Stochastic Models for Cloud Data Backups and Video Streaming on Virtual Reality Headsets

Stochastische modellen voor cloud-data-backups en videostreaming op virtual-reality-headsets

Apoorv Saxena

Promotoren: prof. dr. D. Claeys, prof. dr. ir. J. Walraevens
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen

Vakgroep Industriële Systemen en Productontwerp
Voorzitter: prof. dr. E.-H. Aghezzaf

Vakgroep Telecommunicatie en Informatieverwerking
Voorzitter: prof. dr. ir. J. Walraevens

Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2020 - 2021

UNIVERSITEIT
GENT

ISBN 978-94-6355-414-5
NUR 919, 986
Wettelijk depot: D/2020/10.500/91

Members of Examination Committee:

Chairman:

prof. em. Hendrik Van Landeghem	Ghent University

Promoters:

prof. Dieter Claeys	Ghent University
prof. Joris Walraevens	Ghent University

Reading Committee:

prof. Rob van der Mei	Vrije Universiteit Amsterdam Centrum Wiskunde & Informatica
dr. Thomas Demoor	Western Digital
prof. Nele Noels	Ghent Univesity
prof. Dieter Fiems	Ghent University

Acknowledgements

Undertaking this PhD has been a life changing experience for me which would not have been possible without the guidance and support of many people.

I would like to thank my supervisors Prof. Dieter Claeys and Prof. Joris Walraevens for their guidance on these research ideas. The success of this PhD is possible because of their help, guidance, and the environment they maintain where we can pursue our own research challenges freely. Working with them has helped me pursue my research interests and challenge myself.

I am grateful towards Prof. Dieter Fiems, Prof. Herwig Bruneel, Prof. Sabine Wittevrongel and all other professors at TELIN who have created an atmosphere which has led to a lot of successful projects and PhDs over the past few years. A big thanks to the department secretaries Patrick and Sylvia, and the help-desk Philippe and Davy for always ensuring all the things were running and their help with our local problems.

I would like to thank FWO for funding a part of this project. Many thanks to Prof. Rob van der Mei at CWI and VUA and Prof. Hans van den Berg at TNO and UTwente for giving me the opportunity to work in their research group over the past year. I would also like to thank the faculty at IIT Kanpur, especially Prof. Amit Mitra and Prof. Parasar Mohanty at the Mathematics & Statistics department, for their help and guidance in my career.

I would also like to thank my colleagues Kishor, Freek, Katia, Hossein, Sara and Mustafa for the snooker sessions, squash games and all the fun activities at TELIN. Thanks to Gianni, Martin, Ana, Ivana, Maarten, Nina and Zuhaib for all the interesting lunch discussions. Special thanks to Arnaud for his help in improving my dutch and being my taalmoeder.

Thanks to my mom, my brother and other family members for their constant love and support. My wife Anchal is one of the primary reasons why I moved to live in this amazing city. Thank you for your love and support, and for always keeping me grounded. Looking forward to a lifetime of happiness with you. Thanks to my wingies from IIT Kanpur, especially Braj, who have always been like brothers to me. Thanks to Tim for being a friend/running coach and Lieselot who has been like our family in Belgium.

Lastly, I would like to thank my father, Mr Sundeep Saxena, who is unfortunately not with us today. Thank you for helping me understand the importance of family and being grateful for everything in life. You will always be present in my thoughts.

Contents

List of Figures

Samenvatting

We gebruiken onze smartphone om berichten te verzenden en foto's te nemen, we verzenden en ontvangen e-mails, onze activity tracker registreert onze dagelijkse activiteit en beweging, ... Deze en vele andere apparaten die in ons leven geïntegreerd zijn, creëren continu nieuwe data. Een deel van deze data, zoals e-mails en foto's, kunnen voor ons echt waardevol zijn. We willen er dan ook voor zorgen dat deze niet verloren gaan. Daarom hebben veel mensen een back-up van hun bestanden op een harde schijf of in de cloud, via diensten zoals Dropbox en iCloud. Opslag in de cloud heeft verschillende voordelen. Enerzijds kan je bestanden openen op elk toestel dat verbonden is met het internet, aan de hand van een login. Anderzijds zijn bestanden in de cloud ook beschermd tegen lokale fouten, zoals het falen van je harde schijf. Bestanden opslaan in de cloud heeft natuurlijk ook een prijs. Opslag tot 200GB op iCloud kost ongeveer 3$ per maand.

In een organisatie zijn er, naast de activiteiten van haar werknemers, veel meer mogelijkheden voor het genereren van nieuwe data. Interne e-mails, documenten, contracten, conversaties en de details van alle producten moeten worden opgeslagen voor eventueel later gebruik of voor een audit. In veel organisaties moeten zelfs de telefoongesprekken en videochats worden opgeslagen voor bewakingsdoeleinden. Deze bedrijven kunnen ook het gebruik van hun producten registreren om betere services te bieden. Zo zouden Amazon en Coolblue bijvoorbeeld bijhouden naar welke producten hun gebruikers kijken op hun webpagina en op andere webpagina's, om aan de hand van deze informatie gerichte advertenties te tonen. Nu moet al deze data ergens worden opgeslagen waar het gemakkelijk toegankelijk is én beschermd is tegen lokale storingen. De waarde van deze nieuwe entiteit 'data' is de afgelopen decennia erkend en er wordt gezegd dat data het goud van deze generatie is. Veel grote bedrijven zoals Google en Facebook genereren het grootste deel van hun inkomen door de gegevens van hun gebruikers te gebruiken voor advertenties. In dit proefschrift kijken we naar een veilige opslag van deze waardevolle entiteit tegen lage kosten. Analoog zou je kunnen zeggen dat we kijken naar enkele aspecten van het voorraadbeheer van het goud van onze generatie.

In dit proefschrift onderzoeken we de prestatie van data back-ups op cloud services aan de hand van stochastische modellen. Gebruikers genereren nieuwe data die nadien worden overgedragen en opgeslagen in de vorm van pakketten. Deze pakketten moeten zo snel mogelijk na het creëren ervan

opgeslagen worden. De stochastische modellen omvatten de onzekerheden van onderliggende componenten, zoals de hoeveelheid tijd die nodig is om de opslagoperaties uit te voeren en de onzekere manier waarop nieuwe data gegenereerd wordt. Dit proefschrift heeft voornamelijk als doel om de performantie van deze opslagoperaties te evalueren, en om de afhankelijkheid tussen de verschillende parameters van dataopslag te kwantificeren. Het berekenen van deze prestatiematen helpt ons in het optimaliseren van de performantie van de operaties waarbij we verzekeren dat ze gedaan worden tegen een degelijke kost.

Naast het bestuderen van data back-ups, analyseren we ook de performantie van streaming algoritmes op Virtual Reality (VR) headsets. VR headsets worden gebruikt om gebruikers een meeslepende digitale beleving te geven wanneer ze (bijvoorbeeld) naar een film kijken of een videospel spelen. In dit proefschrift analyseren we het gedrag van gebruikers die kijken naar een 360^0 video en stellen we een Quality of Experience (QoE) bewust algoritme voor. Hierbij wordt enkel dat deel van de video in het gezichtsveld van de gebruiker, genoemd de Field of View, verzonden met de hoogste kwaliteit. Het algoritme dat we voorstellen houdt rekening met de onzekerheden bij het voorspellen van toekomstige posities van het hoofd van de gebruiker, evenals met de onbetrouwbaarheid van het netwerk. Het algoritme selecteert dan de meest geschikt sectie van de video om te verzenden. Het voornamelijkste doel van het algoritme is het vooraf ophalen van de toekomstige video secties wanneer de netwerk condities goed zijn, om zo de impact van de netwerk variaties te verminderen.

Beide onderzoeksonderwerpen omvatten een gemeenschappelijk thema, namelijk het meten van de prestatie en het garanderen van een hoge QoS/QoE, rekening houdend met de onzekerheden van de onderliggende processen en de bijhorende kosten.

Na een inleidend hoofdstuk ontwerpen we in Hoofdstuk 2 een stochastisch model voor het opslaan van data op cloud services. We includeren het 'batch service' karakter van de datatransmissie en de onzekerheid van de bedieningstijden die vereist is om inkomende data te verwerken binnen het model. Met behulp van dit model, kunnen we de distributie van de QoS prestatiematen berekenen zoals de age of data en de backlog size. Aan de hand van deze resultaten formuleren we vervolgens een optimalisatie probleem om de optimale opslagparameters te berekenen. Het doel van dit optimalisatieprobleem is om de kost voor het uitvoeren van deze operaties te minimaliseren. De QoS eisen zoals de limiet op de backlog size en de age of data, zijn geformuleerd als de beperkingen van dit optimalisatie probleem.

In Hoofdstuk 3 analyseren we in meer detail het gedrag van een van de meest cruciale prestatiematen: de age of data. Na het berekenen van de

probabiliteitsgenererende functie (PGF) van de age of data, gebruiken we de dominante singulariteiten analyse om meer inzicht te krijgen in de prestatie van dataopslag systemen. Tenslotte gebruiken we deze resultaten om een de optimale trigger tijd te berekenen voor opslagsystemen met een ingebouwd trigger mechanisme.

Sommige oplossingen leggen strenge beperkingen op de age of data, en het verzekeren van de age of data binnen de opgelegde grenzen kan net cruciaal zijn voor QoS objectieven. Daarom onderzoeken we in Hoofdstuk 4 een model met een drempelwaarde op de tijd sinds de vorige back-up. Het model bestudeerd in Hoofdstuken 2-3 en het model in Hoofdstuk 4 hebben enkele belangrijke verschillen die leiden tot verschillende voordelen, gebaseerd op de wensen van de opslag provider. Bijvoorbeeld, het model bestudeerd in Hoofdstuk 4 omvat een drempelwaarde op de tijd sinds de vorige back-up binnen het model. Het model bestudeerd in Hoofdstuken 2-3 daarentegen, houdt rekening met (probabilistische) herstart-voorwaarden en heeft een lagere computationele complexiteit. Met behulp van beide modellen zijn we in staat om de sensitiviteit te meten van de prestatie van de opslagoperaties op de opslagparameters die het back-up schema bepalen.

In Hoofdstuk 5 stellen we een streaming algoritme op VR headsets voor. We doen een experiment waar we het gedrag bijhouden van een gebruiker tijdens het bekijken van een 360^0 videos op een headset. Met behulp van deze analyse bouwen we een hoofdbeweging voorspellingsmodel. Voorts wordt het draadloos netwerk waarover de video wordt verzonden gemodelleerd als een Markov proces. Elke toestand in het Markov proces definieert de beschikbare bandbreedte, alsook de prestatiekost in die condities. Dit algoritme, gebaseerd op een Markov Decision Process (MDP), bepaalt welk videofragment moet worden gedownload om een hoge QoE voor de gebruikers te garanderen tegen een redelijke kost. Het voornamelijkste doel van dit werk is de balans bewaren tussen het resource verbruik (netwerk en batterij) en de QoE (kwaliteit van de video voor gebruikers), en het voorstellen welk stuk van de video te downloaden om te zorgen dat deze balans wordt behouden.

Summary

Most of our daily activities result in the generation of new data in some way. Our mobile phones are used to send messages and click pictures, we send and receive emails, our health devices log our activities and exercises, and many other devices which have integrated in our lives, generate data. Some of this data such as emails and photos might be really valuable to us and we would want to ensure that it is not lost. Therefore, many individuals keep a backup of their files on hard drives or on a cloud service such as Dropbox, iCloud etc. Storage using a cloud service has an advantage that those files can be accessed by login from any device connected to the internet. It also ensures safety from local failures, that is, if your hard disk fails all the files stored on it are lost permanently. In cloud storage, files are replicated to ensure safety from such failures. However, these storage features come at a price, for instance storage of 200GB on iCloud comes at a price of about 3$ per month.

In an organisation, in addition to the activities of its employees, there are many more avenues of new data generation. Internal emails, documents, client contracts and conversations as well as the details of all the products have to be stored for possible later use or an audit. In many organisations, even the phone calls and video chats have to be stored for monitoring purposes. These businesses may also log the usage of their products to provide better services. For example, websites like Amazon and Coolblue track what their users view on their webpage as well as other webpages they visit on the internet to show targeted advertisement. Now all this data needs to be stored somewhere where it is easily accessible and safe from local failures. The value of this new entity 'data' has been recognized in the past decades and it is said that data is the new oil of this generation. Many big corporations such as Google, Facebook generate a majority of their income by using data of their users for advertisement. In this dissertation, we are looking at a safe storage of this valuable entity at a low cost. Analogously, one could say we are looking at some aspects of the inventory management of the new oil of our generation.

In this dissertation, we investigate the performance of data backups on cloud storage using stochastic models. New data is generated by the users which is transferred and stored in the form of packets. Moreover, these data packets should be stored as quickly as possible after creation. The models capture the uncertainties in the underlying components such as the amount of time it takes to complete storage operations as well as the randomness in

the way new data is generated which needs to be stored. The main goal of this dissertation is to evaluate the performance of these backup operations and quantify their dependence on the different parameters of data storage. Computing these measures helps us in optimizing the performance of these operations while ensuring that they are done at a reasonable cost.

In addition to data backups, we also investigate the performance of streaming algorithms on Virtual Reality (VR) headsets. VR headsets are used to provide an immersive digital experience to its users ranging from watching movies, gaming to virtual tours. In this dissertation, we analyze the behavior of users watching a 360^0 movie and propose a Quality of Experience (QoE) aware streaming algorithm. Videos streamed on these headsets are transmitted such that only the portion of video in the view of the user, called the Field of View, is in the highest quality. The algorithm we propose takes into account the uncertainties of predicting the future head positions of a user as well as the unreliability of the network. The algorithm then proposes the most appropriate section of the video for future transmission. The main goal is to pre-fetch the expected future video sections when the network conditions are good to mitigate the impact of network variations.

Both these research topics have a common theme, which is to compute the performance and ensure a high QoS/QoE while taking into account the unreliability of the underlying processes and their costs.

In the first part, we build a stochastic model of data storage using cloud storage services. We include the batch service nature of data transmission, and the uncertainty of the service time required to serve incoming data within the model. Using this model, we are able to compute the distribution of the Quality of Service (QoS) measures of the data storage processes such as age of data and backlog size. Further, using these results we formulate an optimization problem to compute the optimal storage parameters. In this optimization problem, the objective is to minimize the cost of performing these operations. Moreover, the QoS requirements such as a limit on backlog size and age of data are formulated as the constraints of this problem.

We further investigate the behavior of one of the crucial performance measures, the age of data, in more depth. After computing the probability generating function (PGF) of this measure, we use dominant singularity analysis to gain insight into the performance of storage systems. We then utilize these results to compute the optimal time trigger for storage processes which have an inbuilt trigger mechanism.

Some storage solutions enforce strict limits on the age of data, and ensuring the age of data is within the limits may be crucial to their QoS objectives. Therefore, in further chapters we investigate a model where a

threshold on the time since the last backup is included within the model. The model studied in chapter 2-3 and chapter 4 have some key differences which provide different advantages based on the aim of the storage provider. For instance, the model studied in chapter 4 includes a threshold on the time since last backup within the model. While, the model studied in chapter 2-3 implements probabilistic restarting conditions and has a much smaller computational complexity. Using either of these models, we are able to compute the sensitivity of the performance of storage operations on the storage parameters which determine the backup schedule.

In the last part of the dissertation, we propose a streaming algorithm on VR headsets. We perform an experiment where we log the user behavior while watching 360^0 videos on a headset. Using this analysis we build a head movement prediction model. Further, the wireless network over which the videos are transmitted is modelled as a Markov process where each state defines the available bandwidth as well as the cost of performing operations in those conditions. This Markov Decision Processes (MDP) based algorithm determines which part of the video should be downloaded to ensure the user experiences high service quality at a reasonable cost. The main goal of this work is to ensure a balance between the resource usage (battery and network) and Quality of Experience (Quality of the video shown to the user) and to propose which part of the video to download to ensure this balance is maintained.

1

Introduction

Until a couple of decades ago, setting up a new business, which required an IT branch, was an extremely challenging task. The whole setup of infrastructure had to be implemented independently by each company and maintained individually using their own resources and man power. In 2006 the whole industry was revolutionized when the concept of cloud infrastructure was introduced. This infrastructure service was introduced by Amazon as Amazon Elastic compute cloud (Amazon EC2) service [6] which offered infrastructure for computing on demand. With the vision of cloud services, in particular IAAS (Infrastructure As A Service) (see [7]), this service was opened to other companies for a fee. In a short period, this division became a major revenue generator for Amazon. For instance, in the fourth quarter of 2019, Amazon Web Services (AWS) [8] generated 9.95 Billion $ in sales for Amazon, about 34% higher than previous year [9]. In the past few years, Microsoft, Google, IBM and other companies have introduced their own platforms for such services. However, Amazon still accounts for about 33% of the market share of the whole industrial revenue, about twice the size of the second largest provider Microsoft Azure [10]. Other big corporations such as IBM [11] also offer their platforms for IAAS.

1.1 Overview of cloud services industry

Overtime, this cloud services industry has grown into a 325.1 Billion $ industry in 2018 and expected to grow up to 528 Billion $ by 2022 [12]. With the introduction of this new technology, setting up an IT infrastructure has become quick, easy and cost effective. The companies can now just enter into a service contract with these providers and use standard protocols to run and manage their operations. Most importantly, the cloud service providers guarantee a high Quality of Service (QoS) as a part of the contract. To ensure that such high QoS standards are maintained, they actively manage and maintain the whole cloud infrastructure internally. A downtime/failure of the setup can result in heavy economic losses for such companies, for example see [13]. Therefore, keeping the infrastructure running is of utmost priority to them. Cloud management is therefore one of the top researched

areas of computing, for example see [14, 15, 16]. For a company, it is very convenient and reassuring that they do not have to worry about the intricacies of the infrastructure management and can focus on their business development. Moreover, their engineers do not have to spend days and weeks reinventing the wheel by setting up the infrastructure locally for their company.

One of the challenging aspects of today's world is the reduction of green house gas emission. It has been estimated that about 2% of the green house gas emission is generated from the ICT industry. The US Environment protection Agency also estimates that about 1.5% of the total power produced in the US was used for Data center computing [17]. Therefore, by pooling the resources and having a common infrastructure setup improves the utilization of resources which leads to reduction of energy usage. Studies such as [18] have analyzed that the requirement of data storage technologies will grow exponentially and may grow to as high as 40 ZB by 2020 from 1.8 ZB in 2011. With this rapid increase machines need to schedule data backup processes efficiently and cost effectively.

By pooling the resources, the cloud service providers are able to provide some remarkable benefits of data safety which would not be possible with local storage (see [19]). Some of the main ideas of this technology include

- *Replication* : Companies want to ensure that the data they store is safe. To avoid data loss in case of a failure, multiple copies of data are maintained on different machines across different hard drives and tapes as a safeguard mechanism. That is, when an image is stored, a couple of more copies of this image will be stored on different hard drives. Data centers are able to provide this replication at a much higher safety level and security. Since they are located at multiple locations across different countries, the extra copies of data can be stored on different continents entirely to provide safety in case of a natural calamity.
- *Ease of setup* : Customers are charged a fee based on the space they use and how often these services are used. To access the data stored on the cloud infrastructure, standard protocols have been implemented to do a very wide range of tasks and operations which fulfil most of the industry requirements.
- *Service on demand* : Data centers consist of many small machines operating from a single area. It would be unwise to keep all these machines running at all times. Therefore, the number of machines in operation is determined by the ongoing demand. This feature truly captures the essence of resource pooling which improves the resource utilization of all the machines. By switching off servers when there are few users, the energy consumption of operating the servers and cooling is reduced.
- *Availability* : Customers can decide how critical it is for them to be able to access the data all the time. Since all the machines are not

operational all the time, all the data may not be accessible 100% of the time. Therefore, customers are given an option to select the availability from 95%, 99%, 99.99% etc. The common terminology used is the number of 9s of availability, i.e., five 9 availability implies data is available for at least 99.99999% of the time [20]. The data center then has to manage its operations that it can ensure this level of availability in the long run.

- *Cost of usage* : As stated earlier, these companies charge their customers for the amount of space utilized (amount of data stored). Additionally, they also charge for the number of times their infrastructure is requested for a new operation.

