufficiently large, we assume the download time is negligible. Here we define the age at the monitor of user i as the difference between the present time and user's knowledge about the environment. Since the received update at the monitor i contains the information generated at time $A_{N_i(t_k)}^{(i)}$, the instantaneous age at the monitor i is reset to

$$\Delta_i(t)|_{t=t_k+s} = t - A^{(i)}_{N_i(t_k)}. \tag{4.30}$$

After that, the age at the monitor increases linearly in time until the monitor receives another update.

We denote π as a scheduling policy that determines the job ϕ_k to be processed. In this model, the processor records the time stamp of each processed job, which is the generation time of these jobs. Thus, the processor also knows the set of instantaneous ages $\{\Delta_i(t) : i = 1, \ldots, n\}$ at each monitor at any time t. We let Π represent the set of causal work-conserving policies in which the scheduling decisions are made based on the history of the states of the system up to the present time. Here we only consider *non-preemptive* policies in which the processor must complete the processing of the current update before starting to serve another.

Figure 4.10 demonstrates sample paths of both age processes $\delta_i(t)$ and $\Delta_i(t)$ for a particular user i. The first job is generated by user i at time $A^{(i)}_1 = 0$, and arrives at the buffer B_i at time $A^{(i)}_1 + U_{i1}$. Once its processing is completed, the age at the monitor Δ_i is reset to the age of the original job itself, which is also the buffer age δ_i. Both the third and fourth jobs are skipped by the scheduler since the processor is busy serving other users.

We consider the scenario in which vehicles are moving with different velocities, and thus have different age requirements. Let α_i to be the weighting factor associated with the user i. Here we are interested in *sum age penalty functions* [7], which is a class of penalty functions defined as

$$\mathcal{P}_{\text{sum}}(t) = \sum_{i=1}^{n} \alpha_i f(\Delta_i(t)), \tag{4.31}$$

where $f : [0, \infty) \to \mathbb{R}$ is any non-decreasing penalty function for an individual user, which represents the dissatisfaction associated with information staleness. Since the penalty should be zero if the information is timely, we usually impose the initial condition

$f(0) = 0$. Some examples of penalty functions are:

1. The penalty is simply the age itself $f(\Delta) = \Delta$.
2. The exponential function of the age $f(\Delta) = e^{a\Delta} - 1$ where $a \geq 0$. This suits applications in which the need for information refresh is more desired as the information gets stale.
3. The fraction function $f(\Delta) = (a\Delta)/(a\Delta + b)$ that maps the age to the binary interval $[0, 1]$. This converts the age to a mission failure probability for many attack-defense problems and certain control applications [126].

The time-averaged penalty function is then defined as

$$\mathrm{E}[\mathcal{P}_{\mathrm{sum}}(t)] = \lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \int_{t=0}^{\mathcal{T}} \sum_{i=1}^{n} \alpha_i f(\Delta_i(t))\mathrm{d}t. \tag{4.32}$$

Our objective is to minimize time-averaged sum age penalty function over all n users by choosing the casual traffic scheduling policy π. We note that the scheduling policy π at any time t depend on the history of all prior states of the system, including:

1. the instant age at the monitors $\Delta_1(t), \Delta_2(t), \ldots, \Delta_n(t)$,
2. the generation time of j-th job $A_j^{(i)}$ from user i for all $j \leq N_i(t)$ up to the present time t.

4.2.2 Scheduling Policy

In order to describe the optimal scheduling policy, we first define the penalty reduction after the service by the processor. Assume that the processor becomes idle and there are at least one job waiting in the buffers $B_1, B_2, \ldots, B_n$ at time t_k, and it selects the job from user i for processing. Assuming this job is the k-th job at the processor and the service time is s time units, we observe in Figure 4.10 that the age reduction for user i after the processing time s is

$$D_i(t_k + s) = \Delta_i(t_k) - \delta_i(t_k). \tag{4.33}$$

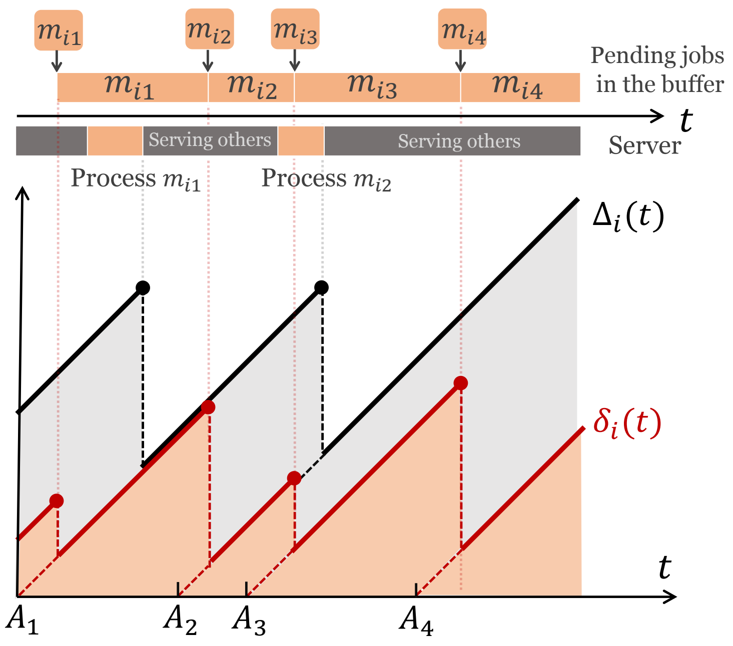


Figure 4.10: Sample path of age processes $\delta_i(t)$ and $\Delta_i(t)$ at the buffer and the user monitor.

We remark that the processor obeys a non-preemptive scheduling policy, in which the server doesn't preempt any job being serviced by new incoming jobs. Thus, the age reduction at time $t_k + s$ depends on the difference between the age at the user monitor and buffer age, both at time t_k. At time t_k, the most recent served and delivered update for user i is denoted by $M_i(t_k)$, which is generated at time $A^{(i)}_{M_i(t_k)}$. On the other hand, the job in the buffer at time t_k is generated at time $A^{(i)}_{N_i(t_k)}$. From Figure 4.10, we also observe that the age reduction in $\Delta_i(t_k + s)$ after service is

$$D_i(t_k + s) = A^{(i)}_{N_i(t_k)} - A^{(i)}_{M_i(t_k)}. \tag{4.34}$$

That is, the reduction in age at user i after the service is exactly the inter-arrival time between the two most recent served jobs. In Figure 4.10, the second job m_{i2} is selected for processing, and the third job m_{i3} arrives at the buffer during the processing period. After the service for m_{i2}, the reduction in age is $A_2 - A_1$ since the job m_{i2} relects the real-time information at time A_2. We also note that the job in the buffer is not necessarily the most recently generated one, since each job m_{ij} takes a random upload time U_{ij} to arrive at the buffer. Since the scheduler has direct access to the timestamp of the job $A_{N_i(t_k)}$ at each buffer i, the effect of the random upload time U_{ij} on the age reduction is not directly reflected in (4.34).