In the past few years, organizations have moved from local storage systems to cloud storage systems. Moreover, new businesses are building their infrastructure using the services of public cloud system providers. The security provided by data replication at geographical level is something most small or medium enterprises can not afford to implement. Amazon publishes case studies listing its users, [21], which contains some well known new companies such as Coursera [22], Netflix [23], Airbnb [24], Spotify [25]. It also contains some well established companies such as General Electric [26], Hitachi [27], NASA [28], SAP [29] and Ubisoft [30]. This range and variety of users of cloud services of Amazon alone highlights the scale at which cloud storage is being adopted by the industry in a span of less than a decade. Therefore, the majority of the data generated in future is expected to be stored on cloud storage infrastructure (see [31]).

Cloud storage industry was worth US$25.171 billion in 2017 and is expected to be worth US$92.488 billion in 2022 [32]. This growth has been driven by the huge volume of data that is being generated every day (see [33]). Recently, studies have shown that this amount is continuing to grow exponentially. A study by the international data corporation (see [34]) has estimated that the amount of data generated would reach 163ZB by 2025 which is approximately 10 times the data generated in 2016. Industries have started adopting public clouds for storage and operations. Therefore, it is important to study and analyze cloud infrastructure systems and processes.

Running operations on the cloud comes with a lot of ease and speed. However, one has to be careful of the costs involved. It is cheaper to run operations on the cloud than setting up their own infrastructure for a medium sized enterprise. However, it still costs a few million dollars to run such operations. In this dissertation we focus on the cloud backups which alone is a multi-Billion $ industry [32]. As stated earlier, the companies are not only charged for the amount of data stored, but also for the number of times the cloud infrastructure is requested for a new operation. Therefore, to be able to run operations smoothly and at a reasonable cost, it is critical to design and schedule these operations. At the same time we need to ensure a good QoS at this reasonable cost.

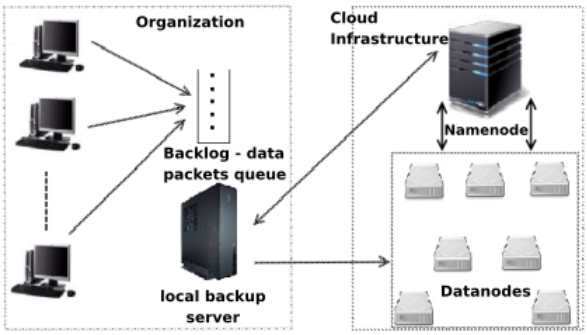


Figure 1.1: Components of the backup process

1.2 Data backups

A cloud data backup process in an organization involves several components as illustrated in Figure 1.1. This setup is commonly used across the industry to perform their cloud data backups, such as backup services offered by company [35]. Generally, the cloud infrastructure services are provided by an external organization such as [8, 10]. Data is stored in form of packets and the backup process as a whole include moving these packets on the infrastructure of cloud service provider as well as creating a replica of the packets. Performing data backups utilizes three major components, one on the side of the organization, and two on the cloud service provider.

- *Backup server:* The organization which rents the computing services of the data center, has a dedicated node to perform the communication and backup operations for the whole organization. Therefore, all the computers and storage nodes communicate to the cloud service provider through this node. When data is available to be uploaded to cloud, this backup server decides when a backup operation needs to be done. An operation request is sent to the cloud infrastructure when an operation is desired by the backup server.

- *Namenode:* The namenode manages all the requests that need to be run on the cloud infrastructure, including the requests from the backup servers. This node monitors the health of the infrastructure and all the components involved. If a failure is detected, necessary operations are taken by the name node. For instance, if it is detected that a storage node has failed which increases the risk of data loss, the namenode initiates a restoration of all the data stored on that node onto other disks. Such actions are crucial to ensure replication and availability standards of the cloud infrastructure.

- *Storage nodes:* These nodes are the actual storage points of the data packets requested to be stored. These storage nodes have to be arranged

in a topology which has to be accounted for in the storage of replicas. This is done to reduce the risk of data loss due to common failures. For example, say six storage nodes are stored on a single rack which is powered by a single source. In case of power failure, all the six nodes will be lost. Therefore, replicas of data stored on the disks of this rack should be stored on a another rack with a different power source.

Briefly put, when the data backup server wants to store data, it sends a write request to the name node of the cloud service. The name node knows the health of all the storage nodes, their remaining capacity as well as their topology. By looking at this information, the name nodes selects the appropriate storage nodes and sends their IP addresses to the data backup server. The backup server can then upload data to those storage nodes. As data is written to these nodes, creation of replicas of these data packets is initiated in the background under the governance of namenode. Data packets are said to have been successfully written once the data packets as well as their replicas have been successfully written to storage nodes.

Note: Not all data centers/ cloud service providers use replications to ensure data safety. A lot of them use Reed Solomon coding (RS coding) or similar coding techniques to reduce the storage overhead. In these methodologies, a few parity packets are generated using the original data packets and stored on different machines. As long as a minimum number of packets, defined by the coding method used, are not lost, the original data can be restored. For example, if RS 9/13 coding is used, then when 9 packets are stored, 4 extra parity packets are generated. The original 9 packets can always be obtained as long as any of the 9 packets out of the total 13 packets can be obtained. This reduces the storage overhead because instead of storing 3 times the number of total packets, we now have to store only about 1.44 times the total number of packets in the above example. The final storage overhead in a general backup scenario would depend on the choice of coding method used.

1.2.1 Costs involved

Our primary aim is to model and optimize the performance of the backup server, therefore we will focus on its operations and the costs incurred due to these operations. The backup server does three main operations on the cloud server. It decides when to perform backup operations and sends the operations requests to the namenode of the cloud server. It also performs the transfer of data packets to the storage nodes. The cloud service provider charges both for the number of operation requests the name node receives as well as the amount of data stored on the data nodes. However, these are only the economic costs involved. These backup operations have to be scheduled and operated such that the security of data packets is ensured. That is, the probability of data loss is as low as guaranteed in the Service Level Agreements (SLAs). Therefore, the data backup server has to perform backup operations

regularly to keep the data loss probability very low.

On one hand the backup frequency should be high enough to keep the probability of data loss low. On the other hand, these backup operations come at an economic cost of sending operation requests to the name node and keeping cloud resources occupied. There is also an economic cost of keeping the backup server active to perform these operations. There is a clear trade-off that needs to be addressed here. In this dissertation, we will model the data backups as a queueing model to numerically compute the two sides of trade-off. In order to achieve this, we will define and compute the following performance indicators

1. *Age of data:* When data packets wait for the backup server to perform the operations, they are at a risk of loss in case of immediate failure. This probability of failure is higher if the packets have to wait for long periods for backup service. Therefore, we define the performance measure *age of data* focused on the data packet which arrives in an empty system. Precisely, it is defined as the time this data packet has to wait until it receives backup service. The backup system alternates between on and off periods with off periods much longer than the on periods. Therefore, the first packet which arrives in an empty system has the longest waiting time to receive the service. By focusing on this packet, we measure the worst performance that a data packet can receive during the backup process. By enforcing QoS limits for this packet, we can truly say that each packet receives high QoS. Such a claim may not be true if we the analysis was focused on the QoS of any general packet. In the section 1.2.2 we will introduce Recovery point objective which will highlight the importance of this performance indicator.

2. *Backlog size:* Packets which have not received backup service wait for the backup server and become a part of the backlog/queue. A bigger backlog implies that more data packets are at risk of loss. We will compute the characteristic properties of this backlog size such as the long-run average number of packets in the backlog.

3. *Probability a new connection is established with the name node:* As defined earlier, each new connection to the namenode comes at an additional cost. We are able to compute the probability of a new connection at a random time epoch, which represents the frequency of new connections initiated.

4. *Probability server is busy:* Backup server usage comes at a cost of energy. Therefore, keeping it on continuously is unwise. The server is therefore switched to sleep mode when there are no packets to serve. Moreover, if the backup server is kept on for long periods, it may keep the cloud resources occupied for longer period than necessary. Therefore, by computing this probability, we are able to capture the resource usage of the backup server.

Note: It is important to note that the data packets written to the storage nodes are immutable. Therefore, if a file is changed and needs to be backed up, all the packets associated with that file would need to be rewritten. This means that the number of connections to the name node and the total amount of data sent to the store node is much higher than the net data stored.

1.2.2 Recovery point objective

Recovery point objective (RPO) is one of the most crucial parameters defined in the literature of data backup processes. Recovery point is defined as the time point in history such that the system is recoverable up to that point in case of an immediate disaster. The RPO gives us a bound on the age of the recovery point i.e. the difference between current time and recovery point (see [36] for more details).

Since the backup server is put into idle mode when it has no data packets to serve, it alternates between service and idle periods. A consecutive idle and service period together forms a data backup cycle. RPO defines a bound on the amount of time any data packet can wait during a data backup cycle. In a stable system, the data packet which arrives first in a data backup cycle suffers maximum wait for restart of backup process. In paragraph 2.4.4, we therefore compute the waiting time of this data packet as ensuring bounds on this waiting time is crucial to satisfy the RPO.

Note that in this dissertation we focus primarily on the QoS measures of the storage processes. For telecommunication services, it is known that the Quality of Experience (QoE) of user is closely related to QoS of the network. [37] show that for services such as cloud storage processes, QoE and QoS are connected together by a generic formula. Therefore, the QoS metrics of cloud storage processes can be quickly translated into QoE metrics if required.

1.3 Uncertainties associated

The primary aim of the backup process is to ensure the safety of data packets at all times. To ensure this safety, data backup operations are performed frequently. This keeps the number of data packets waiting for service, as well as the time they have to wait for service as low as guaranteed in SLAs. However, new data packets can arrive at any instant and have to be served within a reasonable time frame. As we will see in the next chapters, to maximise the resource usage data packets are served in batches. These properties lead to an uncertainty of when the packets actually get serviced from the time of arrival which leads to an uncertainty about the performance of backup processes.

Moreover, once the backup server starts serving these data packets to the cloud, the amount of time required to finish all the operations depends on the internal jobs of the cloud architecture which are not visible to the outside

user. This time is also dependent on the placement of the storage nodes chosen as well as the number of jobs operating on the cloud infrastructure. Additionally, the backup server goes into idle mode when there are no data packets to serve. This off-period is important to save the cost of running the backup server. We will assume this off-period has a known probability distribution. This assumption is made to ensure that different backup operations do not clash or synchronize together which might cause sudden spikes in the traffic at the name node. Similar techniques are used in WiFi routers to avoid such clashes [38].

1.4 Virtual Reality headsets

In chapter 5, we work on a new research topic, streaming of 360^0 videos on Virtual Reality (VR) headsets. This work was done in collaboration with Prof. Rob van der Mei and Prof. Hans van den Berg at CWI, Amsterdam. To maintain the legibility of the thesis for readers, we will provide only a brief overview of this topic. All the necessary details will follow up in chapter 5.

Over the past few years, VR headsets have gained tremendous popularity and this interest is expected to increase even further in the near future. The potential applications of this technology range from education [39], museum tours [40], medical training industry [41] to gaming industry [42]. We are primarily interested in the use of VR headsets to stream 360^0 videos, which is commonly used due to the powerful immersive experience it provides to the users [43]. Such an immersive experience of high quality, however, requires a high bandwidth connection. 5G networks will be able to provide high network bandwidth required to operate these headsets.

A VR headset requires high reliability and robustness against the fluctuations of the channel quality since the latency requirements are between 6 - 15 ms. A latency higher than 15 ms not only degrades the viewing quality, it also leads to issues like motion sickness [44]. These bandwidth and latency requirements have stretched our abilities to provide network services. Moreover, due to the increasing popularity, big companies like Facebook and YouTube have invested heavily in VR streaming. Therefore, it is expected that the usage will increase even further as VR headsets integrate into the life of regular users. This increase in the number of users will put even more stress on the network. Therefore, in chapter 5, we model and analyze the streaming of 360^0 videos on a VR headset over a wireless connection as an Markov Decision Process (MDP) problem. Our focus is to build a streaming algorithm which can optimizes the QoE for the users and resource usage during the streaming of these videos.

1.5 Methodology

In this section we will discuss the mathematical tools used in this dissertation. The exact use of these tools will be discussed in the upcoming chapters.

1.5.1 Markov processes

Before we have a look at Markov processes, we will briefly introduce a more general class of processes called stochastic processes. A stochastic process $\underline{X} = \{X_n, n \in T\}$ is a collection of random variables. That is, for each n in the index set T, X_n is a random variable. In this dissertation, we will work with discrete time stochastic processes, that is, T is a countable set and is an index in time. For example, X_n could represent the number of data packets waiting for backup service at time instant n.

A stochastic process qualifies to be a Markov process if the state of the system in the future depends only on the current state. Consider a discrete-time stochastic process X_n, then for this to be Markov process,

$$P(X_{n+1} = x|X_n, X_{n-1}, \ldots, X_0) = P(X_{n+1} = x|X_n),$$

for any value of $x \in \mathbb{A}$ and n. Here $\mathbb{A}$ is the state space of the process. If the size of the state space $\mathbb{A}$ is finite or countable, the Markov process is called a Markov chain.

1.5.2 Discrete-Time Markov Chains (DTMC)

Consider a discrete-time stochastic process $\{X_n, n \in T\}$ where the index set T and the state space is finite or countable. If

$$\begin{aligned} &P\{X_{n+1} = i_{n+1}|X_n = i_n, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0\} \\ &= P\{X_{n+1} = i_{n+1}|X_n = i_n\}, \end{aligned}$$

holds for any integral n, and the states $i_{n+1}, i_n, \ldots, i_0$, then X_n is said to be a discrete-time Markov chain (DTMC). If further we have,

$$P\{X_{n+m+1} = j|X_{n+m} = i\} = P\{X_{n+1} = j|X_n = i\}, \forall i, j \in \mathbb{A}, \forall m, n \geq 0,$$

the Markov chain is time homogeneous or stationary. We will deal only with this class of DTMC in this dissertation. Therefore, in general we say

$$p_{i,j} = P\{X_{n+1} = j|X_n = i\},$$

is the transition probability from state i at time n to state j at time $n + 1$.

1.5.2.1 State of DTMC

At each successive step, we can obtain the probability of being in a particular state of DTMC using the probability rule as

$$P\{X_{n+1} = i\} = \sum_{k \in \mathbb{A}} P\{X_{n+1} = i | X_n = k\} P\{X_n = k\}.$$

Define $\mathbf{x}^{(n)} = [x_0^{(n)}, x_1^{(n)}, \ldots]$ as the probability vector describing the state of the system at time n, where $x_i^{(n)}$ is the probability of the system being in state i at time n, where $n \geq 0$. We have,

$$x_i^{(n+1)} = \sum_{k \in \mathbb{A}} x_k^{(n)} p_{k,i},$$

which can be transformed into a matrix form as

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} \mathbf{P} = \mathbf{x}^{(0)} \mathbf{P}^n,$$

where $\mathbf{P} = [p_{k,i}]$ is the probability transition matrix of the Markov chain.

1.5.2.2 Chapman Kolmogorov equations

The (i, j)th value in P^n, $p_{i,j}^{(n)}$ represents the probability that the system is in j^{th} state after n transitions, given that it started from state i. Therefore, we have

$$\begin{aligned} p_{i,j}^{(2)} &= \sum_{k \in \mathbb{A}} p_{i,k} p_{k,j}, \\ p_{i,j}^{(n)} &= \sum_{k \in \mathbb{A}} p_{i,k}^{(n-1)} p_{k,j} = \sum_{k \in \mathbb{A}} p_{i,k} p_{k,j}^{(n-1)}, \\ p_{i,j}^{(n)} &= \sum_{k \in \mathbb{A}} p_{i,k}^{(m)} p_{k,j}^{(n-m)}. \end{aligned}$$

1.5.2.3 Classification of states

States of a Markov chain can be classified into different types depending on the transition probabilities $p_{i,j}$.

Communicating states: State j is accessible from state i if $\exists n > 0 : p_{i,j}^{(n)} > 0$. State i and j are communicating states if both are accessible to each other.

Absorbing states: A state is said to be an absorbing state if once the system enters state i it never leaves it. Precisely, state i is absorbing if $p_{i,i} = 1$.

Transient and Recurring states:
Define,

$$f_{i,i} = P\{X_n = i, \text{ for some } n \geq 1 | X_0 = i\}.$$

$f_{i,i}$ computes the probability that the system ever returns to state i.
If $f_{i,i} = 1$, the state i is a recurrent state, otherwise if $f_{i,i} < 1$, the state i is a transient state.

Periodic states: A state i is said to be periodic if the number of steps required to return to it is a multiple of a, $(a > 1)$. Here the period, a, of state i is defined as $gcd\{n : p_{i,i}^{(n)} > 0\}$.
If a state has period equal to 1, it is said to be aperiodic.

1.5.2.4 Classification of Markov chains

- Irreducible Chain: A Markov chain is said to be irreducible if all the states communicate with each other.
- Absorbing Chain: A Markov chain is said to be an absorbing chain if any of its states is an absorbing state, i.e., $\exists i$ such that $p_{i,i} = 1$.
- Recurrent Markov Chain: A Markov chain is said to be recurrent if all the states of the chain are recurrent. Recurrent states are further classified into null recurrent and positive recurrent states.
- Null and Positive recurrent Markov chains: Define $\tau_{i,i}$ as the amount of time required to return to state i given the starting state is i, then $E(\tau_{i,i})$, the expected value of $\tau_{i,i}$ classifies whether the state i is positive or null recurrent.
  If $E(\tau_{i,i}) < \infty$, the recurrent state i is a positive recurrent state.
  Else if $E(\tau_{i,i}) = \infty$, the recurrent state i is null recurrent state.
- Ergodic Markov Chain: An irreducible Markov chain is said to be ergodic if all its states are aperiodic and positive recurrent.

1.5.2.5 Limiting behavior of Markov chains

One key feature of time-homogeneous DTMCs is their behaviour after a long period of time, i.e., their limiting behaviour.

Let P be the transition matrix of an ergodic DTMC and the probability vector of the initial state is given by $\mathbf{x}^{(0)}$, then the probability vector that defines the state of the system at any time n is given by

$$\mathbf{x}^{(n)} = \mathbf{x}^{(n-1)}P = \mathbf{x}^{(0)}P^n.$$

After a long time, $n \to \infty$ we have $\mathbf{x}^{(n)} \to \mathbf{x}$ and therefore $\mathbf{x}^{(n)} \to \mathbf{x}P$, which is known as the stationary probability of the system.

Note: For a non-ergodic system, $\mathbf{x}$ may not be equal to the limiting probability vector of the system even if both exist. That is, for a non-ergodic system even if a $\mathbf{y}$ exists such that

$$P^n \to \begin{bmatrix} \mathbf{y} \\ \mathbf{y} \\ \vdots \\ \mathbf{y} \end{bmatrix}, \text{ as } n \to \infty$$

$\mathbf{x}$ may not be equal to $\mathbf{y}$.

Mean first recurrence time and steady state distributions

For an ergodic system in steady state, the probability of being in state i is x_i. The mean recurrence time $M_{i,i}$ for state i is related to x_i as

$$x_i = \frac{1}{M_{i,i}}.$$

1.5.3 Probability generating functions (PGFs)

In the upcoming chapters, we will use Probability Generating Functions for the analysis of our model. For $z \in \mathbb{C}$, the PGF of a random variable X is defined as

$$X(z) = E(z^X) = \sum_{n=0}^{\infty} P(X = n) z^n.$$

Analyticity

The radius of convergence, $\Re$ of a PGF $X(z)$ is defined such that $X(z)$ is analytic for $|z| < \Re$ and not analytic for $z = \Re$. It can be easily shown that $\Re \geq 1$. Therefore, any PGF $X(z)$ is an analytic function of z in the complex unit disk $\{z \in C : |z| < 1\}$. Moreover, $X(z)$ is bounded in the complex disk $\{z \in C : |z| \leq 1\}$ which implies that a PGF does not have poles in this disk.

Normalization condition

Any PGF satisfies the equation

$$X(1) = \sum_{k=0}^{\infty} P(X = k) = 1. \tag{1.1}$$

Computing the probability mass

From the definition of PGF of a random variable, we know that $P(X = n)$ is the coefficient of z^n, therefore we can compute the mass function using derivatives

as

$$P(X = n) = \frac{1}{n!} \frac{\partial^n X(z)}{\partial z^n}\Big|_{z=0}.$$

We often need to compute the tail probabilities of X, i.e. $P(X > n)$ for a large n. Computing derivatives becomes infeasible in here, therefore we compute approximations of the tail probabilities using the Darboux's theorem (see Theorem 3.2 and [45]).

Computing moments from generating function

We know that the radius of convergence, $\Re$, is at least one. If we further assume that it is larger than one, the random variables are light tailed, meaning that all moments are finite and can be computed using derivatives at $z = 1$. That is, for instance

$$\begin{aligned} E(X) &= X'(1) = \frac{\partial X(z)}{\partial z}\Big|_{z=1}, \\ E(X^2) &= X''(1) + X'(1) = \frac{\partial^2 X(z)}{\partial z^2}\Big|_{z=1} + \frac{\partial X(z)}{\partial z}\Big|_{z=1}, \\ Var(X) &= X''(1) + X'(1) - X'(1)^2. \end{aligned} \tag{1.2}$$

Sum of independent random variables

The PGF of a sum of independent random variables equals the product of their corresponding PGFs, i.e., if X_i, for $i = 1, \dots n$, are independent and $Y = \sum_{i=1}^{n} X_i$, then

$$Y(z) = X_1(z)X_2(z) \dots X_n(z).$$

This unique feature is very convenient for the analysis and is one of the motivations to use PGFs throughout this dissertation.

Joint PGFs

Let $u, v \in \mathbb{C}$, and U and V be two random variables, then the joint PGF of U and V is defined as

$$J(u, v) = E(u^U v^V) = \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} P(U = n_1, V = n_2) u^{n_1} v^{n_2}.$$

Clearly, from this definition we can see that the marginal PGFs of U and V can be extracted from $J(u, v)$ as

$$\begin{aligned} U(u) &= J(u, 1), \\ V(v) &= J(1, v). \end{aligned}$$

1.5.4 Markov Decision Processes (MDP)

In the second part of the thesis, we will present a different research topic, streaming of videos on VR headsets. In this section, we will briefly discuss Discrete-time Markov decision processes which form the base of the mathematical analysis of that chapter.

A Markov decision process has four major components: the state space, the probability transition matrix, the action space and the reward function. The state space defines all the possible states of the system we are observing. Time is divided into small slots of fixed length and the system transitions between different states at the slot boundaries. Additionally, at the end of each slot, an action has to be taken from the action space. The state transitions happen in accordance with the transition probability matrix at this time slot. However, the transition may also depend on the action we chose at the end of the slot. That is, say the system is in state s_1 at the time step k, and we chose to take an action a, the system will move to state s_2 with probability $p_k(s_2|s_1, a)$.

The motivation behind choosing an appropriate action at each time step is that the system is given an immediate reward based on the action chosen in a particular state. That is, $r_k(s, a)$ is the reward for taking action a in state s at time k. Since these rewards accumulate with time, the aim is to select actions which maximize the accumulated reward. A wise decision at any slot boundary would be to think of both immediate and future rewards. This is exactly what the algorithm would do which will be presented in further detail in the next paragraph.

Policy and value function

A sequence of actions from the starting state s_0 to the terminal state forms a policy i.e.,

$$\pi = (\pi_0, \pi_1, \ldots, \pi_k), k \leq \infty,$$
$$\pi_l : \mathbb{S} \to \mathbb{A}, l \geq 0.$$

Moreover, a policy is called stationary if the action depends only on the state and is independent of the time, i.e. $\pi_k = \bar{\pi}$. Our aim is to find a policy which maximizes the total reward received in the long run.

Solving MDPs

There are three basic methods to solve MDPs : value iteration, policy iteration and linear programming. While the first two methods are iterative, the third approach uses the simplex method to solve the problem. Due to our large state space, we will focus only on iterative approaches and in particular use policy iteration as this method converges faster.

We assume that our transition probabilities and rewards do not vary with time. That is, $p_k(s_2|s_1, a) = p(s_2|s_1, a)$ and $r_k(s, a) = r(s, a)$. We therefore have the following expression for the expected utility of a state s under policy π, where γ is the discount factor

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s))V^{\pi}(s').$$

The first part of this expression is the immediate reward received for choosing the action $\pi(s)$ while the second part is the sum of all the future rewards expected. Therefore, the utility of a state under a given policy captures the sum of all the rewards expected in the long run. The goal of using MDP is to find the policy which maximizes this utility for each state.

Policy iteration

Policy iteration consists of two sequential phases, policy evaluation and policy improvement. In policy evaluation, we start with a random policy π and evaluate the policy using the Belmann equation. After that step, we check if a better action is available for each state, that is, if the policy is stable. The final stable policy found is the optimal policy. The whole process is divided into four steps

1. Initialization : we start with a random policy π and assign random values to the value function in a defined range.
2. Policy evaluation : Bellman equation is used to evaluate the policy, see Algorithm 1.
3. Policy improvement : after the evaluation of the policy, we check if a better action is available in each state, see Algorithm 2.
4. Stopping criterion : while evaluating a policy, we have to define a threshold θ which defines the completion of policy evaluation.

The final stable policy is the optimal policy π^* which maximises the long term reward for the system.

Since we consider systems with a finite set of states and a finite number of actions in each state, policy iteration will stop in a finite number of steps.

1.6 Dissertation outline

In this section, we will give an overview of the chapters discussed in this dissertation while focusing on the main idea behind each chapter. We will also discuss the publications that resulted from the work described in each chapter individually. The main focus of this dissertation is data backups in cloud. Our main aim is to develop models for cloud data backups. These models capture

Result: A stable policy
$\pi(s) \in \mathbb{A}$ and $V(s) \in \mathbb{U}[0, 100]$ arbitrarily for all $s \in \mathbb{S}$
while $\Delta > \theta$ **do**
  $\Delta \leftarrow 0$
  foreach $s \in \mathbb{S}$ **do**
    $t \leftarrow V(s)$
    $V(s) \leftarrow \sum_{s'} p(s'|s, \pi(s))\Big(r(s, \pi(s), s') + \gamma V(s')\Big)$
    $\Delta \leftarrow max(\Delta, |t - V(s)|)$
  end
end

Algorithm 1: Policy evaluation

Result: Optimal policy
policy-stable $\leftarrow$ False
while *policy-stable* $\neq$ *True* **do**
  policy-stable $\leftarrow$ True
  foreach $s \in \mathbb{S}$ **do**
    $t \leftarrow \pi(s)$
    $\pi(s) \leftarrow \arg\max_a \sum_{s'} p(s'|s, a)\Big(r(s, a, s') + \gamma V(s')\Big)$
    if $t \neq \pi(s)$ **then** policy-stable $\leftarrow$ False ;
  end
end

Algorithm 2: Policy improvement

the influence of the different stochastic components on the performance of the backup processes. Moreover, they also capture the interaction between different components of the backup process. By analyzing the models, we are able to come up with closed-form expressions of the performance measures of the backup processes which helps us provide an insight into their behavior.

In Chapter 2, we analyze a stochastic model where the backup server goes into idle mode when there are no data packets to serve. Further, at the end of this idle period, the backup service is restarted with some probability dependent on the number of data packets in the backlog. The work in this chapter has resulted in a journal publication [1] and conference presentation [46]. In this chapter we start with a batch service model for the backup server which has probabilistic restarting conditions for service. By using this model, we are able to precisely define the key QoS objectives of a backup process, namely, age of data, number of packets in the backlog, probability of a new connection with the namenode and the probability that the backup server is busy. Using the analysis we are able to numerically compute these objectives; finally, we illustrate the use of these objectives to build an optimization problem whose objective is to minimize the cost of running the backup operation. This optimization problem also includes the constraints on some performance measures to ensure a minimum service quality is guaranteed.

The work in Chapter 2 laid foundations for the work done in Chapter 3. In this chapter, we focus on the primary performance measure of backup processes, the age of Data. The performance of data backup processes is guaranteed by defining a Recovery point objective (RPO). We show how ensuring a low age of data helps us ensure the limits of RPO are always met. The work done in this chapter resulted in a journal publication [47] and a conference presentation [48]. Once an exact expression for the probability generating function of the distribution of the age of data is obtained, we do the analysis of its tail distribution. We also use this analysis to evaluate data backup processes with trigger mechanism.

In Chapter 4, we look at a new model of data backup processes. Our main aim here is to model data backup processes which have an inbuilt threshold on the time since last backup. By including 'time since last backup' in the model, we are able to compute its distribution and provide useful insights of using this measure to ensure RPO limits are always met. We also compare the model studied in this chapter and chapter 2. The work done in this chapter resulted in a journal publication [49].

In Chapter 5, we discuss a new research topic, streaming of 360^0 videos on VR headsets. In this research our focus is to model the transmission of video tiles on the headset as a Markov decision process. We also include the user head movement prediction in this model. Our primary aim is to download

the best possible tiles which will ensure a good QoE for the user when the network conditions are not completely reliable. This work has been accepted as a conference publication at Valuetools 2020 [50] and a journal publication [51] which is currently under review.

In Chapter 6, we summarize our contributions and draw conclusions to the work done in this dissertation. We also provide some possible future directions of this work.

We now provide a list of publications which resulted from the work discussed in this dissertation:

- [1] A. Saxena, D. Claeys, H. Bruneel, B. Zhang, J. Walraevens, Modeling data backups as a batch-service queue with vacations and exhaustive policy, Computer Communications 128 (2018) 46 – 59
- [47] A. Saxena, D. Claeys, H. Bruneel, J. Walraevens, Analysis of the age of data in data backup systems, Computer Networks 160 (2019) 41–50.
- [49] A. Saxena, D. Claeys, B. Zhang, J. Walraevens, Cloud data storage: A queueing model with thresholds, Annals of Operations Research (2019) 1572-9338.
- [50] A. Saxena, S. Subramanyam, P. Cesar, R. van der Mei, H. van den Berg, Efficient, QoE aware delivery of 360^0 videos on VR headsets over mobile links, in: Proceedings of the 13th EAI International Conference on Performance Evaluation Methodologies and Tools, Valuetools '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 156–163.
- [51] A. Saxena, S. Subramanyam, P. Cesar, R. van der Mei, H. van den Berg, QoE aware delivery of 360^0 videos on wireless VR headsets, IEEE Transactions on Network and Service Management, *Under Review.*

2

Cloud data backups

With the rapid increase in data storage requirements and migration towards cloud storage, the right balance between doing backups very frequently and reducing resource consumption has to be found. To help finding this balance, in this chapter, we model a wide set of data backup processes as a generic batch service queueing model with vacations, exhaustive service and probabilistic restarting conditions. Batch service queueing models with vacations provide a very natural way of modeling data backups to cloud infrastructure. We are able to precisely compute the QoS measures of the data backup processes using this model. Nevertheless, to the best of our knowledge, data backup processes have been rarely modeled and analyzed as queueing processes. [52] propose a queueing model of cloud based streaming services to evaluate service quality. [53] model data backups as a two dimensional Markov chain and study optimization of the network bandwidth utilization. [54] present decision algorithms, utilizing Markov decision processes, which use constraints on recovery parameters to decide when to backup and which type of backup to perform.

Queueing models with vacations, in general, have been studied extensively in the literature (see [55] and [56] for details about some applications). In particular, some relevant batch-service queueing models with vacations have been studied. [57] study a server which goes into a vacation when the system is empty and restarts service on return if there is at least one customer in the system. [58] study a batch-service model with multiple vacations where the server requires a minimum threshold of queue size to keep the server on and another threshold to restart it after the vacation. The model also takes into account a setup and close down time for the server. Other models studied by [59] and [60] consider a single vacation case with a threshold where the system waits in idle condition after returning from a vacation if the threshold is not matched. The queueing models listed above have not been built and studied from the perspective of data backup processes. Our model, on the other hand, is specifically designed to model a wide range of exhaustive data backup processes.

We model exhaustive data backup processes as a very general batch-service queueing model with vacations and probabilistic restarting conditions. Discrete-time models are appropriate to model several telecommunication processes (see [61]). We thoroughly analyze the discrete-time queueing model and obtain various performance measures. We then explicitly translate performance measures of the queueing model to QoS measures of data backup processes. Our model and the QoS measures are useful to find the right balance between doing cloud backups frequently enough (to increase data safety) and reducing resource usage (power consumption and communication costs). We study the dependence of QoS measures on the model parameters as well as compute the values of these parameters that minimize the user cost function. Note that, as stated in [62], pure threshold policies may not be optimal. Therefore, we define parameters for probabilistic restarts to model the behavior of the server. Moreover, since the cost of connecting to cloud infrastructure is typically independent of the amount of data backed up, our work focuses on exhaustive policies (see example the cost structure of Amazon Web Services [8] and Microsoft Azure Storage [10]).

This chapter presents an extension of previous work [63]. The model studied by [63] assumes single slot vacations and single slot service times. Our model extends the vacation length and service lengths to general distributions as well as introduces probabilistic restarting conditions. Making the model much more general implies that it models a wider set of exhaustive data backup policies. Additionally, this work looks at significantly more performance measures and a more thorough modeling of data backups [1].

We characterize our model using four types of parameters:

- l : threshold on the queue content to begin service with probability 1
- c : capacity of backup server
- α_i where $i = 1, \ldots n-1$: starting probability when i packets are present
- T : a random variable which defines the length of any vacation

In addition to a wide range of data backup policies, this model covers a wide range of user behavior and backup server features. That is, both the arrival distribution (the number of arrivals in a slot) and service times (which depend on the batch size) can be chosen freely provided that the system remains stable. These flexible parameters empower us to set their values based on the traffic seen by the server and required system performance. Our aim is to answer questions such as: given the service level agreements, what are the right model parameters to maintain a stable and efficient system? Such questions can be answered accurately once we have computed the exact expressions of the main QoS measures in terms of the parameters. The main advantage of our analysis is that we can compute the QoS measures for a general set of parameters.

We start with a description of the model and justify its suitability for data backups in Section 2.1. We also discuss in detail the utility of each parameter of the model. It is then followed by the analysis of the model in Section 2.2. We also highlight some observations at the service/vacation epochs which give us some important relations to solve our model. Using the results of Section 2.2, we deduce the performance measures of the model in Section 2.3 and QoS measures of data backups in Section 2.4. We compute measures such as the age of data at the beginning of a backup period and the probability that the backup server is busy in a slot. We also evaluate the performance of the system for different system parameters in Section 2.5. Further, we construct an optimization problem to highlight how one can compute the optimal parameters to minimize user cost. In the last section we summarize our observations and results from previous sections and present possible directions for future work.

2.1 Backup process model

In data backup processes, packets that have not yet been backed up are part of the backlog. The backlog size then corresponds to the number of packets that have not yet been backed up. In case of exhaustive backup service processes, when the backup server initiates a backup, it continues backing up until the backlog size becomes zero.

We model the backlog as a queue where the customers represent data packets and the backup server by a batch server. As a result, packets arriving in the queue correspond to newly generated packets that have not yet been backed up.

The batch server has a capacity of maximum c packets and it employs an exhaustive policy i.e. the server keeps serving until the system content becomes zero. At that moment, the backup server immediately goes into energy saving mode which is modeled as a vacation of T slots with T a random variable. If upon returning from the vacation, the system content (i.e. the total number of packets in the queue and the server) is larger than or equal to l, the server immediately initiates the service (a new backup); otherwise it starts the service with a probability α_i, where i is the number of packets in queue. With probability $1 - \alpha_i$ the server goes into another vacation i.e. stays in the energy saving mode. Lengths of vacation periods are independent and identically distributed.

2.1.1 Data storage on cloud infrastructures

We discussed the storage process on cloud infrastructure in detail in the section 1.2. Briefly put, when the backup server wants to write data, it sends a request to the namenode. Namenode stores the metadata of the data to be

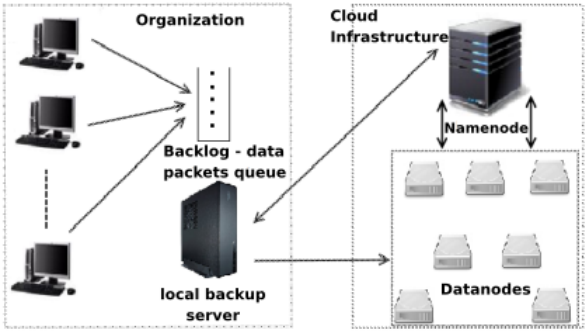


Figure 2.1: Components of the backup process

stored and sends back the IP-addresses of the data nodes on which the backup server is allowed to write the data. A data packet completes its service when it has been written to the cloud infrastructure. Our work assumes that the backups performed are incremental in nature, i.e., only the changes in the system from the last copy are saved and backed up. Figure 2.2 highlights the components of our queueing model and connects it with the cloud data backup process of Figure 2.1.

Now we discuss the model choices and the importance of the model parameters and characteristics.

Batch Service

Data packets are segmented and transmitted in batches of size c from the local backup server to the cloud to improve the efficiency of the transmission. (see [64], [65] for details)

Exhaustive service

A new connection is established when the server communicates with the namenode of the cloud service provider, that is, when a new backup is initiated. Cloud service providers charge cloud users for each new connection. Since the cost of a new connection is independent of the amount of data to be backed up, an exhaustive service policy comes naturally. The details of cost structure of cloud service providers are available at [8] and [10].

Starting probabilities

In contrast to a fixed threshold policy, probabilistic restarting conditions allow the system to restart service even when the threshold l has not been met.

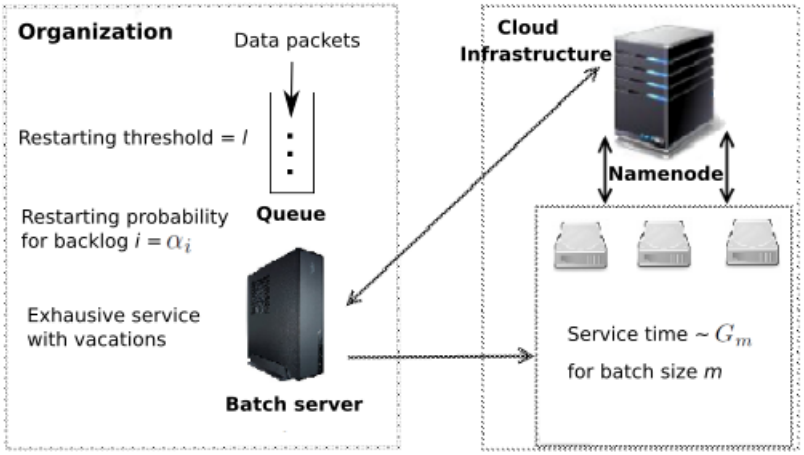


Figure 2.2: Queueing model of cloud data backup process

It is important to note that fixed threshold based policies are just a special case of our model parameters, i.e., $\alpha_i = 0$, $\forall i < l$ where l is the threshold. Therefore, including the restarting probabilities makes the model more generic and encourages a much wider set of exhaustive backup policies.

Further, randomization has often been used to achieve an improvement in the efficiency of the system. For instance [66] use randomization to ease-off the traffic for data backups and therefore reduce the peak load. [67] use randomization to improve the efficiency of distribution of data blocks over a network. Similarly, by including probabilistic restarts in our model, we are able to achieve a trade-off between contrasting QoS measures. This is also illustrated in section 2.4.4.

Vacations

Data processes are very power intensive and a significant amount is spent to keep the system idle while it waits for packets to arrive. As stated earlier from [17], 1.5% of total power produced in the US was used for Data center computing. By introducing vacations, the local backup server sleeps during the idle periods and power consumption of the backup server is reduced. Moreover, by modeling vacation length as a random variable, we allow our model to incorporate more randomization which makes it, again, more generic. Clearly, deterministic fixed length vacations is just a special case of this model parameter.

A_k	number of arrivals in slot k. It has a generating function $A(z)$. Hence, we assume independent and identically distributed (i.i.d.) arrivals in each slot. However, this can be any general distribution.
T	vacation length. This variable has a generating function $T(z)$.
c	capacity of the backup server; defines the maximum number of packets that can be served in a single batch.
l	threshold for service start. When the system wakes up and finds $\geq l$ packets the server resumes service.
α_i	probability that a backup process is started in presence of exactly i packets with $\alpha_0 = 0, 0 \leq \alpha_i \leq 1, \forall 0 < i < l$ and $\alpha_i = 1, \forall i \geq l$.
G_m	service time required for a batch of m packets. This stochastic variable has a (general) generating function $G_m(z)$.
ρ	load of the system defined by $A'(1)G'_c(1)/c$.
k	this subscript defines the slot number.
Q_k	queue content (the number of packets in the queue, excluding the ones receiving service) during the k^{th} slot.
S_k	server content (the number of packets in service) during the k^{th} slot. Since $S_k = 0$ when the backup service is in vacation, we use this event as a proxy to check if the backup is running.
R_k	remaining time in slot k of the current service/vacation period. When the server is serving packets it is the remaining serving time of those packets; otherwise, it is the remaining vacation time of the current vacation.

Table 2.1: List of all the notations used in the analysis

2.2 Analysis of the model

We study the discrete time model of the system defined in section 2.1 using the definitions from Table 2.1. We calculate the joint generating function of the queue content, server content and remaining service time at a random slot boundary. From this generating function we deduce some of the performance measures of the model such as the system content at a random slot boundary and the number of customers served in a random batch.