Since only one job can be processed at a time, only the age of the served user $\Delta_i(t_k+s)$ is reduced, and the age processes of all other users remain unchanged. Thus, the reduction of the sum age penalty function in (4.31) at time t_k+s is

$$R_i(t_k+s) = \alpha_i\Big(f\big(\Delta_i(t_k+s)\big) - f\big(\Delta_i(t_k+s) - D_i(t_k+s)\big)\Big). \tag{4.35}$$

Although the scheduling decision is made at time t_k, the reduction occurs at time t_k+s where the processor service time s is random and unknown to the scheduler. Thus, the scheduler knows the age reduction $D_i(t_k+s)$ in (4.34), but not the penalty reduction $R_i(t_k+s)$ in (4.35). We now consider the following greedy scheduling policy which is independent of the processor service time by assuming the processing can be completed instantaneously, i.e. $s=0$, and the reduction occurs immediately.

Definition 4.2.1. Maximum Immediate Penalty Reduction (MIPR) Policy. *When the server becomes available and the buffers are not empty at time t, the job from user i with the maximum penalty reduction $R_i(t)$ is served, with ties broken arbitrarily. That is, the scheduling indicator is*

$$\phi_k = \arg\max_i \alpha_i \left(f(\Delta_i(t_k)) - f(\Delta_i(t_k) - D_i(t_k))\right). \tag{4.36}$$

Intuitively. the greedy MIPR policy is optimal if the best decision made based on the current benefit still remains as the best decision in the future. This property brings some restrictions on the growth of the penalty function since the service time s is random.

Definition 4.2.2. *A penalty function f has **Base-Independent Growth (BIG)** if for any x and non-negative constant $s \geq 0$, there exists two penalty functions g_1 and g_2 such that*

$$f(x+s) = f(x)g_1(s) + g_2(s).$$

The definition of BIG states that the evolution of function f after s time units can

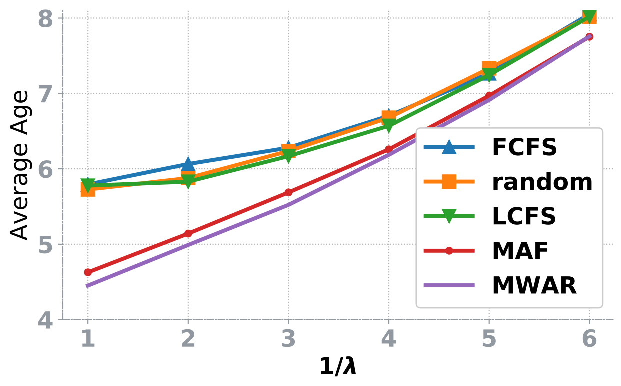


Figure 4.11: Average age vs. average job arrival time with different scheduling policies.

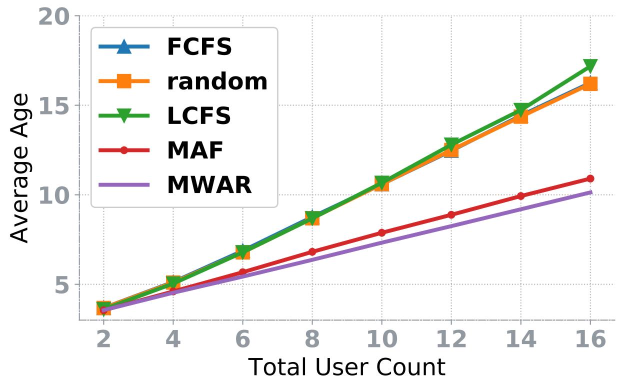


Figure 4.12: Average age vs. number of users with different policies.

be described by a multiplicative term $g_1(s)$ and an additive term $g_2(s)$, both depending on only s. Note that when the shift is $s = 0$, $f(x) = f(x)g_1(0) + g_2(0)$ holds for all x, which requires $g_1(0) = 1$ and $g_2(0) = 0$.

We note that $f(x) = e^{ax}$ is an example of the BIG penalty function since $f(x+s) = f(x)f(s)$. In step (4.42), the initial difference $f(x_1) - f(x_1 - d_1)$ is amplified by $f(s) = e^{as}$ as the time s increases. Similarly, $f(x) = e^{ax} - 1$ is also a BIG penalty function, since we can write

$$\begin{aligned} f(x+s) &= e^{a(x+s)} - 1 \\ &= \left(e^{ax} - 1\right) e^{as} + e^{as} - 1 \\ &= f(x)e^{as} + \left(e^{as} - 1\right). \end{aligned} \tag{4.37}$$

On the other hand, a linear function $f(x) = ax+b$ is another BIG penalty where $g_1(s) = 1$ and $g_2(s) = as$. In this example, $f(x_1 + s) - f(x_1 - d_1 + s) = f(x_1) - f(x_1 - d_1)$ only depends on the initial difference instead of the time difference s.

Theorem 4.2.3. *If the service times are identically distributed across all the jobs from all users, the MIPR policy is the optimal (1) causal, (2) work-conserving and (3) non-preemptive policy for BIG penalty function f, specifically*

$$\mathcal{P}_{\text{sum,MIPR}}(t) \leq_{st} \mathcal{P}_{\text{sum},\pi}(t), \tag{4.38}$$

for any $t \geq 0$ and any $\pi \in \Pi$, where $\leq_{st}$ is the stochastic ordering defined in [127].

It follows from Theorem 4.2.3 that

$$\mathrm{E}[\mathcal{P}_{\text{sum,MIPR}}] \leq \mathrm{E}[\mathcal{P}_{\text{sum},\pi}]. \tag{4.39}$$

The proof of Theorem 4.2.3 in the appendix follows the sample path technique used in [7]. One key idea used in the proof is the inductive comparison between two policies. By greedily choosing the user that gives the maximum penalty reduction, the instantaneous penalty after the service completion is always smaller than that in any other policy.

Definition 4.2.4. Maximum Weighted Age Reduction (MWAR) Policy. *When the server becomes available, the job from the user i with the maximum weighted age reduction $\alpha_i D_i(t_k)$ is served among all packets in the buffer, with ties broken arbitrarily.*

Corollary 4.2.5. *If the penalty function is $f(\Delta) = \Delta$, then MWAR is the age optimal MIPR policy.*

Corollary 4.2.5 follows directly from Theorem 4.2.3. In this special case, the scheduling policy is now independent of the current age of an individual user $\Delta_i(t_k)$. MWAR policy is also the maximum-age-first (MAF) policy in [7] if the job arrivals are synchronized among users.

4.2.3 Performance Evaluation

In this section, we evaluate the effectiveness and the fairness of the proposed maximum weighted age reduction (MWAR) policy by considering the average age over users $(1/n)\sum_{i=1}^{n}\alpha_i\Delta_i$ as the penalty function. Here we let all the users to be equally weighted, $\alpha_i = 1$ for all i. We compare the MWAR policy with four other policies with different system setup. The four reference policies are:

1. first-come-first-served (FCFS): the scheduler selects the job with earliest arrival time in the buffer;

2. last-come-first-served (LCFS): the scheduler selects the most recent arrived job in the buffer;

3. max-age-first (MAF): the scheduler selects the job corresponding to the user with maximum age;

4. random: the scheduler selects one of the jobs in the buffer uniformly at random.

Figure 4.11 compares the average age for each policy by fixing the processing rate $S = 1$ and varying the job arrival rate λ_i. The number of users is set to $n = 5$ and each user submits jobs according to Poisson process with average inter-update time $1/\lambda$. The service time is exponentially distributed and thus the average job processing time is $1/S = 1$. As the average inter-job submission time $1/\lambda$ increases, all the curves increase and the gap between any two policies becomes smaller. This is mainly because the age becomes dominated by the idle time between updates instead of the processing delay, and the scheduling doesn't provide much performance gain. Among all five policies, MWAR policy gives the lowest average age. And the MAF policy, which is shown to be optimal when the job arrival times are synchronized in [7], provides slightly larger average age. On the other hand, the other three policies (FCFS, LCFS and random) lead to almost the same much larger average age regardless of the job arrival rate.

Figure 4.12 depicts the comparison with server processing rate $S = 1$ and user job submission rate $\lambda_i = 1/2$ by varying the total number of users n. As n increases, the

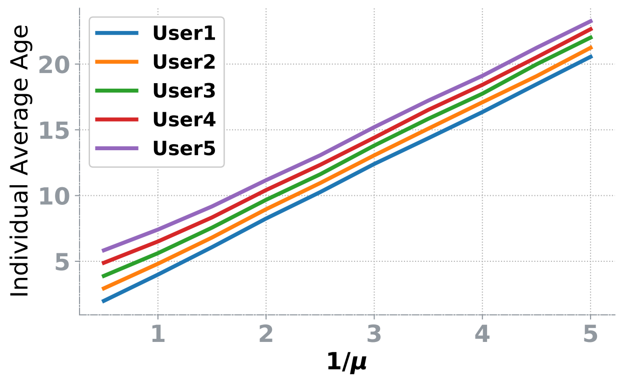


Figure 4.13: Average age of each user in MWAR policy.

processor becomes busier and thus MWAR provides larger performance gain. We also notice the average age grows almost linear as n increases.

While all the experiments in Figure 4.11 sets all the user update rates λ_i identical for all i. In Figure 4.13 and 4.14, we choose different job submission rate λ_i for each user i and evaluate how the scheduling policy treats users with different λ_i. Figure 4.13 depicts the average age of each user Δ_i by varying the processing rate S. The job submission rates for the $n = 5$ users are $\lambda_1 = 2, \lambda_2 = 1, \lambda_3 = 2/3, \lambda_4 = 1/2, \lambda_5 = 2/5$. As the average processing time $1/S$ increases, the individual average age increases almost linearly and the gap between any pair of users stays almost the same, which implies the MWAR policy keeps the difference between users regardless of the available resource of the service facility.

Figure 4.14 demonstrates the fraction of served job at the processor corresponding to each user. This is equivalent to the allocated rate for each source in Section 4.1. For example, when the average service processing time is $1/S = 1$, around 30% of jobs served by the processor are from user 1. When the processor is operating very fast, it can handle most of the jobs and thus the fraction of jobs is almost proportional to the rate of each user λ_i. As the processing time gets larger, we observe the scheduler starts to treat all users fairly and service in an equal way. Since every user experience long waiting time when the traffic load is high, the age for each user is almost equally large. As a result, the scheduler is busy serving every user one by one as soon as it finishes an old job.

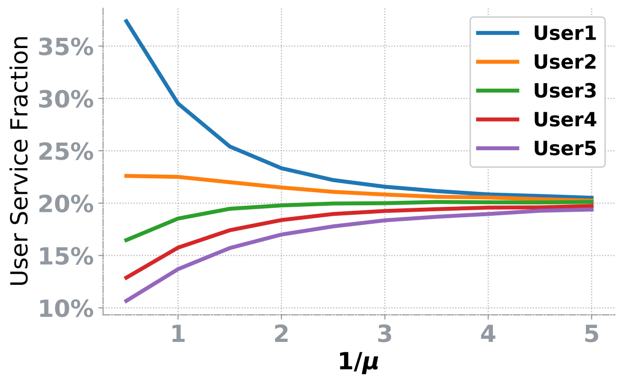


Figure 4.14: The fraction of service corresponding to each user in MWAR policy.

4.2.4 Future Work

Since this is our initial step to examine the resource scheduling system at the edge cloud, the abstract model is oversimplified, and the resulting scheduling policy is straightforward but difficult to be generalized. We believe this work can be extended in the following directions:

1. The most difficult part of the scheduling problem is to handle jobs with asynchronous arrivals since those jobs with different time-stamps give different age reduction or penalty reduction in general. We've proved that the greedy policy is optimal for the weighted sum of the average age over all users, but this conclusion does not hold for most other penalty functions. Our next step should be the study of other scheduling policies, especially Index form policies [26, 128], for other types of penalty functions.