This analysis is critical to be able to compute the QoS measures of the data backup process introduced in section 1.2.1. It also helps us to select the starting parameters. We illustrate the selection of parameters using an optimization problem in section 2.5.3.

Note that, all the terms defined in the Table 2.1 are discrete variables. Without loss of generality, in the analysis we have assumed that $l \geq c$. This is because a model with $l = l_0, l_0 < c$ can be rewritten as another equivalent model with $l = c$ and $\alpha_i = 1, \forall l_0 \leq i < c$ while keeping all the remaining parameters the same. Further, we assume that the probability of zero arrivals in a slot $A(0) > 0$. Otherwise, the backlog would never be empty and the system would never enter a vacation.

2.2.1 Transition equations

We relate the random vectors (Q_k, S_k, R_k) and $(Q_{k+1}, S_{k+1}, R_{k+1})$. The relations are governed by following observations

- If the remaining service/vacation time of the current batch is more than one slot ($R_k > 1$), then there is no change in the system in slot $k+1$ other than new arrivals ($Q_{k+1} = Q_k + A_k$) and the R_k reduces by one.
- If the server is serving during slot k, the remaining service time is equal to 1 slot ($R_k = 1$) and there are a positive number of packets in queue at the end of the slot ($Q_k + A_k > 0$), then the server starts service of a new batch in slot $k+1$ due to the exhaustive service policy.
- If the remaining service/vacation time is equal to 1 slot ($R_k = 1$) and there are no packets in the queue at the end of the slot ($Q_k + A_k = 0$), the server goes for a vacation starting in slot $k+1$.
- If the server is in vacation and is about to wake up in the next slot ($R_k = 1$), the server begins service with probability α_i in slot $k+1$, where i is the number of packets in queue at the end of the slot k ($i = Q_k + A_k$).

This leads to the following relations

$$\begin{pmatrix} Q_{k+1} \\ S_{k+1} \\ R_{k+1} \end{pmatrix} = \begin{cases} (m, S_k, R_k - 1) & \text{if} \quad R_k > 1, Q_k + A_k = m \\ (0, m, G_m) & \text{if} \quad R_k = 1, S_k > 0, Q_k + A_k = m, 0 < m < c \\ (m-c, c, G_c) & \text{if} \quad R_k = 1, S_k > 0, Q_k + A_k = m, c \leq m < l \\ (0, 0, T) & \text{if} \quad R_k = 1, Q_k + A_k = m, m = 0 \\ (m-c, c, G_c) & \text{if} \quad R_k = 1, Q_k + A_k = m, m \geq l \\ & \text{if} \quad R_k = 1, S_k = 0, Q_k + A_k = m, 0 < m < c \\ (m, 0, T) & \text{with probability} \quad (1 - \alpha_m) \\ (0, m, G_m) & \text{with probability} \quad \alpha_m \\ & \text{if} \quad R_k = 1, S_k = 0, Q_k + A_k = m, c \leq m < l \\ (m, 0, T) & \text{with probability} \quad (1 - \alpha_m) \\ (m-c, c, G_c) & \text{with probability} \quad \alpha_m \end{cases}$$

Note that $R_k \geq 1$, because as soon as it becomes one, it either changes to the service time of next batch or a multiple slot length vacation in the next slot. For this reason we will work with $R_k - 1$ rather than R_k in the generating function.

2.2.2 Limiting generating function

Define,

$$V_{k+1}(z,y,x) = E(z^{Q_{k+1}} y^{S_{k+1}} x^{R_{k+1}-1})$$

and

$$E\Big(z^X\{\text{condition}\}\Big) = E\big[z^X|\text{condition}\big]P(\text{condition}).$$

Using the transition equations defined in section 2.2.1, one can write

$$\begin{aligned}
&V_{k+1}(z,y,x) = \\
&E\Big(z^{Q_k+A_k} y^{S_k} x^{R_k-1-1}\{R_k>1\}\Big) + E\Big(x^{T-1}\{R_k=1, Q_k+A_k=0\}\Big) \\
&+ \sum_{m=1}^{c-1} E\Big(y^m x^{G_m-1}\{R_k=1, S_k>0, Q_k+A_k=m\}\Big) \\
&+ \sum_{m=c}^{l-1} E\Big(z^{m-c} y^c x^{G_c-1}\{R_k=1, S_k>0, Q_k+A_k=m\}\Big) \\
&+ \sum_{m=l}^{\infty} E\Big(z^{m-c} y^c x^{G_c-1}\{R_k=1, Q_k+A_k=m\}\Big) + \sum_{m=1}^{c-1} E\Big(\big(z^m x^{T-1}(1-\alpha_m) \\
&+ y^m x^{G_m-1}\alpha_m\big)\{R_k=1, S_k=0, Q_k+A_k=m\}\Big) + \sum_{m=c}^{l-1} E\Big(\big(z^m x^{T-1}(1-\alpha_m) \\
&\quad + z^{m-c} y^c x^{G_c-1}\alpha_m\big)\{R_k=1, S_k=0, Q_k+A_k=m\}\Big)
\end{aligned}$$

which can be transformed in

$$\begin{aligned}
V_{k+1}(z,y,x) = {} & \frac{A(z)}{x} E\Big(z^{Q_k} y^{S_k} x^{R_k-1}\Big) - \frac{A(z)}{x} E\Big(z^{Q_k} y^{S_k} x^{R_k-1}\{R_k=1\}\Big) \\
& + E\big(x^{T-1}\{R_k=1, Q_k+A_k=0\}\big) \\
& + \sum_{m=1}^{c-1} E\Big(y^m x^{G_m-1}\{R_k=1, S_k>0, Q_k+A_k=m\}\Big) \\
& + \sum_{m=c}^{l-1} E\Big(z^{m-c} y^c x^{G_c-1}\{R_k=1, S_k>0, Q_k+A_k=m\}\Big) \\
& + \sum_{m=l}^{\infty} E\Big(z^{m-c} y^c x^{G_c-1}\{R_k=1, Q_k+A_k=m\}\Big) \\
& + \sum_{m=1}^{c-1} E\Big(\big(z^m x^{T-1}(1-\alpha_m) + y^m x^{G_m-1}\alpha_m\big)\{R_k=1, \\
& \quad S_k=0, Q_k+A_k=m\}\Big) + \sum_{m=c}^{l-1} E\Big(\big(z^m x^{T-1}(1-\alpha_m)
\end{aligned}$$

$$+ z^{m-c}y^c x^{G_c-1}\alpha_m)\{R_k = 1, S_k = 0, Q_k + A_k = m\}\Big). \tag{2.1}$$

In heavy traffic conditions, the system never takes vacations and the batch size is always the maximum c. Therefore, for the system to remain stable, the average amount of work arriving during a service period should be less than the capacity of the batch server. The stability condition of this model is therefore given by $A'(1)G'_c(1) < c$, i.e. $\rho < 1$. The vacations do not play a role in the stability condition which is a standard results in queueing systems with vacation. [68] prove the stability condition of such queueing systems by proving that the underlying Markov chain of the system content is ergodic under the condition $\rho < 1$. Under the assumption that $\rho < 1$, we look at the state of the system variables in steady state.
Define

$$d_1(m) = \lim_{k\to\infty} P(Q_k + A_k = m, R_k = 1, S_k > 0), \tag{2.2}$$

$$d_0(m) = \lim_{k\to\infty} P(Q_k + A_k = m, R_k = 1, S_k = 0), \tag{2.3}$$

$$d(m) = \lim_{k\to\infty} P(Q_k + A_k = m, R_k = 1) = d_0(m) + d_1(m), \tag{2.4}$$

$$V(z, y, x) = \lim_{k\to\infty} V_k(z, y, x).$$

Our first aim is to write all the expressions in terms of the unknown boundary probabilities $\mathbb{U} = \{d(0), d(1), d(2), \dots d(c-1), d_0(1), d_0(2), \dots d_0(l-1)\}$ and then formulate a linear system to solve for these unknowns. From (2.1), by letting $k \to \infty$ and replacing the terms by their generating functions, we can write

$$\begin{aligned} V(z, y,x) = {} & \frac{A(z)}{x}V(z, y, x) - \frac{A(z)}{x}V(z, y, 0) + \frac{T(x)}{x}d(0) + \sum_{m=1}^{c-1} y^m \frac{G_m(x)}{x}d(m) \\ & + \Big(\frac{y}{z}\Big)^c \frac{G_c(x)}{x} \lim_{k\to\infty} \sum_{m=0}^{\infty} z^m P(Q_k + A_k = m, R_k = 1) \\ & - \Big(\frac{y}{z}\Big)^c \frac{G_c(x)}{x} \sum_{m=0}^{c-1} z^m d(m) + \sum_{m=1}^{c-1} \frac{\big(z^m T(x) - y^m G_m(x)\big)}{x}(1 - \alpha_m)d_0(m) \\ & + \sum_{m=c}^{l-1} \Big(z^m \frac{T(x)}{x} - z^{m-c}y^c \frac{G_c(x)}{x}\Big)(1 - \alpha_m)d_0(m). \end{aligned} \tag{2.5}$$

Since $\sum_{m=0}^{\infty} z^m P(Q_k + A_k = m, R_k = 1)$ is easily computed, we have

$$\begin{aligned} \lim_{k\to\infty} \sum_{m=0}^{\infty} z^m P(Q_k + A_k = m, R_k = 1) &= \lim_{k\to\infty} A(z)V_{k+1}(z, 1, 0) \\ &= A(z)V(z, 1, 0). \end{aligned} \tag{2.6}$$

Using the relation (2.6) we can rewrite (2.5) as

$$\begin{aligned} V(z,y,x)\big(x-A(z)\big) &= -A(z)V(z,y,0) \\ &\quad + \sum_{m=1}^{c-1}\big(y^m G_m(x) - z^{m-c}y^c G_c(x)\big)d(m) \\ &\quad + \Big(\frac{y}{z}\Big)^c A(z)V(z,1,0)G_c(x) + \Big(T(x) - \Big(\frac{y}{z}\Big)^c G_c(x)\Big)d(0) \\ &\quad + \sum_{m=1}^{c-1}\Big(z^m T(x) - y^m G_m(x)\Big)(1-\alpha_m)d_0(m) \\ &\quad + \sum_{m=c}^{l-1}\Big(z^m T(x) - z^{m-c}y^c G_c(x)\Big)(1-\alpha_m)d_0(m). \quad (2.7) \end{aligned}$$

From this equation, we can find expressions for $V(z,y,0)$ and $V(z,1,0)$ by different substitutions of x, y and z. Substituting $x = A(z)$ followed by $y \to 1$ gives us

$$\begin{aligned} A(z)V(z,1,0)\big(z^c - G_c(A(z))\big) &= \Big(z^c T(A(z)) - G_c(A(z)\Big)d(0) \\ &\quad + \sum_{m=1}^{c-1}\Big(z^c G_m(A(z)) - z^m G_c(A(z))\Big)d(m) \\ &\quad + \sum_{m=1}^{c-1}\Big(z^m T(A(z)) - G_m(A(z))\Big)z^c(1-\alpha_m)d_0(m) \\ &\quad + \sum_{m=c}^{l-1}\Big(z^c T(A(z)) - G_c(A(z))\Big)z^m(1-\alpha_m)d_0(m). \quad (2.8) \end{aligned}$$

Similarly, using the substitution $x = A(z)$ in (2.7) yields

$$\begin{aligned} A(z)V(z,y,0) &= A(z)V(z,1,0)G_c(A(z))\Big(\frac{y}{z}\Big)^c - d(0)\Big(\Big(\frac{y}{z}\Big)^c G_c(A(z)) \\ &\quad - T(A(z))\Big) + \sum_{m=1}^{c-1}\Big(y^m G_m(A(z)) - z^{m-c}y^c G_c(A(z))\Big)d(m) \\ &\quad + \sum_{m=1}^{c-1}\Big(z^m T(A(z)) - y^m G_m(A(z))\Big)(1-\alpha_m)d_0(m) \\ &\quad + \sum_{m=c}^{l-1}\Big(z^m T(A(z)) - z^{m-c}y^c G_c(A(z))\Big)(1-\alpha_m)d_0(m). \quad (2.9) \end{aligned}$$

Clearly, one can use (2.8) in the latter expression and get an expression for $V(z,y,0)$ in terms of the unknown boundary probabilities $\mathbb{U}$. This expression would be useful to compute an expression of system content in section 2.2.3. In the remaining part of this section, our primary aim is to form a solvable linear system of equations in $\mathbb{U}$. As a step towards this objective, we look at the generating function of the system content.

2.2.3 Generating function of system content

The system content in any time slot is defined as the number of packets either in queue waiting for service or currently in service. Therefore, its generating function is given by $V(z,z,1)$. In (2.7), replacing $x=1$, and $y=z$ gives us,

$$\Big(1-A(z)\Big)V(z,z,1) = -A(z)V(z,z,0) + A(z)V(z,1,0). \tag{2.10}$$

To further solve (2.10), replace $x=A(z)$ and $y=z$ in (2.7) which gives us

$$\begin{aligned} A(z)V(z,z,0) = G_c(A(z))A(z)V(z,1,0) &+ \Big(T(A(z)) - G_c(A(z))\Big)d(0) \\ &+ \sum_{m=1}^{c-1} z^m \Big(G_m(A(z)) - G_c(A(z))\Big)d(m) \\ &+ \sum_{m=1}^{c-1} z^m \Big(T(A(z)) - G_m(A(z))\Big)(1-\alpha_m)d_0(m) \\ &+ \sum_{m=c}^{l-1} z^m \Big(T(A(z)) - G_c(A(z))\Big)(1-\alpha_m)d_0(m). \end{aligned} \tag{2.11}$$

Using eqns (2.8)-(2.11) we get an expression for $V(z,z,1)$ in terms of the unknowns $\mathbb{U}$, which after some rewriting yields

$$\begin{aligned} \Big(1-A(z)\Big)\Big(z^c - G_c(A(z))\Big)V(z,z,1) = B_0(z)d(0) &+ \sum_{m=1}^{c-1} B_m(z)d(m) \\ &+ \sum_{m=1}^{c-1} W_m(z)d_0(m) + \sum_{m=c}^{l-1} R_m(z)d_0(m), \end{aligned} \tag{2.12}$$

where

$$\begin{aligned} B_0(z) &= G_c(A(z))(1-z^c)\Big(T(A(z))-1\Big), \\ B_m(z) &= G_m(A(z))G_c(A(z))\Big(z^m - z^c\Big) + z^m\Big(z^c-1\Big)G_c(A(z)) \\ &\quad + z^c\Big(1-z^m\Big)G_m(A(z)), \\ W_m(z) &= \Big(G_m(A(z))G_c(A(z))\big(z^c - z^m\big) + z^c\big(z^m-1\big)G_m(A(z)) \\ &\quad + G_c(A(z))T(A(z))z^m\big(1-z^c\big)\Big)(1-\alpha_m), \\ R_m(z) &= z^m G_c(A(z))(1-z^c)\big(T(A(z))-1\big)(1-\alpha_m). \end{aligned}$$

For a given value of z, the values $A(z)$, $B_m(z)$, $W_m(z)$ and $R_m(z)$ are known quantities. We now use a result from [69] to prove that $z^c - G_c(A(z))$ has c zeros inside the complex unit circle $|z| \leq 1$. They have proved that if $F(z)$ is a

probability generating function with $F(0) > 0$ and $F'(1) < c$ with $c \in \mathbb{N}$, then the function $z^c - F(z)$ has exactly c zeros inside the unit circle $|z| \leq 1$.

Clearly, $G_c(A(z))$ is the probability generating function of the number of arrivals in a service period of c packets with $G_c(A(0)) > 0$ since $A(0) > 0$. Moreover, from the stability condition $F'(1) = G'_c(1)A'(1) < c$. Therefore, from Theorem 3.2 in [69], $z^c - G_c(A(z))$ has c zeros in the complex unit disk $|z| \leq 1$. This includes the trivial solution $z = 1$ which will be excluded because at $B_0(1) = 0$ and $B_m(1) = W_m(1) = R_m(1) = 0, \forall m$.

Therefore, eqn (2.12) has $c - 1$ zeros in the complex unit disk with $z = 1$ excluded i.e. $\{z : |z| \leq 1, z \neq 1\}$. This gives us $c - 1$ linear equations in unknowns $\mathbb{U}$, however the cardinality of $\mathbb{U}$ is $c+l-1$. Therefore, we need l more relations to form a solvable linear system of equations. For more relations, we look at *service initiation opportunities*. These opportunities are defined as the epochs of start/end of service/vacation times. These observations enable us to find more linear relations between the unknowns $\mathbb{U}$ which is a step further towards forming a solvable linear system in these unknowns.

2.2.4 Observation at service initiation opportunities

We observe the system content at the epoch points of service initiation opportunities. We start by defining a few terms which are also illustrated in Figure 2.3:

- j : this subscript defines the epoch number being looked at.
- U_j = the system content at the *end* of j^{th} epoch, i.e., the number of packets waiting in the queue or in service at the end of the epoch.
- $\tau_j = 1$, if the j^{th} period is a service period, 0 if vacation period.

At the end of a vacation (v1), if the system has an empty backlog it will enter a new vacation (v2). This can happen iff the following conditions are true

- at the beginning of the vacation (v1), the system had an empty backlog, and
- there were no arrivals during this vacation (v1).

This gives us (2.13).

$$P(U_j = 0, \tau_j = 0) = P(U_{j-1} = 0)T(A(0)) \tag{2.13}$$

Similarly, the system wakes up with n packets ($n > 0$) in the backlog iff

- the system entered the vacation with empty backlog and there were n arrivals during the vacation, or

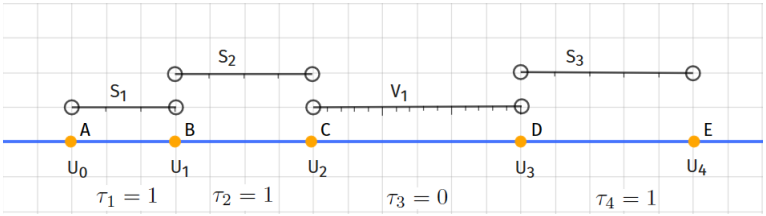


Figure 2.3: Illustration of service initiation opportunities, service periods and vacation periods

- the system entered the vacation with i packets present ($i > 0$) and there were $n - i$ arrivals during the vacation.

Note that, for the system to enter a vacation with i packets where $0 < i \leq n$, the previous epoch also has to be a vacation epoch because of the exhaustive service policy. So for $0 < n < l$, where $t_A(n)$ denotes the probability of n arrivals in a vacation period, we have

$$P(U_j = n, \tau_j = 0) = \sum_{i=1}^{n} P(U_{j-1} = i, \tau_{j-1} = 0)(1 - \alpha_i)t_A(n - i) + P(U_{j-1} = 0)t_A(n). \tag{2.14}$$

Moreover, $t_A(n)$ is a known constant because it is the coefficient of z^n in $T(A(z))$. Under the assumption of $\rho < 1$, the system would reach a steady state and under this steady state, the limiting probabilities of these terms are defined as

$$\pi_i(m) = \lim_{j\to\infty} P(U_j = m, \tau_j = i),$$
$$\pi(m) = \lim_{j\to\infty} P(U_j = m).$$

Hence, we can write

$$\pi_0(0) = \pi(0)T(A(0)), \tag{2.15}$$
$$\pi_0(n) = \sum_{i=1}^{n} \pi_0(i)(1 - \alpha_i)t_A(n - i) + \pi(0)t_A(n), \qquad 0 < n < l, \tag{2.16}$$
$$\pi(n) = \pi_0(n) + \pi_1(n). \tag{2.17}$$

2.2.5 Relating π, $d_0(m)$ and $d(0)$

Now we relate the terms defined in (2.15)-(2.17) and (2.2)-(2.4). In earlier sections we defined,

$$d_0(m) = \lim_{k\to\infty} P(Q_k + A_k = m, R_k = 1, S_k = 0)$$

$$= \lim_{k\to\infty} P(Q_k + A_k = m, S_k = 0 | R_k = 1) P(R_k = 1).$$

Observe that when $R_k = 1$, the starting boundary of slot $k+1$ is the beginning of a service initiation opportunity epoch which has been analyzed in section 2.2.4. Therefore, $\lim_{k\to\infty} P(Q_k + A_k = m, S_k = 0 | R_k = 1) = \lim_{j\to\infty} P(U_j = m, \tau_j = 0)$ and $\lim_{k\to\infty} P(Q_k + A_k = m | R_k = 1) = \lim_{j\to\infty} P(U_j = m)$. These can be used to come up with the following relations,

$$d_0(m) = \pi_0(m) \lim_{k\to\infty} P(R_k = 1),$$

$$d(m) = \pi(m) \lim_{k\to\infty} P(R_k = 1).$$

Therefore, we can translate (2.15)-(2.17) to equations in $d_0(m)$ and $d(0)$ and we obtain

$$d_0(0) = d(0) T(A(0)), \tag{2.18}$$

$$d_0(n) = \sum_{i=1}^{n} d_0(i)(1-\alpha_i) t_A(n-i) + d(0) t_A(n), \qquad 0 < n < l. \tag{2.19}$$

(2.19) can be rewritten as,

$$B \cdot \begin{bmatrix} d(0) \\ d_0(1) \\ d_0(2) \\ \vdots \\ d_0(l-1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{2.20}$$

where B is defined as

$$B = \begin{bmatrix} -t_A(1) & 1-(1-\alpha_1)t_A(0) & 0 & \dots & 0 \\ -t_A(2) & -(1-\alpha_1)t_A(1) & 1-(1-\alpha_2)t_A(0) & \dots & 0 \\ -t_A(3) & -(1-\alpha_1)t_A(2) & -(1-\alpha_2)t_A(1) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -t_A(l-1) & -(1-\alpha_1)t_A(l-2) & -(1-\alpha_2)t_A(l-3) & \dots & 1-(1-\alpha_{l-1})t_A(0) \end{bmatrix}.$$

We do not include (2.18) in the matrix equations because $d_0(0) \notin \mathbb{U}$. However, this equation will be useful while looking at the performance measures in section 2.3. Summarizing, (2.20) gives us $(l-1)$ linearly independent equations between the unknowns in $\mathbb{U}$ which are crucial to create a solvable system.

Combining the results of sections 2.2.3 and 2.2.5, we have in total $c + l - 2$ equations which is still less than the number of unknowns $(c + l - 1)$. However, by the normalization property of generating functions, we know that $V(z, z, 1)|_{z=1} = 1$. This normalization condition, discussed in paragraph 2.2.6, gives us an additional relation in the unknowns. Combining all these equations together, we get a solvable system of $l + c - 1$ linearly independent equations and $l + c - 1$ unknowns $\mathbb{U}$.

2.2.6 Normalization condition

As stated above, the normality condition $V(1,1,1)=1$ plays a critical role to get a solvable linear system. This equation is obtained by applying L'Hospital's rule twice to (2.12) which gives us

$$\left(\frac{cT'(1)}{c-G_c'(1)A'(1)}\right)d(0)+\sum_{m=1}^{l-1}\left(\frac{cT'(1)}{c-G_c'(1)A'(1)}\right)(1-\alpha_m)d_0(m) + \sum_{m=1}^{c-1}\left(\frac{cG_m'(1)-mG_c'(1)}{c-G_c'(1)A'(1)}\right)\Big(d(m)-(1-\alpha_m)d_0(m)\Big)=1. \tag{2.21}$$

To easily write the system of equations in matrix form, eqn (2.21) can be rewritten as

$$\left(\frac{cT'(1)}{c-G_c'(1)A'(1)}\right)d(0)+\sum_{m=1}^{c-1}\left(\frac{mG_c'(1)+cT'(1)-cG_m'(1)}{c-G_c'(1)A'(1)}\right)(1-\alpha_m)d_0(m)+ \sum_{m=c}^{l-1}\left(\frac{cT'(1)}{c-G_c'(1)A'(1)}\right)(1-\alpha_m)d_0(m)+\sum_{m=1}^{c-1}\left(\frac{cG_m'(1)-mG_c'(1)}{c-G_c'(1)A'(1)}\right)d(m)=1. \tag{2.22}$$

2.2.7 Original generating function

In this section we show that just like the generating function of system content, it is possible to express the original generating function defined in (2.2) in terms of the unknowns $\mathbb{U}$. We have already written $V(z,y,x)$ in terms of $V(z,y,0)$, $V(z,1,0)$ and the unknowns $d_0(n)$s and $d(n)$s in (2.7). We have also shown that we can write $V(z,y,0)$ and $V(z,1,0)$ in terms of the unknowns in (2.8) and (2.9). Therefore, substituting (2.8) and (2.9) into (2.7) we get

$$(x-A(z))V(z,y,x)=B_0(z,y,x)d(0)+\sum_{m=1}^{c-1}B_m(z,y,x)d(m) + \sum_{m=1}^{c-1}W_m(z,y,x)(1-\alpha_m)d_0(m)+\sum_{m=c}^{l-1}R_m(z,y,x)(1-\alpha_m)d_0(m).$$

where

$$B_0(z,y,x)=T(x)-T(A(z))+\frac{y^c\big(G_c(x)-G_c(A(z))\big)\big(T(A(z))-1\big)}{z^c-G_c(A(z))},$$

$$B_m(z,y,x)=y^m(G_m(x)-G_m(A(z))) + \frac{y^c\big(G_c(x)-G_c(A(z))\big)\big(G_m(A(z))-z^m\big)}{z^c-G_c(A(z))},$$

$$W_m(z,y,x)=z^m(T(x)-T(A(z)))-y^m(G_m(x)-G_m(A(z)))$$

$$+\Big(\frac{z^m T(A(z)) - G_m(A(z))}{z^c - G_c(A(z))}\Big) y^c \big(G_c(x) - G_c(A(z)),$$

$$R_m(z,y,x) = z^m \left(T(x) - T(A(z)) + \frac{y^c \big(G_c(x) - G_c(A(z))\big)\big(T(A(z)) - 1\big)}{z^c - G_c(A(z))} \right).$$

2.3 General performance measures of the model

In this section we write expressions of generating functions of some common stochastic variables of this queueing model in terms of $V(z,y,x)$ and input functions. Since we have shown that $V(z,y,x)$ can be expressed in terms of the boundary probabilities in the set $\mathbb{U}$, the generating functions below can also be easily written as functions of these boundary probabilities.

2.3.1 System content at random slot boundaries

As the system content includes both the customers in queue and those in service, the pgf of the system content is given by the generating function $V(z,z,1)$. Exact expression of this generating function was computed in (2.12).

2.3.2 System content at service initiation opportunities

We have already looked at the generating function of $d(m)$. From (2.6), the generating function of the system content at service initiation opportunities is

$$\frac{A(z)V(z,1,0)}{V(1,1,0)}.$$

Moreover, one can write the generating function of the system content at vacation completion times and service completion times respectively as

$$\frac{A(z)V(z,0,0)}{V(1,0,0)}, \qquad \frac{A(z)\big(V(z,1,0) - V(z,0,0)\big)}{V(1,1,0) - V(1,0,0)}. \tag{2.23}$$

2.3.3 Number of customers served in a random batch

This is an important measure which reflects how efficiently the server resources are being utilized. The sample space for the number of customers in server at a slot boundary can be divided into two mutually exclusive and exhaustive events, $S_k = 0$ and $S_k > 0$. $S_k = 0$ implies that the system is in vacation in the k^{th} slot since server is empty. Therefore, the generating function for the server content in a random batch is given by

$$\frac{V(1,y,1) - V(1,0,1)}{1 - V(1,0,1)}. \tag{2.24}$$

2.3.4 Queue content when the server is in vacation

Since $S_k = 0$ implies that the system is in vacation in k^{th} slot, the queue content during such a slot is given by the generating function

$$\frac{V(z,0,1)}{V(1,0,1)}. \tag{2.25}$$

2.4 QoS measures of data backup policy

In this section we focus on the performance measures that are of interest to the backup policy which were discussed in detail in section 1.2.1. The analysis done in section 2.2 directly gives us the first three QoS measures namely the backlog size, probability that a new connection is made and the probability that the backup server is busy. We also compute the age of the data at the beginning of a backup period, i.e. time spent by first data packet of a backup cycle waiting for backup to restart. In section 2.5 we evaluate the performance of the system by looking at the change in the first moment of these stochastic variables as a function of the load as well as the mean vacation time.

2.4.1 Backlog size

The queue content of the system is the size of the backlog of data packets which are waiting for service. A bigger backlog size puts more data at the risk of loss if the server crashes. Moreover, we have a finite capacity of buffer size in practice. Therefore, we would want to keep this quantity as small as possible. By the definition of generating function $V(z,y,x)$, the queue content has a generating function $V(z,1,1)$. The mean queue content equals $\frac{\partial V(z,1,1)}{\partial z}\Big|_{z=1}$.

2.4.2 Probability that a new connection is made

As mentioned previously, cloud service providers charge for new connections made to their infrastructure. A new connection is made whenever a new service period starts after a vacation period and we would like to compute the frequency of these new connections. Since the restarting conditions are probabilistic, we compute the probability of a new connection at a random slot, i.e., $\lim_{k\to\infty} P(S_{k+1} > 0, S_k = 0)$. A higher value of this probability measure would imply more connections are made in the long run. We can compute this probability in terms of known constants as below.

$$\begin{aligned} P(\text{New connection}) &= \lim_{k\to\infty} \Big(P(S_k = 0) - P(S_{k+1} = 0, S_k = 0) \Big) \\ &= \lim_{k\to\infty} \Big(P(S_k = 0, R_k = 1) + P(S_k = 0, R_k > 1) \\ &\quad - P(S_{k+1} = 0, S_k = 0, R_k = 1) - P(S_{k+1} = 0, S_k = 0, R_k > 1) \Big) \end{aligned}$$

$$= V(1,0,0) - \sum_{m=0}^{l-1} \lim_{k\to\infty} P(Q_k + A_k = m, S_{k+1} = 0, S_k = 0, R_k = 1)$$

$$= V(1,0,0) - \sum_{m=0}^{l-1} (1-\alpha_m) d_0(m) \tag{2.26}$$

Note that $d_0(0) \notin \mathbb{U}$; however one can use (2.18) to compute the value of this boundary probability. Using (2.26), we can also compute the probability of a new connection at a random service initiation opportunity. It is more practical to use this measure in analysis as we are restricting ourselves to the slots where the service can be potentially restarted.

$$P(\text{New Service at Service initiation opportunity})$$

$$= \lim_{j\to\infty} P(\tau_j = 1 | \tau_{j-1} = 0) = \frac{P(\text{New connection})}{V(1,0,0)}$$

$$= 1 - \sum_{m=0}^{l-1} \frac{(1-\alpha_m) d_0(m)}{V(1,0,0)}. \tag{2.27}$$

As for the backlog size, we would want to keep $P(\text{New connection at Service initiation opportunity})$ as small as possible.

2.4.3 Probability that the backup server is busy

If the backup server is busy for longer periods, more power would be utilized to keep it running. It would also imply that the connections established with the data nodes to serve data packets would have to be maintained for longer period. The duration for which the backup server are kept on is directly proportional to the probability that the backup server is busy. Since the backup server is busy iff it is not on vacation, the probability that it is busy is given by

$$1 - V(1,0,1). \tag{2.28}$$

2.4.4 Age of data at the beginning of a backup period

A data backup cycle consists of one vacation period followed by one service period. During a data backup cycle, one can observe that the first data packet arrival has to wait for the longest time for backup service to restart (illustrated in figure 2.4). We randomly select a data backup cycle and label the first data packet to arrive in that cycle as ν.

At any slot, we define the age of data based on the earliest packet in the system. It is defined as the time spent by the earliest packet waiting for restart of backup service. Clearly, under this definition if the backup service is ON or if there are no packets in the system, the age of data is 0. Moreover, the age of data attains it maximum just before the start of

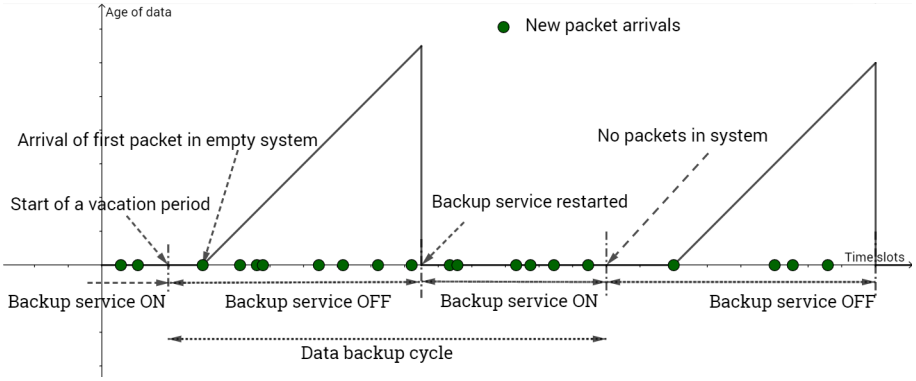


Figure 2.4: Illustration of evolution of age of data in a backup system

backup service (also illustrated in figure 2.4). Therefore, the total time spent waiting by ν is the age of data at the beginning of a backup period. We label this age as Age_ν. In this chapter, we will compute the mean of Age_ν and an in-depth analysis of the higher moments will follow in chapter 3.

In section 1.2.2, we introduced the recovery point objective (RPO) which is one of the most crucial parameters defined in literature of data backup processes. It defines the maximum time any data packet can wait in the data packets queue. To meet the QoS objectives defined by RPO, it is necessary to select the backup parameters such that Age_ν is within acceptable limits. By computing this QoS measure, we compute the impact of vacations on the waiting time of data packets.

We first compute the number of vacations ν witnesses, excluding the one in which it arrives, before the service is restarted. This will help us compute the amount of time it spends waiting for the backup service to restart. Note that ν might arrive in any vacation period after the service is stopped (not necessarily the first one) but those earlier vacation periods are not relevant as they are not part of age of ν. We refer to Figure 2.5 for an illustration of the delay of ν.

We observe the system at consecutive vacation epochs starting at the end of the vacation period during which ν arrives. We use the queue content at these epochs to define an absorbing Markov Chain for this process. That is, define the state space $\Omega = \{1, 2, \ldots, l-1, B\}$, where in state i, the system has i packets backlogged during a vacation while ν is in the system. State B corresponds to the absorbing state in which the backup process restarts. Notice that our backup system can transit from state a to state b before being absorbed in state B, where $a \leq b$, if there are exactly $b-a$ arrivals during the vacation period and the backup process does not start at the end of the vacation. The probability of this transition to happen is $(1-\alpha_b)t_A(b-a)$

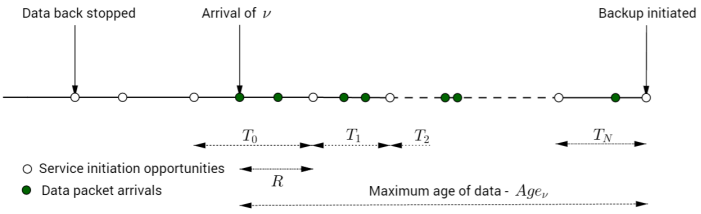


Figure 2.5: Illustration of delay of ν

where $t_A(n)$ is defined in section 2.2.4 (probability of n arrivals in a vacation period).

Since our system jumps between states until it finally gets absorbed, we can model the number of vacations seen by ν as a discrete phase-type distribution. This distribution is characterized by $(\boldsymbol{\beta}, M)$ where the state space is Ω. Such a distribution comprises of $l-1$ transient states ($\{1, 2, \dots l-1\}$) and 1 absorbing state (B) . The transition matrix $P = \begin{bmatrix} M & \boldsymbol{M^0} \\ \mathbf{0} & 1 \end{bmatrix}$ and $(\boldsymbol{\beta}, \beta_l)$ is the initial probability vector; M is given by

$$M = \begin{bmatrix} (1-\alpha_1)t_A(0) & (1-\alpha_2)t_A(1) & (1-\alpha_3)t_A(2) & \dots & (1-\alpha_{l-1})t_A(l-2) \\ 0 & (1-\alpha_2)t_A(0) & (1-\alpha_3)t_A(1) & \dots & (1-\alpha_{l-1})t_A(l-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & (1-\alpha_{l-1})t_A(0) \end{bmatrix} \tag{2.29}$$

and vector $\boldsymbol{M^0}$ can be calculated using the fact that P is a valid probability transition matrix, i.e., rows sum to 1.

Since the system is non-empty at the beginning of first (relevant) vacation period, our starting state is

$$\boldsymbol{\beta} = (\tfrac{(1-\alpha_1)t_A(1)}{1-t_A(0)}, \tfrac{(1-\alpha_2)t_A(2)}{1-t_A(0)}, \tfrac{(1-\alpha_3)t_A(3)}{1-t_A(0)}, \dots, \tfrac{(1-\alpha_{l-1})t_A(l-1)}{1-t_A(0)}), \beta_l = 1 - \sum_{i=1}^{l-1} \beta_i.$$

Define N as the number of phases till absorption to B, i.e., the number of vacation epochs seen by ν. The generating function of N is given by $N(z) = z \cdot \boldsymbol{\beta} \cdot (I - zM)^{-1} \cdot \boldsymbol{M^0} + \beta_l$ where I is the identity matrix (see [70]).
Using this, Age_ν can be written as

$$Age_\nu = \sum_{i=1}^{N} T_i + R,$$

where $T_i \sim T$ while R is the remaining vacation time of ν as shown in Figure 2.5. Since $E(T_n 1_{\{N \geq n\}}) = E(T_n)P(N \geq n)$, we can use Wald's equation (see

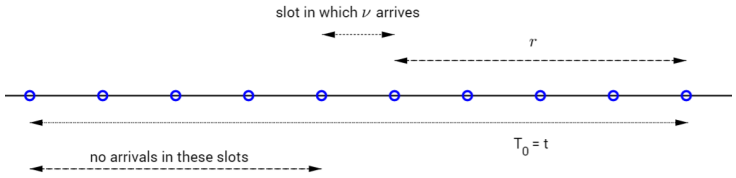


Figure 2.6: Illustration of arrival of ν to compute joint distribution of T_0 and R

[71]) to get the first moment of Age_ν as

$$E(Age_\nu) = E(N) \times E(T) + E(R). \tag{2.30}$$

We are left with the computation of $E(R)$. We can do this by analyzing the joint Probability Mass Function(PMF) of the length of vacation the first packet arrives in, T_0, and R. Note that the vacation time in which ν arrives, T_0 (see figure 2.5), does not have the same distribution as T. This is because there is at least one arrival in the vacation in which ν arrives. The joint pmf can therefore be written as, for $r \geq 0, t \geq 1$,

$$\begin{aligned} P(T_0 = t, R = r) &= P(T = t, R = r | \sum_{i=1}^{T} A_i \geq 1) \\ &= \frac{P(T = t, R = r, \sum_{i=1}^{T} A_i \geq 1)}{P(\sum_{i=1}^{T} A_i \geq 1)}. \end{aligned} \tag{2.31}$$

Looking at figure 2.6, we can see that $R = r$ when $T = t$ if there are no arrivals in the first $t - r - 1$ slots and at least one arrival in the slot in which ν arrives. Hence (2.31) can be written as

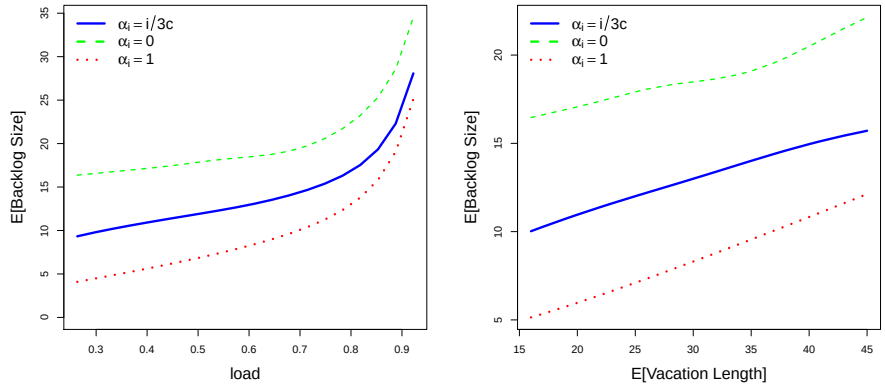
$$P(T_0 = t, R = r) = \frac{A(0)^{t-r-1}\big(1 - A(0)\big)P(T = t)}{1 - T(A(0))}. \tag{2.32}$$

From (2.32), we can calculate the first moment of the remaining vacation time by computing

$$E(R) = \sum_{r=0}^{\infty} \sum_{t=r+1}^{\infty} \frac{rA(0)^{t-r-1}\big(1 - A(0)\big)P(T = t)}{1 - T(A(0))}$$

which can be simplified to

$$E(R) = \frac{T'(1)}{1 - T(A(0))} - \frac{1}{1 - A(0)}. \tag{2.33}$$



(a) E[Vacation Length] = 30 (b) Load=0.6

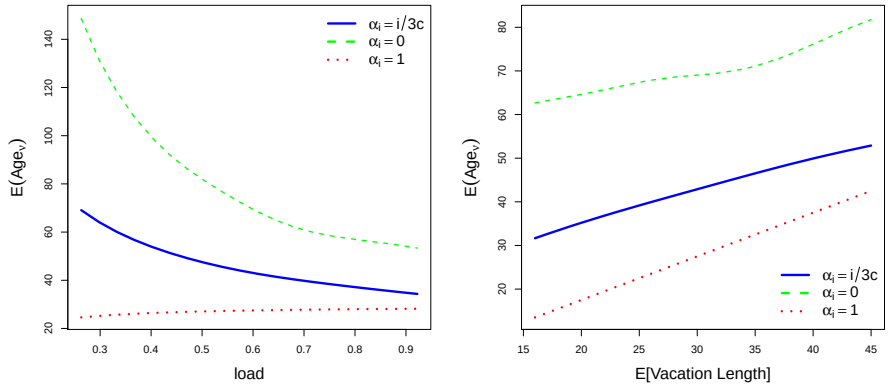
Figure 2.7: Mean backlog size

2.5 Numerical evaluation

To demonstrate the sensitivity of the QoS measures on the backup parameters, we evaluate the performance of backup system using realistic parameter values. Therefore, we create a scenario from synthetic data based on our experience working with real, yet proprietary data of an actual data backup system. We translate the backup parameters into model parameters of the batch service queueing system to analyze the performance. These measures are the key elements which can help us select the right parameters based on the performance requirements.