2. The computational tasks of visual applications usually depends on the amount of information contained in an image/frame and the amount of new objects needed to be updated in the point cloud [129, 130], and thus a more realistic model should categorize the jobs into different types with different computational resource requirements and various service time distributions.

3. Our current effort models the edge cloud as a whole computing unit that processes incoming jobs sequentially. However, the edge cloud is expected to be a cluster that consists of massive number of heterogeneous computing units that allows

tenants to perform distributed and parallel computing.

4.2.5 Appendix: Proof of Theorem 4.2.3.

We denote P as the MIPR policy and $\mathcal{P}_{\text{sum},\pi}(t)$ as the penalty function of policy π at time t. We will compare P and any other work-conserving policy $\pi \in \Pi$ on a sample path of $\mathcal{P}_{\text{sum}}(t)$.

For any sample path in policy P and π, we set the initial ages $\Delta_{i,P}(t=0) = \Delta_{i,\pi}(t=0)$ for users $i = 1, 2, \ldots, n$. The initial penalties are $\mathcal{P}_{\text{sum},P}(t=0) = \mathcal{P}_{\text{sum},\pi}(t=0)$. The system evolution is described by the following cases:

1. If no update completes in $[t', t'+s]$, the age process of every user $\Delta_i(t) = \Delta_i(t') + (t - t')$ for $t \in [t', t'+s]$.
2. If there is an update completion at time t, the age of the served user $\Delta_i(t)$ is reduced.

Now we define the following class of penalty functions.

Definition 4.2.6. *Function f is a* ***Present-Determines-Future (PDF)*** *function if f satisfies the following conditions:*

1. *If a pair of n-tuple sequences $\{x_{1i}\}, \{x_{2i}\}$ and non-negative n-tuple constants $\{\alpha_i\}$ satisfy $\sum_{i=1}^n \alpha_i f(x_{1i}) \le \sum_{i=1}^n \alpha_i f(x_{2i})$, then for any $s \ge 0$,*

$$\sum_{i=1}^{n} \alpha_i f(x_{1i} + s) \le \sum_{i=1}^{n} \alpha_i f(x_{2i} + s). \tag{4.40}$$

2. *If x_1, x_2 and non-negative constants $\beta \ge 0, d_1 \ge 0, d_2 \ge 0$ satisfy*

$$f(x_1) - f(x_1 - d_1) \ge \beta\big(f(x_2) - f(x_2 - d_2)\big),$$

   then for any $s \ge 0$,

$$\begin{aligned} &f(x_1 + s) - f(x_1 - d_1 + s) \\ &\qquad \ge \beta\big(f(x_2 + s) - f(x_2 - d_2 + s)\big). \end{aligned}$$

Lemma 4.2.7. *If a penalty function f is BIG, then f is PDF.*

Proof. We need to show BIG f satisfies both conditions of PDF. For condition 1),

$$\begin{aligned}\sum_{i=1}^{n}\alpha_i f(x_{1i}+s) &= g_1(s)\sum_{i=1}^{n}\alpha_i f(x_{1i})+\sum_{i=1}^{n}\alpha_i g_2(s)\\ &\le g_1(s)\sum_{i=1}^{n}\alpha_i f(x_{2i})+\sum_{i=1}^{n}\alpha_i g_2(s)\\ &=\sum_{i=1}^{n}\alpha_i f(x_{2i}+s). \end{aligned}\tag{4.41}$$

Similarly, condition 2) is met as follows

$$\begin{aligned}&f(x_1+s)-f(x_1-d_1+s)\\ =&\Big(f(x_1)g_1(s)+g_2(s)\Big)-\Big(f(x_1-d_1)g_1(s)+g_2(s)\Big)\\ =&\Big(f(x_1)-f(x_1-d_1)\Big)g_1(s)\\ \ge&\beta\Big(f(x_2)-f(x_2-d_2)\Big)g_1(s) \qquad (4.42)\\ =&\beta\Big(f(x_2+s)-f(x_2-d_2+s)\Big). \qquad (4.43)\end{aligned}$$

□

Now we can start the proof of Theorem 4.2.3 by the following lemma about the first case with no update completion.

Lemma 4.2.8. *For PDF f, if $\mathcal{P}_{\text{sum},P}(t)\le\mathcal{P}_{\text{sum},\pi}(t)$ and there is no update completion between t and $t+s$, then*

$$\mathcal{P}_{\text{sum},P}(t+s)\le\mathcal{P}_{\text{sum},\pi}(t+s).\tag{4.44}$$

The proof follows directly from the condition 1) in the definition of PDF by setting $x_{1i}=\Delta_{iP}(t)$ and $x_{2i}=\Delta_{i\pi}(t)$ for $i=1,2,\ldots,n$. Note that Lemma 4.2.8 guarantees that given the sum age penalty in policy P is smaller than that in policy π at some time t, the same ordering holds for any time beyond t if there is no update completion.

Now we move to the second case with an update completion at time $t_k + s$. Whether under policy P or policy π, an update is serviced from time t_k to $t_k + s$. The penalty of policy P is $\mathcal{P}_{\text{sum},P}$ before the completion and becomes $\mathcal{P}'_{\text{sum},P}$ after the completion. Similarly, the penalty of policy π is $\mathcal{P}_{\text{sum},\pi}$ before the completion and becomes $\mathcal{P}'_{\text{sum},\pi}$ after the completion. All policies have the same update arrival process and service process. We first prove the following lemma about inductive comparison between two sample paths.

Lemma 4.2.9. *For PDF function f, if $\mathcal{P}_{\text{sum},P} \le \mathcal{P}_{\text{sum},\pi}$, then $\mathcal{P}'_{\text{sum},P} \le \mathcal{P}'_{\text{sum},\pi}$.*

Proof. When job k is to go into service at time t_k, the MIPR policy P chooses the user $\phi_k = \arg\max_i R_i(t_k)$. At the service completion time $t_k + s$,

$$\mathcal{P}'_{\text{sum},P}(t_k + s) = \mathcal{P}_{\text{sum},P}(t_k + s) - \max_i R_i(t_k). \tag{4.45}$$

By the MIPR policy, choosing user i yields larger immediate penalty reduction than choosing user j, $R_i(t_k) \ge R_j(t_k)$ and

$$\begin{aligned} &f(\Delta_i(t_k)) - f(\Delta_i(t_k) - D_i(t_k)) \\ \ge&\frac{\alpha_j}{\alpha_i}\big[f(\Delta_j(t_k)) - f(\Delta_j(t_k) - D_j(t_k))\big] \end{aligned} \tag{4.46}$$