Backup system

We consider an organization which uses cloud services to backup the data on laptops of its employees using a centralized backup server. This data is generated due to incoming and outgoing emails, documents created or edited etc. From our practical experience and some real data, we define the parameters of the backup process. The average data generate by all users in 1 minute is between 20 to 100 MB data. The backup system on the other hand, has a bandwidth that is capable of serving at a maximum speed of 100 MB per minute. The data files are zipped into data packets of size 16 MB each and backup server can serve up to 10 packets in a single batch. Moreover, once the backup server enters a vacation period, it has to restart service if there are more than 30 packets in the backlog.



(a) E[Vacation Length] = 30 (b) Load=0.6

Figure 2.8: Mean age of data at the beginning of backup cycle

Queueing model

We define our slot length to be 10 sec. Using the information of the backup system, the model parameters are defined below:

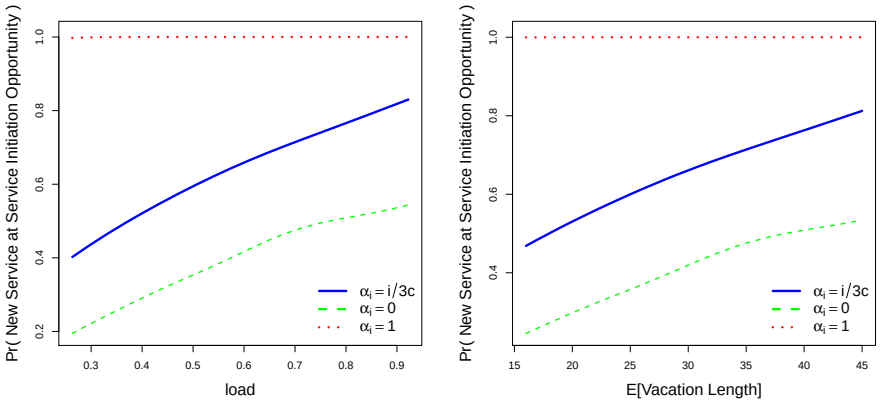
- It is well known that the arrival distribution has a heavy tail (see example [72]), i.e, there are some slots which will see a big burst of arrivals of packets. Therefore, we model the arrival distribution to be a mixture of Poisson and Power law distribution. The generating function of the number of arrivals in a slot can be written as

$$A(z) = p \times e^{\lambda(z-1)} + (1-p) \times \frac{\zeta_\gamma(z,a)}{\zeta_\gamma(1,a)}$$
$$\zeta_\gamma(z,a) = \sum_{k=a}^{\infty} \frac{z^k}{k^\gamma} \tag{2.34}$$

  where λ is the arrival rate of the Poisson process and γ is the order of power law distribution. For this example, we use $a = 25, p = 0.999, \gamma = 3.5$ while λ varies to model change in the system load. We also need to ensure that the system remains stable, i.e., the load of the system is less than 1.

- Since the backup server can serve up to 10 packets in a single batch, the batch server capacity $c = 10$.

- Since the backup service has to restart if there are more than 30 packets in the backlog, the restarting threshold $l = 30$.



(a) E[Vacation Length] = 30 (b) Load = 0.6

Figure 2.9: Probability of a new connection at a service initiation opportunity

- Since the writing speed, on an average, allows the backup server to write about 1 packet in a slot, we model the service time of m packets as discrete uniform distribution U$[m, m+2]$. Therefore, the $E(G_m) = m + 1$.

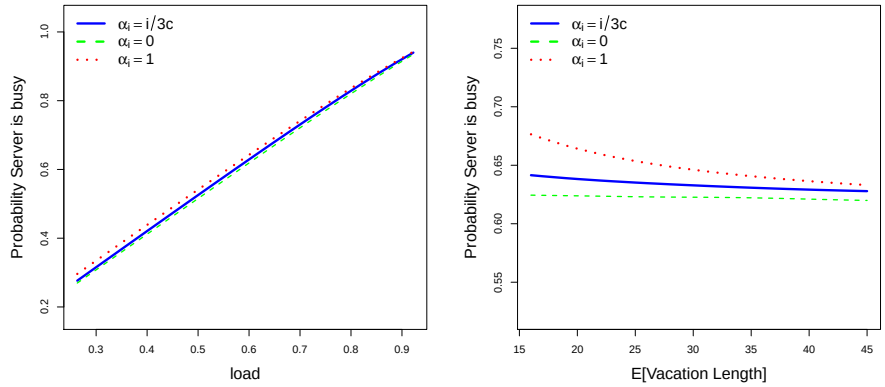
- We assume that the backup server goes into vacations of length approximately equal to 5 minutes. Therefore, the vacation period T is modeled as $U[28, 32]$, i.e., uniformly distributed discrete random variable between 28 and 32 slots. Moreover, while analyzing the impact of vacation lengths on the model, we model T as $U[v-2, v+2]$ such that the expected vacation length, $E(T) = v$.

It is important to note that we have used fairly simple distributions for service time and vacation periods. However, the analytic results obtained in the chapter are valid for any general distributions of service time, vacation periods and number of arrivals in a slot defined by generating functions $G_m(z), T(z)$ and $A(z)$ respectively.

2.5.1 Observations

In the graphs in figures 2.7-2.11 we show both the performance measures of queueing model as well as the QoS measures of the backup process under varying load or vacation length keeping other parameters constant. These graphs represent the first moment of the random variables defined in section 2.3 and section 2.4. Three policies are compared

1. $\alpha_i = 0, \forall 0 < i < l$, this corresponds to the policy where the service does not start until the server threshold is met.



(a) E[Vacation Length] = 30 (b) Load=0.6

Figure 2.10: Probability server busy

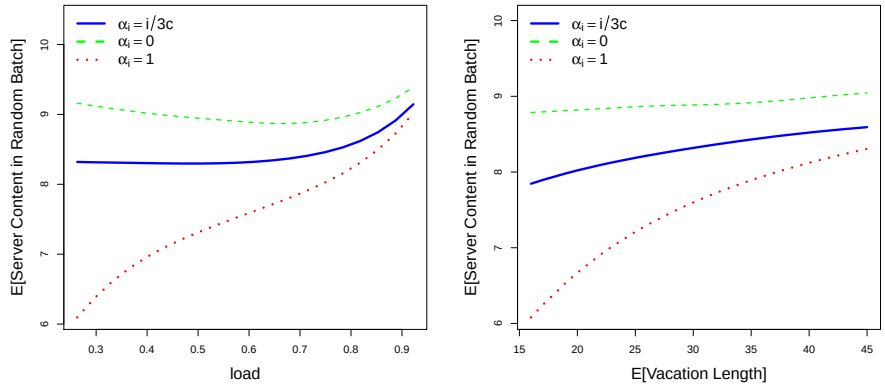
2. $\alpha_i = 1, \forall 0 < i < l$, this corresponds to the vacation policy which resumes service if there is at least one packet in the queue when the server wakes up from vacation. Equivalently, it can be defined as a policy with $l = 1$.

3. $\alpha_i = \frac{i}{3c}, \forall 0 < i < l$, this corresponds to the policy which starts with a probability that is linear to the number of packets in the queue.

Note that by comparing these policies, we are comparing the threshold policy with a policy which also has probabilistic restarts. Moreover, the policy with $\alpha_i = i/3c$ is an example policy that we have chosen for comparison while the model allows us to substitute any valid value. Our primary aim while doing this comparison is to observe the dependence of QoS measures on the model parameters.

While comparing these policies in Figures 2.7-2.11, the ideal performance for each QoS measure individually would be obtained either by policy (1) or (2). In particular, while policy (1) performs best for QoS measures related to energy consumption and connection cost, policy (2) performs best for QoS measures related to data safety. However, these policies individually are not able to meet all the expectations. We observe that the policy (3) is able to tackle the trade off and gives a performance which lies in between policy (1) and policy (2). Also, the choice of parameters of our model, αs, T, l, c, allows us to get closer to the desired QoS measures. We illustrate selection of optimal parameters in section 2.5.3 by defining the cost function of a user. One can make the following general observations from the graphs

- from figures 2.7 and 2.8, the values of mean backlog size and the age of data at the beginning of a backup period for policy (3) is in the middle of



(a) E[Vacation Length] = 30 (b) Load=0.6

Figure 2.11: Average server content in random batch

the two policies (1) and (2). Limiting ourselves to these measures, while the system performs better than policy (1), it performs worse than policy (2).

- from figure 2.9, using policy (3), the probability of a new connection at a service initiation opportunity reduces drastically as compared to policy (2). Although, this probability under policy (3) is higher when compared to policy (1), it is closer to policy (1) than (2).
- from figure 2.10, probability that the server is busy may be significantly lowered by using policy (3) as compared to policy (2). This probability is higher than the value obtained using policy (1) but fairly close to it.
- from figure 2.11, the mean server content in a random batch using policy (3) is higher than policy (2) and smaller than (1).

2.5.2 Illustrating the trade-off using cost function

In this section we use the QoS measures of the backup service computed in section 2.4 to define a cost function for a user. Our primary aim is to address the trade-off between doing frequent backups and reducing resource usage. The frequency of performing backups is dependent on all backup parameters and there are multiple ways of changing this frequency. For instance one can increase the values of restarting probabilities or decrease the restarting threshold l. Here, we change the frequency by changing the restarting threshold l.

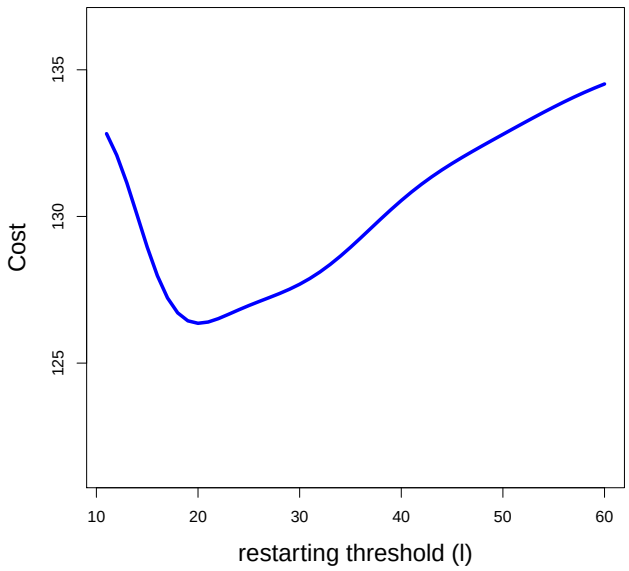


Figure 2.12: Cost function for varying frequency of backups

We define the cost function as a function of the restarting threshold l as follows

$$Cost(l) = 0.5 \times E(Age_\nu) + 20 \times e^{P(\text{ New connection })} + 0.25 \times E(\text{ Backlog content}) + 15 \times e^{P(\text{ Server busy })} \quad (2.35)$$

Further, we define $\alpha_i = min(1, \frac{i}{100})$ for $i < l$. The remaining parameters c and ρ are kept constant at 10 and 0.6. Figure 2.12 shows the change in the cost function with change in the backup frequency. Clearly, the backup frequency has to be chosen optimally which can be done by selecting the backup parameters optimally. We show how we can compute the optimal backup parameters in the next paragraph.

2.5.3 Cost optimization

In contrast to the comparisons done in section 2.5.1 between three particular policies, in this section we look at computing the optimal parameters for our model. In section 2.5.2, we defined a cost function of the user to highlight the need of selecting the optimal backup parameters. However, to compute optimal parameters, it is more appropriate to define the user objective as a combination of cost function and QoS constraints. We use the four QoS measures, $E(Age_\nu)$,

$P($ New connection $)$, $E($ Backlog content $)$, $P($ Server busy $)$ computed in section 2.4 to define the following cost function

$$W(l, c, \alpha_1, \ldots, \alpha_{l-1}) = 20 \times e^{P(\text{ New connection })} + 15 \times e^{P(\text{ Server busy })}$$

and QoS constraints of a user as

$$E(Age_\nu) \leq 30, \qquad E(\text{ Backlog content }) \leq 20. \tag{2.36}$$

For illustration purposes, we minimize this function by varying the starting probabilities α_is and threshold l while keeping the server capacity c constant. More precisely, we propose a simple mechanism to find the optimal l and $\alpha_1, \alpha_2, \ldots, \alpha_{l-1}$. Our primary aim is to show that the optimal policy may not be a pure threshold policy.

A pure threshold policy can be represented as $\alpha_i = 0, \forall i < m$ and $\alpha_i = 1, \forall i \geq m$, where m defines the threshold. Further, a pure threshold policy with threshold which minimizes cost is called the optimal pure threshold policy.

The parameters used here are as defined in the beginning of section 2.5. For a fixed load ρ, the task of finding the optimal l and α_is for our model is equivalent to solving the following optimization problem

$$\begin{aligned}
\underset{\alpha_1, \ldots \alpha_{l-1}, l}{\text{minimize}} \quad & W(l, \alpha_1, \alpha_2, \ldots, \alpha_{l-1}) \\
\text{subject to:} \quad & E(Age_\nu) \leq 30 \\
& E(\text{ mean backlog content }) \leq 20 \\
& 0 \leq \alpha_i \leq 1, \; i = 1, \ldots, (l-1) \\
& \alpha_i = 1, i \geq l, \qquad l \leq l_{max}
\end{aligned}$$

We use $l_{max} = 15$ to find the solution. For a given load ρ, solution of this problem gives us the optimal parameters l^*, α_i^*s. We use an iterative solver, $auglag()$ in **R**, to solve the above non-linear optimization problem. We set the starting point of the solver as the optimal pure threshold policy. Therefore, the solution found by the solver would be better than the optimal pure threshold policy. These may not be globally optimal solutions; however, we have shown that optimal threshold policies with probabilistic restarts perform better than optimal pure threshold policies with threshold l less than l_{max}.

In Figure 2.13 we compare the cost graph of the user using the optimal parameters with pure optimal threshold based policies. Note that we have considered a fairly simple optimization problem. One could also have constraints as well as utility function dependent on higher moments of QoS measures such as $Var($mean backlog content $)$ which can be derived using the generating functions computed in paragraph 2.3.

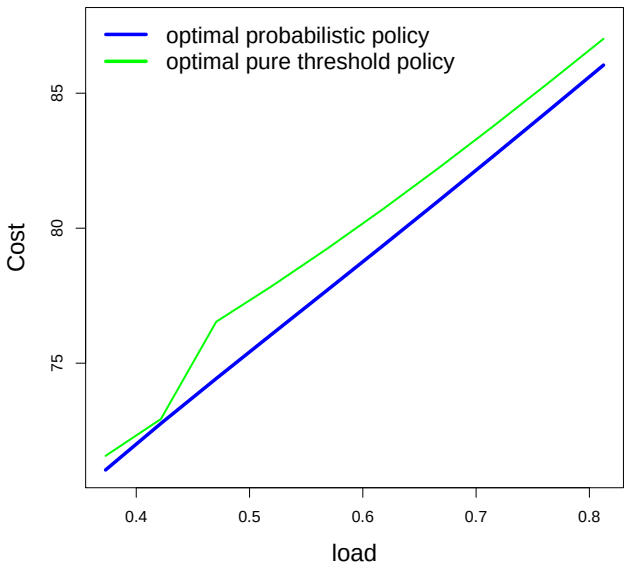


Figure 2.13: Cost of doing backup for varying load

2.5.4 Final insights

In section 2.5.1 we illustrate the dependence of QoS measures on the service parameters. In section 2.5.3 we look at an optimization problem to compute the system parameters using the cost function of the user. The key observations are

- a policy with probabilistic restarting conditions is able to address the trade off between performing backups too frequently and reducing resource consumption.
- for a give system load ρ, the probabilistic restarting policy with optimal parameters α_i^ρ s may outperform the fixed threshold policies.
- the QoS measures can be precisely computed numerically and the parameters can be chosen to meet the desired QoS.
- the performance changes significantly if the model parameters are varied. Therefore, users with different desired QoS need to be treated differently.

2.6 Summary

We have been able to model exhaustive data backup processes as a very general queueing process with batch service, vacations and probabilistic restarting conditions. Moreover, the number of arrivals in a slot and service time can have any general distribution. We have shown it is natural to model data backup processes this way. By computing the QoS measures and analyzing their dependence on the system parameters, we have illustrated how one can strike a balance between doing frequent backups and reducing resource usage for a given QoS. Our observations in section 2.5.1 summarize the dependence of performance measures on model parameters. Selecting the optimal parameters allows the system to take longer vacations or support a higher load for the same QoS. For a given system, such a model empowers us to select the parameters based on the arrival process and desired system performance while keeping the system stable. We have illustrated this using an example of how to use the cost function of the user to select the starting probability parameters.

3

Analysis of age of data

In this chapter, we study a similar discrete-time queueing model described in Chapter 2 with the aim to calculate the distribution of the age of data just before a backup starts (Age_ν). In a real backup system, the backup service provider decides the configuration and schedule of backup operations as well as the communication with the cloud. The backup server alternates between backup-on and backup-off periods where the backup-on periods are much smaller than the backup-off periods. And because the age of data is insensitive to the batch server capacity, in the analysis we will assume that the batch server has an unlimited capacity. The age of the data is defined as the waiting time of the oldest packet in the system just before a backup starts. We focus on the age of the data (Age_ν) because it is closely related to the Recovery Point Objective (RPO) described in section 1.2.2.

The distribution of this age is important for various reasons. In addition to the mean value, the higher moments of a performance measure such as variance determine the performance of the system. For instance, a user may prefer lower variance of a performance measure over lower mean value. In Chapter 2 we were successful in computing only the first moment of Age_ν. In this chapter, we compute the generating function of the distribution of Age_ν using recursive relations. Using this generating function, one can then compute all moments of Age_ν.

Moreover, the tail probabilities represent the exceptional situations in which Age_ν can attain really large values. Backup service providers guarantee high QoS and even promise very heavy compensation in case of breach of SLA. For instance, [13] gives 100% service credits if the RPO is more than 4 hours. Similarly, [73], an IT solutions provider, gives similar guarantees of RPO between 15mins-24hrs for different service costs. To provide these guarantees, they charge users based on their usage. The costs associated with running backup operations has been discussed in section 1.2.1. Therefore, the probability that such extreme situations occur is an important performance metric. We analyze the tail of the distribution of Age_ν and scenarios in

which a user might see such a breach. In our analysis, we use the recursive relations and the dominant singularity of the generating function of Age_ν to compute the asymptotic behavior of the distribution of the age of data in the backup system. Note that the backup service provider just provides services and management of the backup process and may not actually own the cloud infrastructure. They would be hired by an organization to provide these storage solutions or could be a team within the organization. Such an model of service is known as Software as a Service (SaaS) (see [7]).

Some storage systems utilize a timer mechanism to avoid the backup-off period to become very long. The analysis of the model studied in this chapter is used to find the optimal timer length for storage systems with time mechanism. In particular, our analysis can be used to measure the performance of such systems in terms of age of data and the backup frequency. More details about this are presented in section 3.5.

We start with calculating some generating functions that are required in the further sections. It is followed by computation of generating function of Age_ν and the analysis of its tail distribution. In the last sections we analyze the storage systems with time trigger mechanism and present a numerical example.

Table 1: List of notations used in the analysis	
l	the restarting threshold of the backlog size.
T	the random variable which denotes the length of any random vacation, its generating function is denoted by $T(z)$.
$A(z)$	the generating function of the number of arrivals in a single slot.
α_i	the probability of restart of the batch service at the end of a vacation if there are i packets backlogged.
R	remaining vacation time, i.e., the number of slots remaining in the current vacation epoch
T_0	the length of the vacation in which ν arrives

3.1 Some useful generating functions

In chapter 2 we computed the first moment of the distribution of Age_ν. In this chapter we will compute the PGF of Age_ν. This computation requires a few results to be established first. In particular we compute $R(z)$, the generating function of the remaining vacation time, $S(y, z)$, the joint generating function of the length of remaining vacation time and the number of arrivals in T_0, and $W(y, z)$, the joint generating function of the length of a random vacation and the number of arrivals in it.

Lemma 1. *The generating function of the length of the remaining vacation R*

is given by

$$R(z) = \frac{\big[1 - A(0)\big]\big[T(A(0)) - T(z)\big]}{\big[1 - T(A(0))\big]\big[A(0) - z\big]}. \tag{3.1}$$

Proof. Using formula (2.14), this generating function can be written as the following sum

$$\begin{aligned} R(z) &= \sum_{t=1}^{\infty} \sum_{r=0}^{t-1} z^r P(T_0 = t, R = r) \\ &= \sum_{t=1}^{\infty} \frac{A(0)^{t-1}[1 - A(0)]P(T = t)}{1 - T(A(0))} \sum_{r=0}^{t-1} \frac{z^r}{A(0)^r} \\ &= \frac{\big[1 - A(0)\big]\big[T(A(0)) - T(z)\big]}{\big[1 - T(A(0))\big]\big[A(0) - z\big]}. \end{aligned}$$

□

Lemma 2. *The joint generating function of the length of the remaining vacation R and the number of arrivals in T_0, n_{T_0}, is given by*

$$S(y, z) = E(y^{n_{T_0}} z^R) = \frac{\big[A(y) - A(0)\big]\big[T(zA(y)) - T(A(0))\big]}{\big[zA(y) - A(0)\big]\big[1 - T(A(0))\big]}. \tag{3.2}$$

Proof. Note that the slot in which ν arrives is not considered as part of R and can have more than one arrival. The generation function $S(y, z)$ can be written as the following sum

$$\begin{aligned} S(y, z) = \sum_{i=1}^{\infty} \sum_{r=0}^{\infty} y^i z^r P(\ & i \text{ arrivals in } r+1 \text{ slots} \ \big|\ \text{at least one arrival} \\ & \text{in slot 1 }) P(R = r) \\ = \sum_{r=0}^{\infty} A(y)^r \frac{A(y) - A(0)}{1 - A(0)} z^r P(R = r) \\ = \frac{A(y) - A(0)}{1 - A(0)} R(zA(y)). \end{aligned}$$

Using the results of Lemma 1, we arrive at the following expression,

$$S(y, z) = \frac{\big[A(y) - A(0)\big]\big[T(zA(y)) - T(A(0))\big]}{\big[zA(y) - A(0)\big]\big[1 - T(A(0))\big]}.$$

□

We can write the series expansion of $S(y, z)$ in its radius of convergence as

$$S(y, z) = \sum_{i=1}^{\infty} r_i(z) y^i. \tag{3.3}$$

The coefficients $r_i(z)$ give us the partial generating functions of the length of the remaining vacation slots R with total of i arrivals in the whole vacation T_0. We will use $r_i(z)$ in our analysis in the further sections.

Lemma 3. *The joint generating function of the length of the vacation $T_i, i \geq 1$, and the number of arrivals in it n_T is given by*

$$W(y,z) = E(y^{n_T} z^T) = T(zA(y)). \tag{3.4}$$

Proof. This generating function can be written as the following sum

$$\begin{aligned} W(y,z) &= \sum_{t=1}^{\infty} \sum_{j=0}^{\infty} y^j z^t P(\ j \text{ arrivals in } t \text{ slots }) P(T=t) \\ &= \sum_{t=1}^{\infty} A(y)^t z^t P(T=t) = T(zA(y)). \end{aligned}$$

□

We can write the series expansion of $W(y,z)$ in its radius of convergence as

$$W(y,z) = \sum_{i=0}^{\infty} t_i(z) y^i. \tag{3.5}$$

The coefficients $t_i(z)$ are the partial generating functions of a vacation length with i arrivals.

3.2 Generating function of Age_ν

In this section we compute the generating function of Age_ν. Therefore, define

- $h(t) = P(Age_\nu = t)$: the probability mass function of Age_ν.
- $\phi_i(t)$: Given the backlog size equals i at the beginning of a vacation, $\phi_i(t)$ is the probability mass function of the remaining time until the restart of the batch service.

The end of each vacation gives the system a *service initiation opportunity*. The backup server then decides to either restart the batch service or enter a new vacation. To find the distribution of Age_ν, we first condition on the action chosen by the backup server at the end of vacation T_0. That is, we write

$$\begin{aligned} h(t) &= P(Age_\nu = t \cap \text{ backup service restarts after vacation } T_0) \\ &\quad + P(Age_\nu = t \cap \text{ backup service does not restart after vacation } T_0) \\ &= \sum_{i=1}^{l-1} \alpha_i P(i \text{ arrivals in } t+1 \text{ slots } \Big|\ \text{at least 1 arrival in slot 1 }) P(R=t) \end{aligned}$$

$$+\sum_{i=l}^{\infty} P(i \text{ arrivals in } t+1 \text{ slots } \Big| \text{ at least 1 arrival in slot 1 })P(R=t)$$
$$+\sum_{k=0}^{t-1}\sum_{i=1}^{l-1} P(i \text{ arrivals in } k+1 \text{ slots } \Big| \text{ at least 1 arrival in slot 1})$$
$$\times (1-\alpha_i)P(R=k)\phi_i(t-k). \tag{3.6}$$

Similarly, $\phi_i(t)$ can be written recursively as

$$\begin{aligned}\phi_i(t) = &\sum_{j=i}^{l-1} \alpha_j P(j-i \text{ arrivals in } t \text{ slots })P(T=t)\\ &+\sum_{j=l}^{\infty} P(j-i \text{ arrivals in } t \text{ slots })P(T=t)\\ &+\sum_{k=1}^{t-1}\sum_{j=i}^{l-1}(1-\alpha_j)P(j-i \text{ arrivals in } k \text{ slots})P(T=k)\phi_j(t-k).\end{aligned} \tag{3.7}$$

We will now transform eqn (3.6) and eqn (3.7) to generating functions. Therefore, we define

$$H(z)=\sum_{t=0}^{\infty} z^t h(t), \qquad \Phi_i(z)=\sum_{t=1}^{\infty} z^t \phi_i(t), \qquad i=1\ldots l-1.$$

the generating functions of the distributions $h(t)$ and $\phi_i(t)$. We can then use eqn (3.6) to write

$$\begin{aligned}H(z) = &\sum_{t=0}^{\infty}\sum_{i=1}^{l-1}(\alpha_i-1)z^t P(i \text{ arrivals in } t+1 \text{ slots } \Big| \text{ at least 1 arrival}\\ &\text{in slot 1 })P(R=t)+R(z)\\ &+\sum_{t=1}^{\infty}\sum_{k=0}^{t-1}\sum_{i=1}^{l-1} z^t(1-\alpha_i)P(i \text{ arrivals in } k+1 \text{ slots } \Big| \text{ at least 1}\\ &\text{arrival in slot 1 })P(R=k)\phi_i(t-k)\\ =&\sum_{i=1}^{l-1}(\alpha_i-1)r_i(z)+R(z)+\sum_{i=1}^{l-1}(1-\alpha_i)\sum_{k=0}^{\infty}P(i \text{ arrivals in } k+1 \text{ slots}\\ &\text{with at least 1 arrival in slot 1})\times P(R=k)\sum_{t=k+1}^{\infty} z^t\phi_i(t-k)\\ =&\sum_{i=1}^{l-1}(\alpha_i-1)r_i(z)+R(z)+\sum_{i=1}^{l-1}(1-\alpha_i)\Phi_i(z)\sum_{k=0}^{\infty} z^k P(i \text{ arrivals in}\end{aligned}$$

$$k+1 \text{ slots with at least 1 arrival in slot 1})P(R=k)$$
$$= \sum_{i=1}^{l-1}(\alpha_i - 1)r_i(z) + R(z) + \sum_{i=1}^{l-1}(1-\alpha_i)\Phi_i(z)r_i(z).$$

Similar calculations using eqn (3.7) are possible. We end up with the following recursion:

$$\Phi_i(z) = T(z) + \sum_{j=i}^{l-1}(1-\alpha_j)\Big(\Phi_j(z) - 1\Big)t_{j-i}(z), \qquad \forall 1 \le i \le l-1. \tag{3.8}$$

We summarize our results in the following theorem.

Theorem 3.1. *The generating function of Age_ν, $H(z)$, is given by*

$$H(z) = R(z) + \sum_{i=1}^{l-1}(1-\alpha_i)r_i(z)\Big(\Phi_i(z) - 1\Big), \tag{3.9}$$

$$\Phi_i(z) = \frac{1}{1-(1-\alpha_i)t_0(z)}\Bigg(T(z) - (1-\alpha_i)t_0(z) + \sum_{j=i+1}^{l-1}(1-\alpha_j)t_{j-i}(z)\Big(\Phi_j(z)-1\Big)\Bigg) \forall 1 \le i \le l-1, \tag{3.10}$$

where $r_i(z)$ and $t_i(z)$ can be computed from eqns (3.2-3.5). Note that the functions $\Phi_i(z)$ can be computed recursively starting from $\Phi_{l-1}(z)$.

3.3 Mean and variance of Age_ν

Using eqn (3.9) and (3.10) one can compute moments of Age_ν. First, the mean value is given by

$$E(Age_\nu) = H'(1) = R'(1) + \sum_{i=1}^{l-1}(1-\alpha_i)r_i(1)\Phi_i'(1),$$
$$\Phi_i'(1) = \frac{1}{1-(1-\alpha_i)t_0(1)}\Big(T'(1) + \sum_{j=i+1}^{l-1}(1-\alpha_j)\Phi_j'(1)t_{j-i}(1)\Big) \qquad \forall 1 \le i \le l-1. \tag{3.11}$$

We verified that the mean of Age_ν obtained using eqn (3.11) is exactly the same as obtained in chapter 2 where Wald's equation was used to obtain the mean value. Similarly, we can use the second derivatives of $H(z)$ and $\Phi_i(z)$ to compute the variance of Age_ν.

$$H''(1) = R''(1) + \sum_{i=1}^{l-1}(1-\alpha_i)\Big(2r_i'(1)\Phi_i'(1) + \Phi_i''(1)r_i(1)\Big),$$

$$\Phi_i''(1)\Big(1-(1-\alpha_i)t_0(1)\Big) = T''(1) + 2\Phi_i'(1)(1-\alpha_i)t_0'(1) + \sum_{j=i+1}^{l-1}(1-\alpha_j)\Big(t_{j-i}(1)\Phi_j''(1) + 2\Phi_j'(1)t_{j-i}'(1)\Big),$$

$$Var(Age_\nu) = H''(1) + H'(1) - H'(1)^2. \tag{3.12}$$

Computing $r_i^{(n)}(1), t_i^{(n)}(1)$, the n^{th} derivatives of the partial generating functions, involves inversion of derivatives of the joint generating functions in eqn (3.3) and eqn (3.5). This is a non trivial task for a general distribution function. However, for well behaved functions such as rational functions, we can compute the derivatives to get $r_i^{(n)}(1), t_i^{(n)}(1)$. One can also use a numerical inversion algorithms, such as Fourier series method in [74], to compute these values.

For a Poisson arrival process and vacation lengths distribution with support in $[a, b]$ (i.e. $P(a \leq T \leq b) = 1$), the coefficients and their derivatives can be directly computed and have a closed form expression given below.

$$t_i(1) = \sum_{t=a}^{b} e^{-\lambda t}(\lambda t)^i \frac{P(T=t)}{i!},$$

$$t_i'(1) = \sum_{t=a}^{b} te^{-\lambda t}(\lambda t)^i \frac{P(T=t)}{i!},$$

$$r_i(1) = \frac{t_i(1)}{1-T(A(0))},$$

$$r_i'(1) = \frac{t_i'(1)}{1-T(A(0))} - \frac{1}{1-T(A(0))}\sum_{t=a}^{b} e^{-\lambda t}\lambda^i \frac{P(T=t)}{i!}\sum_{s=1}^{t} s^i.$$

These constants can then be used to compute the mean and variance of Age_ν.

3.4 Tail asymptotics

As mentioned earlier in this chapter, computing the tail probabilities is of equal importance, since extreme events result in a high financial loss. Therefore, it is important to analyze the cases in which this age can take high values and compute the probability of their occurrence. Therefore, in this section we compute the tail asymptotic of Age_ν using the generating function $H(z)$. For this analysis we carry out dominant singularity analysis of $H(z)$. For the ease of presentation we make some naturally applicable assumptions stated below. It is important to note that this analysis is valid even when these assumptions are not met but would complicate the presentation.

Assumptions:

1. Intuitively, an efficient data backup policy should restart the backup service with higher probability if there are more packets in the backlog. This is because when more packets are waiting for service, there are more packets at risk. Therefore, we assume that α_i is non decreasing, i.e., $i < j$ implies $\alpha_i \leq \alpha_j$.

2. The first s restarting probabilities are equal, i.e., $\alpha_1 = \alpha_2 = \ldots = \alpha_s$. This is not a restriction as any value of s between 1 and $l-1$ is allowed. However, the value of s has an impact on tail probabilities.

3. $T(z)$, the generating function of the vacation period is defined by the user. Since it is a user defined function, it is reasonable to expect it to be analytic everywhere. Therefore, we assume that $T(z)$ does not have any singularities. This implies that $R(z)$, $t_i(z)$ and $r_i(z)$ also do not have any singularities.

A function $f(z)$ is said to have a singularity at a point z_0 if it is not analytic at this point. Further, a dominant singularity of a function is a singularity with smallest absolute value among all singularities. In this work, we make extensive use of the following result (see [45] page 436, Theorem VI.14 for details).

Theorem 3.2 (Darboux theorem)**.** *Suppose the power series* $X(z) = \sum_{n=0}^{\infty} x(n)z^n$ *with positive real coefficients* $x(n)$ *is analytic near 0 and has only algebraic singularities* σ_k *on its circle of convergence* $|z| = \Re_X$*, in other words, in a neighborhood of* σ_k *we have*

$$X(z) \sim \left(1 - \frac{z}{\sigma_k}\right)^{-w_k} Y_k(z),$$

where $w_k \in \mathbb{C}\backslash\{0, -1, -2, \ldots,\}$ *and* $Y_k(z)$ *denotes a nonzero analytic function near* σ_k*. Let* $w = max_k Re(w_k)$ *denote the maximum of the real parts of the* w_k*. Then we have*

$$x(n) = \sum_k \frac{Y_k(\sigma_k)}{\Gamma(w_k)} n^{w_k - 1} \sigma_k^{-n} + o(n^{w-1}\Re^{-n}),$$

with the sum taken over all k *with* $Re(w_k) = w$ *and* $\Gamma(w)$ *the Gamma-function of* w *(with* $\Gamma(n) = (n-1)!$ *for* n *discrete).*

Therefore, if the dominant singularities, $\sigma_k \in \mathbb{C}$, of the generating function $X(z)$ and the behavior of the generating function in the neighborhood of these dominant singularities (w_k and $Y_k(\sigma_k)$) are identified, this theorem expresses an approximation for the tail of the corresponding distribution ($x(n)$ for large n).

From eqn (3.9), $H(z)$ contains the generating functions $\Phi_i(z)$. Therefore, we start with the analysis of $\Phi_i(z)$ to compute the singularities of $H(z)$.

Lemma 4. *The singularities of $\Phi_i(z)$ are given by*

$$\{z : \prod_{j=i}^{l-1} \big(1-(1-\alpha_j)t_0(z)\big) = 0\}.$$

Proof. From eqn (3.10), we can compute

$$\Phi_{l-1}(z) = \frac{T(z)-(1-\alpha_{l-1})t_0(z)}{1-(1-\alpha_{l-1})t_0(z)}.$$

Clearly, $\Phi_{l-1}(z)$ has a singularity of order 1 at the zeros of $1-(1-\alpha_{l-1})t_0(z)$, which proves the lemma for $i = l-1$. For $i = 1, \ldots, l-2$ we will prove this result by induction, i.e., for all $m > i$ assume $\Phi_m(z)$ has singularities at

$$\{z : \prod_{j=m}^{l-1} \big(1-(1-\alpha_j)t_0(z)\big) = 0\},$$

we will show that the result holds for $\Phi_i(z)$. We can rewrite eqn (3.10) as,

$$\Phi_i(z) = \frac{1}{1-(1-\alpha_i)t_0(z)}\Bigg(T(z)-(1-\alpha_i)t_0(z)-(1-\alpha_{i+1})t_1(z) + \sum_{j=i+2}^{l-1}(1-\alpha_j)t_{j-i}(z)\Big(\Phi_j(z)-1\Big)\Bigg) + (1-\alpha_{i+1})t_1(z)\frac{\Phi_{i+1}(z)}{1-(1-\alpha_i)t_0(z)}.$$

Hence, the singularities of $\Phi_i(z)$ are the singularities of $\Phi_{i+1}(z)$ and the zeros of $1-(1-\alpha_i)t_0(z)$. Therefore, these singularities are given by

$$\{z : \prod_{j=i}^{l-1} \big(1-(1-\alpha_j)t_0(z)\big) = 0\}.$$

□

Lemma 5. *The singularities of $H(z)$ are given by zeros of*

$$\prod_{i=1}^{l-1} \big(1-(1-\alpha_i)t_0(z)\big) = 0, \tag{3.13}$$

where the dominant singularities are the solutions of

$$t_0(z) = \frac{1}{1-\alpha_1}. \tag{3.14}$$

Moreover, the order of these singularities is equal to s.

Proof. From eqn (3.9) and (3.10), $H(z)$ is a linear combination of $\Phi_1(z), \ldots, \Phi_{l-1}(z)$. Therefore, $H(z)$ has a singularity at every singularity of $\Phi_i(z), 1 \leq i < l$. Note that there are no additional singularities of $H(z)$ which come from $T(z)$ and $R(z)$ because of assumption 3. This proves the first part of the lemma.

From eqns (3.4) and (3.5), we find $t_0(z) = T(zA(0))$, i.e. $t_0(z)$ is the partial generating function of the length of a vacation with 0 arrivals.

From Pringsheims Theorem, [45] page 240 Theorem IV.6, we know that for probability generating functions, at least one dominant singularity lies on the real axis. Moreover, for $x \in \mathbb{R}$, $T(x)$ is an increasing function of x as it is a probability generating function. Since α_i are non decreasing in i,

$$\min_i \left(\frac{1}{1-\alpha_i} \right) = \frac{1}{1-\alpha_1},$$

and dominant singularities on the real axis are given by

$$\min\{z : t_0(z) = \frac{1}{1-\alpha_i}, 0 < i < l, z \in \mathbb{R}\} = \min\{z : t_0(z) = \frac{1}{1-\alpha_1}, z \in \mathbb{R}\}.$$

Moreover, since all the dominant singularities have the same modulus (equal to the radius of convergence), all the dominant singularities are given by the solutions of $(1-\alpha_1)t_0(z) = 1$.

From assumption 2, we know that

$$\alpha_1 = \alpha_2 = \ldots = \alpha_s < \alpha_{s+1} \leq \alpha_{s+2} \leq \ldots \alpha_{l-2} \leq \alpha_{l-1}.$$

Using this relation, the singularities of $H(z)$ can be rewritten as

$$\big(1-(1-\alpha_1)t_0(z)\big)^s \prod_{i=s+1}^{l-1} \big(1-(1-\alpha_i)t_0(z)\big) = 0.$$

Therefore, the order of each dominant singularity equals s. □

Note that if we did not have assumption 1, eqn (3.14) would still be valid with α_1 replaced by $\alpha_{min} = \min_i \alpha_i$.