Since the age reductions D_i and D_j are independent of the service time s, by the definition of a PDF function we have

$$\begin{aligned} &f(\Delta_i(t_k + s)) - f(\Delta_i(t_k + s) - D_i(t_k + s)) \\ &= f(\Delta_i(t_k + s)) - f(\Delta_i(t_k + s) - D_i(t_k)) \\ &\ge \frac{\alpha_j}{\alpha_i}\big[f(\Delta_j(t_k{+}s)){-}f(\Delta_j(t_k{+}s){-}D_j(t_k))\big] \\ &= \frac{\alpha_j}{\alpha_i}\big[f(\Delta_j(t_k{+}s)){-}f(\Delta_j(t_k{+}s){-}D_j(t_k{+}s))\big]. \end{aligned} \tag{4.47}$$

Hence, the penalty reduction at time $t_k + s$ is $R_i(t_k + s) \ge R_j(t_k + s)$. Thus,

$$\arg\max_i R_i(t_k) = \arg\max_i R_i(t_k + s). \tag{4.48}$$

The penalty of any policy π after the service completion is

$$\begin{aligned}\mathcal{P}'_{\text{sum},\pi}(t_k+s) &= \mathcal{P}_{\text{sum},\pi}(t_k+s) - R_i(t_k+s)\\ &\geq \mathcal{P}_{\text{sum},\pi}(t_k+s) - \max_i R_i(t_k+s)\\ &\geq \mathcal{P}_{\text{sum},P}(t_k+s) - \max_i R_i(t_k+s)\\ &= \mathcal{P}'_{\text{sum},P}(t_k+s). \qquad (4.49)\end{aligned}$$

□

Now given that the penalty function evolves under the condition of either Lemma 4.2.8 and 4.2.9, by induction over time, we have $\mathcal{P}_{\text{sum},P}(t) \leq \mathcal{P}_{\text{sum},\pi}(t)$, for all $t \geq 0$. And thus

$$\mathrm{E}[\mathcal{P}_{\text{sum},P}(t)] \leq \mathrm{E}[\mathcal{P}_{\text{sum},\pi}(t)]. \qquad (4.50)$$

for any casual work-conserving policy π.

Chapter 5

Conclusions

5.1 Summary

The emergence of ubiquitous connectivity and pervasive computing have engendered lots of real-time applications in which the interested recipients keep track of the information status at the remote sources. This is achieved by the transmission of status updates between the source and destination through some communication networks. In this thesis, we formulated the status updating problem in different network applications, and evaluated the corresponding age of information (AoI) metrics.

In Chapter 2, we studied the lossless source coding problem on arriving status updates in favor of both the average AoI and the average peak AoI. In streaming source coding, we converted the AoI minimization problem to an existing codebook optimization problem with penalties depending on moments of the encoded sequence length. For deterministic symbol arrivals, we showed that the age benefits from short blocklength, which is essentially different from conventional lossless compression results. For random symbol arrivals, the codebook design involves some special techniques to encode the idleness at the source.

In Chapter 3, we extended the AoI analysis to updating systems with multiple destinations. Each update is replicated and distributed to multiple nodes using multicast, and the source controls when to terminate the transmission of the current update and generate a new one. We showed that the average age performance at the receiver, which has access to a subset of nodes, can be significantly improved by allowing the source to carefully choose how long it waits for each update transmission. The system model we considered is applicable to a wide range of applications, including live content distribution to the edge cloud, and content updates in quorum-type distributed storage.

In Chapter 4, we examined two updating systems with shared resource at the service facility. We first looked at a remote cache updating system in which the local server maintains snapshots of the content at different remote sources and refreshes these snapshots with a total rate limit. The AoI metric was compared to an alternative AoS metric. We showed that the the optimal rate allocation policies for the two different age metrics coincide under some circumstances. The second application we investigated is the edge cloud computational offloading system shared by multiple users. The AoI-optimal scheduling policy for incoming jobs at the edge server was proved to be a greedy policy that always processes the job with the maximum age reduction.

5.2 Future Work

There are numerous interesting open problems associated with information freshness as captured by the age, mainly because of the importance of the age being a performance metric for massive real-time applications. Our effort in this thesis only revealed a small portion of them. Moreover, some limitations in our work also point to several new directions for future research.

For the streaming source coding problem, we have proposed a near-optimal coding scheme that outperforms Huffman codes, which minimizes the average codelength only. However, we also numerically observed that the advantage is very minor in some cases, especially when a large blocklength is used. We believe this is related to limitations on the codelength variance for large source alphabet, which requires more thorough understanding of the binary tree structure for prefix-free codes.

When we study the job scheduling problem at the edge cloud, we assume the entire edge cloud to be a single powerful computing unit, which acts as a proxy for the complicated modern edge systems. As what we discussed at the end of Chapter 4, an interesting research direction is to relax this assumption and investigate whether greedy policies stay optimal for edge systems that is equipped with heterogeneous computing units and processing multi-type jobs.

References

[1] Cisco Visual Networking Index, "Global mobile data traffic forecast update, 2017-2022 white paper," *Cisco: San Jose, CA, USA*, 2017.

[2] C. Cookson, "Time is money when it comes to microwaves," *Financial Times*, vol. 10, 2013.

[3] S. Kaul, R. D. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. INFOCOM*, Apr. 2012, pp. 2731–2735.

[4] M. Costa, M. Codreanu, and A. Ephremides, "Age of information with packet management," in *Proc. IEEE Int. Symp. Inform. Theory*, 2014, pp. 1583–1587.

[5] Y. Sun, E. Uysal-Biyikoglu, R. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," in *Proc. INFOCOM*, 2016.

[6] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "The cost of delay in status updates and their value: Non-linear ageing," *arXiv preprint arXiv:1812.09320*, 2018.

[7] Y. Sun, E. Uysal-Biyikoglu, and S. Kompella, "Age-optimal updates of multiple information flows," *arXiv preprint arXiv:1801.02394*, 2018.

[8] A. Garnaev, W. Zhang, J. Zhong, and R. D. Yates, "Maintaining information freshness under jamming," in *IEEE INFOCOM AoI Workshop*, 2018.

[9] S. K. Kaul, R. D. Yates, and M. Gruteser, "Status updates through queues." *Conf. on Inf. Sciences and Systems*, pp. 1–6, 2012.

[10] R. D. Yates and S. Kaul, "Real-time status updating: Multiple sources," in *Proc. IEEE Int. Symp. Inform. Theory*, Jul. 2012, pp. 2666–2670.

[11] C. Kam, S. Kompella, and A. Ephremides, "Age of information under random updates," in *Proc. IEEE Int. Symp. Inform. Theory*, Jul. 2013, pp. 66–70.

[12] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *Proc. IEEE Int. Symp. Inform. Theory*, Jun. 2015, pp. 1681–1685.

[13] E. Najm and R. Nasser, "Age of information: The gamma awakening," in *Proc. IEEE Int. Symp. Inform. Theory*, 2016, pp. 2574–2578.

[14] E. Najm, R. D. Yates, and E. Soljanin, "Status updates through M/G/1/1 queues with HARQ," in *Proc. IEEE Int. Symp. Inform. Theory*, 2017.

[15] E. Najm and E. Telatar, "Status updates in a multi-stream M/G/1/1 preemptive queue," in *IEEE INFOCOM AoI Workshop*, 2018, pp. 124–129.

[16] Y. Inoue, H. Masuyama, T. Takine, and T. Tanaka, “A general formula for the stationary distribution of the age of information and its application to single-server queues,” *arXiv preprint arXiv:1804.06139*, 2018.

[17] R. D. Yates and S. K. Kaul, “The age of information: Real-time status updating by multiple sources,” *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1807–1827, 2018.

[18] R. D. Yates, “The age of information in networks: Moments, distributions, and sampling,” *arXiv preprint arXiv:1806.03487*, 2018.

[19] S. Farazi, A. G. Klein, and D. R. Brown, “Average age of information for status update systems with an energy harvesting server,” in *IEEE INFOCOM AoI Workshop*, 2018, pp. 112–117.

[20] ——, “Age of information in energy harvesting status update systems: When to preempt in service?” in *Proc. IEEE Int. Symp. Inform. Theory*, 2018, pp. 2436–2440.

[21] R. D. Yates, “Lazy is timely: Status updates by an energy harvesting source,” in *Proc. IEEE Int. Symp. Inform. Theory*, Jun. 2015, pp. 3008–3012.

[22] A. M. Bedewy, Y. Sun, and N. B. Shroff, “Optimizing data freshness, throughput, and delay in multi-server information-update systems.” *Proc. IEEE Int. Symp. Inform. Theory*, 2016.

[23] ——, “Age-optimal information updates in multihop networks,” in *Proc. IEEE Int. Symp. Inform. Theory*, 2017, pp. 576–580.

[24] R. Talak, S. Karaman, and E. Modiano, “Minimizing age-of-information in multi-hop wireless networks,” in *Proc. Allerton Conf. on Commun., Control and Computing*, 2017, pp. 486–493.

[25] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, “Minimizing the Age of Information in broadcast wireless networks.” *Proc. Allerton Conf. on Commun., Control and Computing*, pp. 844–851, 2016.

[26] P. Whittle, “Multi-armed bandits and the gittins index,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 42, no. 2, pp. 143–149, 1980.

[27] Y.-P. Hsu, “Age of information - Whittle index for scheduling stochastic arrivals,” *Proc. IEEE Int. Symp. Inform. Theory*, 2018.

[28] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, “Scheduling policies for minimizing age of information in broadcast wireless networks,” *IEEE/ACM Trans. on Networking*, no. 6, pp. 2637–2650, 2018.

[29] Y. Sun and B. Cyr, “Information aging through queues: A mutual information perspective,” in *IEEE Int. Workshop on Signal Proc. Advances in Wireless Commun (SPAWC)*, 2018, pp. 1–5.

[30] Y. Sun, Y. Polyanskiy, and E. Uysal-Biyikoglu, “Remote estimation of the Wiener process over a channel with random delay,” in *Proc. IEEE Int. Symp. Inform. Theory*, 2017, pp. 321–325.

[31] S. Bhambay, S. Poojary, and P. Parag, “Differential encoding for real-time status updates,” in *IEEE Wireless Commun. and Networking Conf. (WCNC)*, 2017, pp. 1–6.

[32] R. D. Yates, E. Najm, E. Soljanin, and J. Zhong, “Timely updates over an erasure channel,” in *Proc. IEEE Int. Symp. Inform. Theory*, 2017.

[33] S. Gopal and S. K. Kaul, “A game theoretic approach to DSRC and WiFi coexistence,” in *IEEE INFOCOM AoI Workshop*, 2018, pp. 565–570.

[34] Waze. [Online]. Available: https://www.waze.com/

[35] B. Li and J. Liu, “Can we achieve fresh information with selfish users in mobile crowd-learning?” *arXiv preprint arXiv:1902.06149*, 2019.

[36] J. Liu, X. Wang, B. Bai, and H. Dai, “Age-optimal trajectory planning for UAV-assisted data collection,” in *IEEE INFOCOM AoI Workshop*, 2018, pp. 553–558.

[37] A. Kosta, N. Pappas, and V. Angelakis, “Age of information: A new concept, metric, and tool,” *Foundations and Trends® in Networking*, vol. 12, no. 3, pp. 162–259, 2017.

[38] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, “Dynamo: Amazon’s highly available key-value store,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 205–220, 2007.

[39] C. E. Shannon, “A mathematical theory of communication,” *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[40] D. A. Huffman, “A method for the construction of minimum-redundancy codes,” *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[41] R. Gallager, “Variations on a theme by huffman,” *IEEE Trans. Inf. Theory*, vol. 24, no. 6, pp. 668–674, 1978.

[42] J. Ziv and A. Lempel, “Compression of individual sequences via variable-rate coding,” *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[43] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression,” *Communications of the ACM*, vol. 30, no. 6, pp. 520–541, 1987.

[44] P. A. Humblet, “Source coding for communication concentrators,” Ph.D. dissertation, Massachusetts Institute of Technology, 1978.

[45] ——, “Generalization of Huffman coding to minimize the probability of buffer overflow,” *IEEE Trans. Inf. Theory*, 1981.

[46] J. F. Kingman, “Inequalities in the theory of queues,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 32, no. 1, pp. 102–110, 1970.

[47] A. D. Wyner, “On the probability of buffer overflow under an arbitrary bounded input-output distribution,” *SIAM Journal on Applied Mathematics*, vol. 27, no. 4, pp. 544–570, 1974.

[48] C. Flores, "Encoding of bursty sources under a delay criterion." Ph.D. dissertation, University of California, Berkeley, 1984.

[49] L. L. Larmore, "Minimum delay codes," *SIAM Journal on Computing*, 1989.

[50] L. L. Larmore and D. S. Hirschberg, "A fast algorithm for optimal length-limited huffman codes," *Journal of the ACM (JACM)*, vol. 37, no. 3, pp. 464–473, 1990.

[51] M. B. Baer, "Coding for general penalties," Ph.D. dissertation, Stanford University, 2003.

[52] ——, "Source coding for quasiarithmetic penalties," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4380–4393, 2006.

[53] O. Shayevitz, R. Zamir, and M. Feder, "Bounded expected delay in arithmetic coding," in *Proc. IEEE Int. Symp. Inform. Theory*, 2006, pp. 2662–2666.

[54] C. Cheng and A. Sahai, "The error exponent with delay for lossless source coding," in *IEEE Inf. Theory Workshop*, Mar. 2006, pp. 252–256.

[55] S. C. Draper, C. Chang, and A. Sahai, "Lossless coding for distributed streaming sources," *IEEE Trans. Inf. Theory*, vol. 60, no. 3, pp. 1447–1474, 2014.

[56] L. Zhou, V. Y. F. Tan, and M. Motani, "On error exponents and moderate deviations for lossless streaming compression of correlated sources," *arXiv preprint arXiv:1507.03190*, 2015.

[57] N. Ma and P. Ishwar, "On Delayed Sequential Coding of Correlated Sources," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3763–3782, Jun. 2011.

[58] P. Mayekar, P. Parag, and H. Tyagi, "Optimal lossless source codes for timely updates," in *Proc. IEEE Int. Symp. Inform. Theory*, 2018.

[59] J. Zhong and R. D. Yates, "Timeliness in lossless block coding," in *Data Compression Conf. (DCC)*, 2016, pp. 339–348.

[60] J. Zhong, R. D. Yates, and E. Soljanin, "Backlog-adaptive compression: Age of information," in *Proc. IEEE Int. Symp. Inform. Theory*, 2017, pp. 566–570.

[61] ——, "Timely lossless source coding for randomly arriving symbols," in *2018 IEEE Information Theory Workshop (ITW)*, 2018, pp. 1–5.

[62] L. D. Servi, "D/G/1 queues with vacations," *Operations Research*, vol. 34, no. 4, pp. 619–629, 1986.

[63] K. Marshall and R. V. Evans, "Some inequalities in queuing," *Operations Research*, pp. 651–668, 1968.

[64] C. Chang and A. Sahai, "Delay-constrained source coding for a peak distortion measure," in *Proc. IEEE Int. Symp. Inform. Theory*, 2007, pp. 576–580.

[65] Z. G. Zhang and N. Tian, "Discrete time Geo/G/1 queue with multiple adaptive vacations," *Queueing Syst.*, vol. 38, no. 4, pp. 419–429, 2001.

[66] T. Meisling, "Discrete-time queuing theory," *Operations Research*, vol. 6, no. 1, pp. 96–105, Jan. 1958.

[67] H. Kobayashi and A. Konheim, "Queueing models for computer communications system analysis," *IEEE Trans. Commun.*, vol. 25, no. 1, pp. 2–29, January 1977.

[68] M. Heindlmaier and E. Soljanin, "Isn't hybrid ARQ sufficient?" in *Proc. Allerton Conf. on Commun., Control and Computing*, 2014, pp. 563–568.

[69] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, 2014.

[70] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5G ultra-dense cellular networks," *arXiv preprint arXiv:1512.03143*, 2015.

[71] A. Gupta and R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE access*, vol. 3, pp. 1206–1232, 2015.

[72] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[73] J. Choi, V. Va, N. Gonzalez-Prelcic, R. Daniels, C. R. Bhat, and R. W. Heath, "Millimeter-wave vehicular communication to support massive automotive sensing," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 160–167, 2016.

[74] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE communications surveys & tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[75] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through internet of things," *IEEE Internet of Things journal*, vol. 1, no. 2, pp. 112–121, 2014.

[76] M. Condoluci, G. Araniti, T. Mahmoodi, and M. Dohler, "Enabling the IoT machine age with 5G: Machine-type multicast services for innovative real-time applications," *IEEE Access*, vol. 4, pp. 5555–5569, 2016.

[77] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[78] M. Satyanarayanan, "The emergence of edge computing," *IEEE Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[79] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[80] M. Sardari, R. Restrepo, F. Fekri, and E. Soljanin, "Memory allocation in distributed storage networks," in *Proc. IEEE Int. Symp. Inform. Theory*, 2010, pp. 1958–1962.

[81] D. Leong, A. G. Dimakis, and T. Ho, "Distributed Storage Allocations," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4733–4752, Jul. 2012.

[82] M. Noori, E. Soljanin, and M. Ardakani, "On storage allocation for maximum service rate in distributed storage systems," in *Proc. IEEE Int. Symp. Inform. Theory*, 2016, pp. 240–244.

[83] P. Peng and E. Soljanin, "On distributed storage allocations of large files for maximum service rate," in *Proc. Allerton Conf. on Commun., Control and Computing*, 2018, pp. 784–791.

[84] J. Postel, "Internet control message protocol," 1981.

[85] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 26: Frame Preemption," *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)*, pp. 1–52, Aug 2016.

[86] "IEEE Standard for Ethernet," *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)*, pp. 1–5600, Aug 2018.

[87] J. Zhong, E. Soljanin, and R. D. Yates, "Status updates through multicast networks," in *Proc. Allerton Conf. on Commun., Control and Computing*, 2017, pp. 463–469.

[88] J. Zhong, R. D. Yates, and E. Soljanin, "Multicast with prioritized delivery: How fresh is your data?" in *IEEE Int. Workshop on Signal Proc. Advances in Wireless Commun (SPAWC)*, 2018, pp. 1–5.

[89] ——, "Minimizing content staleness in dynamo-style replicated storage systems." in *IEEE INFOCOM AoI Workshop*, 2018, pp. 361–366.

[90] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic service placement in geographically distributed clouds," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 762–772, 2013.

[91] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2016.

[92] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, "Towards efficient edge cloud augmentation for virtual reality mmogs," in *Proceedings of ACM/IEEE Symp. on Edge Computing (SEC)*, 2017.

[93] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[94] A. Shokrollahi, M. Luby *et al.*, "Raptor codes," *Foundations and trends in communications and information theory*, vol. 6, no. 3–4, pp. 213–322, 2011.

[95] C. Lott, O. Milenkovic, and E. Soljanin, "Hybrid ARQ: Theory, state of the art and future directions," in *IEEE Info. Theory Workshop on Info. Theory for Wireless Networks*, 2007, pp. 1–5.

[96] B. Buyukates, A. Soysal, and S. Ulukus, "Age of information in two-hop multicast networks," in *Asilomar Conf. on Signals, Systems, and Computers*, 2018, pp. 513–517.

[97] ——, "Age of information in multihop multicast networks," *arXiv preprint arXiv:1812.10455*, 2018.

[98] Y. Sang, B. Li, and B. Ji, "The power of waiting for more than one response in minimizing the age-of-information," in *IEEE Global Comm. Conf. (Globecom)*, 2017, pp. 1–6.

[99] G. Yadgar, O. Kolosov, M. F. Aktas, and E. Soljanin, "Modeling the edge: Peer-to-peer reincarnated," in *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*, Renton, WA, 2019.

[100] A. Spiegelman, Y. Cassuto, G. Chockler, and I. Keidar, "Space Bounds for Reliable Storage," in *ACM Symp. on Principles of Distributed Computing*, New York, New York, USA, 2016, pp. 249–258.

[101] Z. Wang and V. R. Cadambe, "Multi-version coding - an information-theoretic perspective of consistent distributed storage," in *IEEE Trans. Inf. Theory*, 2017.

[102] D. Blackwell and M. Girshick, "On functions of sequences of independent chance vectors with applications to the problem of the "random walk" in k dimensions," *The Annals of Mathematical Statistics*, vol. 17, no. 3, pp. 310–317, 1946.

[103] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja, *A first course in order statistics.* SIAM, 2008.

[104] K. Fujimoto, S. Ata, and M. Murata, "Statistical analysis of packet delays in the internet and its application to playout control for streaming applications," *IEICE Transactions on Communications*, vol. 84, no. 6, pp. 1504–1512, 2001.

[105] X.-L. Zhang and P. Liu, "A new delay jitter smoothing algorithm based on pareto distribution in cyber-physical systems," *Wireless Networks*, vol. 21, no. 6, pp. 1913–1923, Aug 2015.

[106] M. F. Aktas and E. Soljanin, "Straggler mitigation at scale," *arXiv preprint arXiv:1906.10664*, 2019.

[107] H. Attiya, A. Bar-Noy, and D. Dolev, "Sharing memory robustly in message-passing systems," *J. ACM*, pp. 1–12, 1995.

[108] D. Peleg and A. Wool, "The availability of quorum systems," *Information and Computation*, vol. 123, no. 2, pp. 210–223, 1995.

[109] D. Malkhi, M. Reiter, and R. Wright, "Probabilistic quorum systems," in *Proceedings of ACM Symposium on Principles of Distributed Computing*, 1997, pp. 267–273.

[110] D. Malkhi and M. Reiter, "Byzantine quorum systems," *Distributed Computing*, vol. 11, no. 4, pp. 203–213, 1998.

[111] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica, "Probabilistically bounded staleness for practical partial quorums," *Proceeding of VLDB*, 2012.

[112] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, Apr. 2010.

[113] W. Golab, X. Li, and M. A. Shah, "Analyzing consistency properties for fun and profit," in *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing(PODC)*, New York, NY, USA, 2011, pp. 197–206.

[114] M. R. Rahman, L. Tseng, S. Nguyen, I. Gupta, and N. Vaidya, "Characterizing and adapting the consistency-latency tradeoff in distributed key-value stores," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 11, no. 4, pp. 1–36, 2017.

[115] J. Zhong, R. D. Yates, and E. Soljanin, "Two freshness metrics for local cache refresh," in *Proc. IEEE Int. Symp. Inform. Theory*, 2018, pp. 1924–1928.

[116] J. Zhong, W. Zhang, R. D. Yates, A. Garnaev, and Y. Zhang, "Age-aware scheduling for asynchronous arriving jobs in edge applications," in *IEEE INFOCOM AoI Workshop*, 2019.

[117] J. Cho and H. Garcia-Molina, "Effective page refresh policies for Web crawlers." *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 390–426, 2003.

[118] K. C. Sia, J. Cho, and H.-K. Cho, "Efficient monitoring algorithm for fast news alerts," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 950–961, May 2007.

[119] X. Li, D. B. H. Cline, and D. Loguinov, "On sample-path staleness in lazy data replication," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2858–2871, Oct. 2016.

[120] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Information freshness and popularity in mobile caching." in *Proc. IEEE Int. Symp. Inform. Theory*, 2017, pp. 136–140.

[121] R. D. Yates, P. Ciblat, A. Yener, and M. A. Wigger, "Age-optimal constrained cache updating." in *Proc. IEEE Int. Symp. Inform. Theory*, 2017, pp. 141–145.

[122] P. Jain, J. Manweiler, and R. Roy Choudhury, "Low bandwidth offload for mobile AR," in *ACM International Conf. on emerging Networking EXperiments and Technologies (CoNEXT)*, 2016, pp. 237–251.

[123] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 155–168.

[124] W. Zhang, S. Li, L. Liu, Z. Jia, Y. Zhang, R. Yates, and D. Raychaudhuri, "Hetero-edge: Orchestration of real-time visionapplications on heterogeneous edge clouds," in *Proc. INFOCOM*, 2019.

[125] H. B. Beytur and E. Uysal-Biyikoglu, "Minimizing age of information for multiple flows," *2018 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1–5, Jun. 2018.

[126] A. Garnaev, M. Baykal-Gursoy, and H. V. Poor, "Security games with unknown adversarial strategies," *IEEE Trans. on Cybernetics*, vol. 46, no. 10, pp. 2291–2299, 2016.

[127] M. Shaked and J. G. Shanthikumar, *Stochastic orders.* Springer Science & Business Media, 2007.

[128] J. C. Gittins, "Bandit processes and dynamic allocation indices," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 41, no. 2, pp. 148–164, 1979.

[129] A. Arsalan Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum, "Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1511–1519.

[130] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," in *Computer Graphics Forum*, vol. 36, no. 1. Wiley Online Library, 2017, pp. 301–329.