Lemma 6. *For $1 \leq i \leq s$, $\Phi_i(z)$ has a singularity at σ_k of the order of $s-i+1$ where σ_k is the k^{th} dominant singularity of $H(z)$.*

Proof. From Lemma 4, singularities of $\Phi_i(z)$ are solutions of

$$\prod_{j=i}^{l-1} \big(1-(1-\alpha_j)t_0(z)\big) = 0,$$

which can be rewritten as

$$\prod_{j=i}^{s}\big(1-(1-\alpha_j)t_0(z)\big)\prod_{j=s+1}^{l-1}\big(1-(1-\alpha_j)t_0(z)\big)=0. \tag{3.15}$$

We know that $\alpha_1=\alpha_2\ldots=\alpha_s$. Therefore eqn (3.15) can be rewritten as

$$\big(1-(1-\alpha_1)t_0(z)\big)^{s-i+1}\prod_{j=s+1}^{l-1}\big(1-(1-\alpha_j)t_0(z)\big)=0.$$

Moreover, from Lemma 5, if σ_k is a dominant singularity of $H(z)$, $1-(1-\alpha_1)t_0(\sigma_k)=0$. Therefore, $\Phi_i(z)$ also has a singularity at σ_k of the order of $s-i+1$. □

We now compute the coefficient $Y_k(\sigma_k)$ which defines the behavior of the generating function $H(z)$ in the neighbourhood of the k^{th} dominant singularity σ_k.

Theorem 3.3. *The tail distribution of Age_ν is given by*

$$P(Age_\nu=n)\sim\sum_{k=1}^{M}Y_k(\sigma_k)\frac{n^{s-1}}{(s-1)!\sigma_k^n}, \tag{3.16}$$

where M is the total number of zeros of $t_0(z)=\frac{1}{1-\alpha_1}$ with smallest norm and

$$Y_k(\sigma_k)=\frac{r_1(\sigma_k)[t_1(\sigma_k)]^{s-1}}{[T'(\sigma_kA(0))A(0)]^s}\bigg(T(\sigma_k)-1-\sum_{j=s+1}^{l-1}(1-\alpha_j)t_{j-s}(\sigma_k)$$
$$+\sum_{j=s+1}^{l-1}(1-\alpha_j)t_{j-s}(\sigma_k)\Phi_j(\sigma_k)\bigg). \tag{3.17}$$

Proof. Since σ_k is a dominant singularity of $H(z)$ of order s, $Y_k(\sigma_k)$ (see theorem 3.2) is given by

$$\begin{aligned}Y_k(\sigma_k)&=\lim_{z\to\sigma_k}(\sigma_k-z)^sH(z)\\&=\lim_{z\to\sigma_k}(\sigma_k-z)^s\bigg(R(z)+\sum_{i=1}^{l-1}(1-\alpha_i)r_i(z)(\Phi_i(z)-1)\bigg).\end{aligned}$$

From Lemma 6, we know that for $1\le i\le s$, $\Phi_i(z)$ has a singularity of order $s-i+1$ at σ_k. Moreover, $\Phi_i(z)$, $i>s$, does not have a singularity at σ_k. Therefore, the expression for $Y_k(\sigma_k)$ simplifies to

$$\begin{aligned}Y_k(\sigma_k)&=\lim_{z\to\sigma_k}(\sigma_k-z)^s(1-\alpha_1)r_1(z)\Phi_1(z)\\&=(1-\alpha_1)r_1(\sigma_k)\lim_{z\to\sigma_k}(\sigma_k-z)^s\Phi_1(z).\end{aligned} \tag{3.18}$$

Therefore, to compute $Y_k(\sigma_k)$, we need to compute the c_{Φ_1}, the constant which defines the behavior of $\Phi_1(z)$ around the dominant singularity σ_k.

$$c_{\Phi_1} = \lim_{z\to\sigma_k} (\sigma_k - z)^s \Phi_1(z).$$

Using eqn (3.10), we can write

$$\Phi_i(z) = \frac{T(z) - \sum_{j=i}^{l-1}(1-\alpha_j)t_{j-1}(z) + \sum_{j=s+1}^{l-1}(1-\alpha_j)t_{j-1}(z)\Phi_j(z)}{1-(1-\alpha_s)t_0(z)} + \frac{(1-\alpha_s)\sum_{j=i+1}^{s} t_{j-1}(z)\Phi_j(z)}{1-(1-\alpha_s)t_0(z)}. \tag{3.19}$$

To compute c_{Φ_1}, we first compute the coefficient of $\frac{1}{[1-(1-\alpha_s)t_0(z)]^s}$ in eqn (3.19) recursively which results in

$$(1-\alpha_s)^{s-1} t_1(z)^{s-1} \Bigg(T(z) - \sum_{j=s}^{l-1}(1-\alpha_j)t_{j-s}(z) + \sum_{j=s+1}^{l-1}(1-\alpha_j)t_{j-s}(z)\Phi_j(z) \Bigg).$$

Note that while computing c_{Φ_1} the terms other than the coefficient of $\frac{1}{[1-(1-\alpha_s)t_0(z)]^s}$ will become 0 under the limits $z \to \sigma_k$. Therefore, the coefficient c_{Φ_1} is given by

$$\begin{aligned} c_{\Phi_1} &= \lim_{z\to\sigma_k} (\sigma_k - z)^s \Phi_1(z) \\ &= \lim_{z\to\sigma_k} (\sigma_k - z)^s \frac{(1-\alpha_s)^{s-1}[t_1(z)]^{s-1}}{[1-(1-\alpha_s)t_0(z)]^s} \Bigg(T(z) \\ &\quad - \sum_{j=s}^{l-1}(1-\alpha_j)t_{j-s}(z) + \sum_{j=s+1}^{l-1}(1-\alpha_j)t_{j-s}(z)\Phi_j(z) \Bigg). \end{aligned}$$

Using L'Hôpital's rule s times in the above expression and using the relation $(1-\alpha_s)t_0(\sigma_k) = 1$ we get

$$c_{\Phi_1} = \frac{[t_1(\sigma_k)]^{s-1}}{(1-\alpha_s)[T'(\sigma_k A(0))A(0)]^s} \Bigg(T(\sigma_k) - 1 + \sum_{j=s+1}^{l-1}(1-\alpha_j)t_{j-s}(\sigma_k)\big(\Phi_k(\sigma_k) - 1\big) \Bigg), \tag{3.20}$$

where we also used $t_0(z) = T(zA(0))$. Therefore, from eqn (3.18)

$$\begin{aligned} Y_k(\sigma_k) &= (1-\alpha_1) r_1(\sigma_k) c_{\Phi_1} \\ &= \frac{r_1(\sigma_k)[t_1(\sigma_k)]^{s-1}}{[T'(\sigma_k A(0))A(0)]^s} \Bigg(T(\sigma_k) - 1 \\ &\qquad + \sum_{j=s+1}^{l-1}(1-\alpha_j)t_{j-s}(\sigma_k)\big(\Phi_j(\sigma_k) - 1\big) \Bigg). \end{aligned}$$

Therefore, from theorem 3.2, for large n, we can approximate the tail of Age_ν as

$$P(Age_\nu = n) \sim \sum_{k=1}^{M} Y_k(\sigma_k) \frac{n^{s-1}}{(s-1)!\sigma_k^n}.$$

□

Note that, to compute the tail distribution, we need to compute $r_1(\sigma_k), t_1(\sigma_k), \ldots t_{l-1-s}(\sigma_k), \Phi_{s+1}(\sigma_k), \ldots, \Phi_{l-1}(\sigma_k)$. Using eqn (3.10), one can recursively compute $\Phi_{s+1}(\sigma_k), \ldots, \Phi_{l-1}(\sigma_k)$ starting from $\Phi_{l-1}(\sigma_k)$. Moreover, from eqn (3.3) we have,

$$\begin{aligned} r_1(\sigma_k) &= \left.\frac{\partial S(y,z)}{\partial y}\right|_{\{y=0,z=\sigma_k\}} \\ &= \frac{A'(0)\big(T(\sigma_k A(0)) - T(A(0))\big)}{1 - T(A(0))}. \end{aligned}$$

Similarly, to compute $t_1(\sigma_k), \ldots, t_{l-1-s}(\sigma_k)$ use eqn (3.5), for example,

$$\begin{aligned} t_1(\sigma_k) &= \left.\frac{\partial W(y,z)}{\partial y}\right|_{\{y=0,z=\sigma_k\}} \\ &= T'(\sigma_k A(0))\sigma_k A'(0). \end{aligned}$$

3.5 Backup systems with time trigger mechanism

Some backup systems have a time trigger mechanism to initiate a backup period if the backup-off period becomes too long. Let us assume this timer is of length ψ, then the maximum age of data is given by

$$Age = \min(Age_\nu, \psi). \tag{3.21}$$

Therefore, the age of data for such backup systems can be computed from our analysis using eqn (3.21).

Note: We assume that the timer starts when the first packet arrives during the vacation period. It can be defined from beginning of the off-period or start of the vacation in which the packet arrived. In those cases, eqn (3.21) would have to be modified accordingly. Similarly, we assume that ψ is a constant; however, it can be a stochastic variable.

It is important to note that, for such backup systems to work efficiently, ψ has to be chosen wisely. On one hand, a small value of ψ would result in backup operations being triggered too often. While on the other hand, a large value

of ψ defeats the whole purpose of a time trigger mechanism, which is to avoid long backup off periods. Therefore, it is important for such backup systems to choose an optimal timer length ψ. To measure this efficiency, we define a new performance measure, the frequency of backups.

Frequency of backups

For any general event repeating at a period of P, its frequency is defined as $F = \frac{1}{P}$. Since the backup time itself is typically very short as compared to backup-off periods, we define the frequency of backup operations as

$$Freq = \frac{1}{E(Age)} = \frac{1}{E(\min(Age_\nu, \psi))}. \tag{3.22}$$

To be able to select the optimal trigger parameter, it is important to study the effect of ψ on both the frequency and age of data. The QoS of backup processes is defined in terms of the RPO. That is, a service provider would guarantee an RPO to its user, such as 15min-24hrs guaranteed by [73] for different costs, as mentioned in introduction. Using eqns (3.21) and (3.22), we can compute the moments of Age and the frequency for such backup mechanisms.

$$\begin{aligned}
E(Age) &= E(min(Age_\nu, \psi)) \\
&= \psi \times P(Age_\nu > \psi) + E(Age_\nu | Age_\nu \leq \psi) P(Age_\nu \leq \psi) \\
&= \psi P(Age_\nu > \psi) + \sum_{n=1}^{\psi} n P(Age_\nu = n) \\
&= \psi P(Age_\nu > \psi) + E(Age_\nu) - \sum_{n=\psi+1}^{\infty} n P(Age_\nu = n) \\
&= E(Age_\nu) + \sum_{n=\psi+1}^{\infty} (\psi - n) P(Age_\nu = n).
\end{aligned}$$

$E(Age_\nu)$ can be computed from eqn (3.11). Assuming ψ is reasonably large, using the Theorem 3.3, we can approximate the $E(Age)$ as

$$\begin{aligned}
E(Age) &\approx E(Age_\nu) + \sum_{n=\psi+1}^{\infty} (\psi - n) \sum_{k=1}^{M} Y_k(\sigma_k) \frac{n^{s-1}}{(s-1)! \sigma_k^n} \\
&= E(Age_\nu) + \sum_{k=1}^{M} Y_k(\sigma_k) \sum_{n=\psi+1}^{\infty} (\psi - n) \frac{n^{s-1}}{(s-1)! \sigma_k^n}.
\end{aligned}$$

Using eqn (3.22), the frequency is therefore given by

$$F \approx \left[E(Age_\nu) + \sum_{k=1}^{M} Y_k(\sigma_k) \sum_{n=\psi+1}^{\infty} (\psi - n) \frac{n^{s-1}}{(s-1)! \sigma_k^n} \right]^{-1}. \tag{3.23}$$

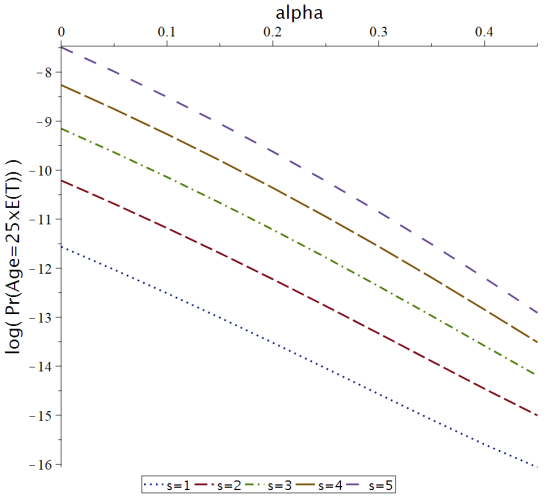


Figure 3.1: Impact of restarting probability α_1 on tail of Age_ν, Arrival rate = 0.1, $l = 20$

3.6 Numerical evaluation

In this section, we analyze the performance of the system using the numerical example used in section 2.5. Since it is well known that the arrival distribution can have a heavy tail (see example [75]) we assume that the number of arrivals in a slot follows a Mixed distribution of Poisson and Power Law. The vacation lengths have a discrete uniform distribution.

$$\begin{aligned} A(z) &= pe^{\lambda(z-1)} + (1-p)\frac{L_\gamma(z)}{L_\gamma(1)}, \qquad p = 0.999, \\ L_\gamma(z) &= \sum_{k=a}^{\infty} \frac{z^\gamma}{k^\gamma}, \qquad a = 25, \qquad \gamma = 2.5, \\ T(z) &= \frac{1}{5}\left(z^{v-2} + z^{v-1} + z^v + z^{v+1} + z^{v+2}\right). \end{aligned} \tag{3.24}$$

Using **Maple**, we find the solutions of the equation (3.14). We find that the generating function $H(z)$ has only 1 dominant singularity and plot $Age_\nu(n) = P(Age_\nu = n)$ with varying system parameters[1].

Using the tail distribution of Age_ν, F and $E(Age)$ can be computed easily using eqn (3.23). Therefore, analyzing the tail distribution of Age_ν is sufficient. To do this analysis with different restarting probabilities, we vary α_s between 0 and 0.5, for values of s between 1 and 5, while keeping the remaining α_i for

[1]The code used for tail distribution analysis is available at `https://github.com/saxe405/tail_distribution`

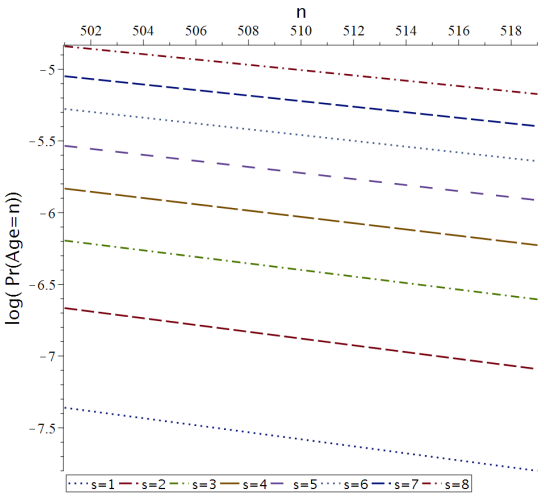


Figure 3.2: Tail of Age_ν with increasing n, Arrival rate $= 0.1$, $l = 20$.

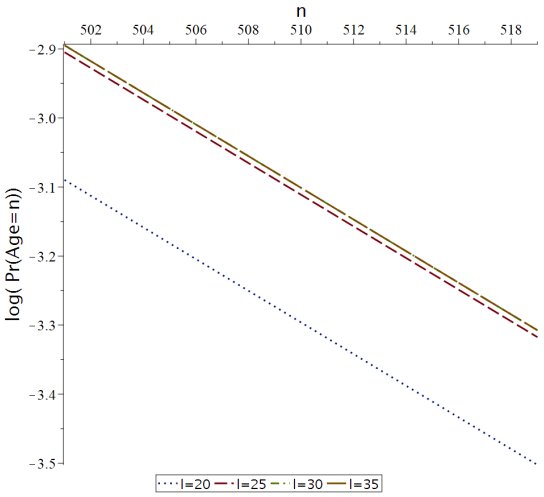


Figure 3.3: Tail of Age_ν with increasing n, Arrival rate $= 0.1$, $s = 1$

$i > s$ fixed. That is,

$$\alpha_1 = \alpha_2 = \ldots = \alpha_s = \alpha,$$
$$\alpha_i = 0.6 + 0.4 \times \frac{i}{l}, \qquad \forall i > s.$$

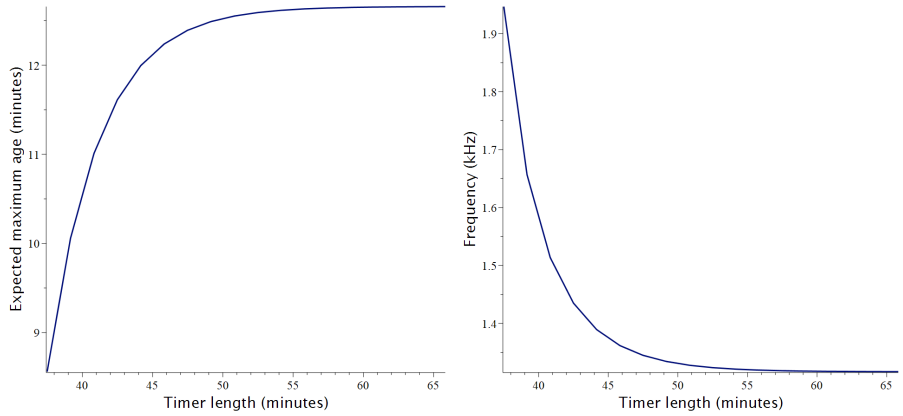


Figure 3.4: Maximum age of data and frequency of backup with change in timer length. $\lambda = 0.07$, slot length = 5 second, $\alpha_i = \frac{i}{30}$, and $l = 20$.

3.6.1 Observations and key insights

A system administrator would desire to keep the age of data as low as possible. However, increasing the frequency of backups would come with an increased economic cost. We are able to exactly compute the generating function of Age_ν as well as the tail distribution. We observe that the tail distribution is sensitive to the change in model parameters.

In the figures 3.1-3.3, we plot the tail probabilities of Age_ν with respect to the change in the model parameters α_1, s and l. In figure 3.4, we compute the expected maximum age, $E(Age)$ and frequency (F) for backup operations with time trigger mechanism.

- Choice of the smallest restarting probability α_1 needs to be smart. A smaller value of α_1 would give a dominant singularity of smaller modulus from eqn (3.14). Since σ_k^n is in the denominator of eqn (3.16), it would lead to larger tail probabilities. This is also observed in Figure 3.1.

- The order of dominant singularities s is directly determined by the choice of restarting probabilities α_j, $0 < j < l$. Since in the eqn (3.16) of tail probability distribution, n^s appears in the numerator, a smaller value of s leads to smaller tail probabilities. This is also observed in Figure 3.1 and 3.2.

- A larger restarting threshold l results in a higher value of $Y_k(\sigma_k)$ from eqn (3.17). Therefore, tail probabilities are higher for higher restarting threshold. This is also observed in Figure 3.3.
- For backup operations with time trigger mechanism, increasing the timer length drastically decreases the backup frequency. As shown in example in Figure 3.4, selecting timer length of 45min instead of 40min reduces the backup frequency by approximately 15%. Users can define their cost functions based on $E(Age), F$ and other performance measures computed in chapter 2. Minimizing this cost function would give the optimal parameters, including the optimal trigger length.

3.7 Summary

In our previous chapter, we modeled data backup systems as a batch service queueing model with vacations and restarting probabilities. We obtained various performance measures such as the moments of the backlog size and the mean value of the age of data at the beginning of a backup. The age of data is one of the most important quantities in backup systems because it captures the risk of data loss in case of a disaster. In this chapter we have computed the generating function of the age of data at the beginning of the backup. As a consequence, we can now compute higher order moments and analyze the tail of this distribution by computing the dominant singularities of the generating function. The choice of restarting probabilities determines the value of these singularities as well as their order. We have analyzed the behavior of the tail of the distribution of Age_ν for different values of model parameters. We are able to precisely compute the tail distribution characterized by the model parameters. We also use this analysis to compute and analyze the performance measures of backup systems with time trigger mechanism.

4

Queueing model with two thresholds

4.1 Introduction

In the previous two chapters, we modeled the backup processes as a queueing model with probabilistic restarting rules. Using that model, we were able to compute the performance measures such as the distribution of backlog size, probability the server is busy. We also computed the PGF of the age of data and analyze its tail distribution. The backup service provider's aim is to ensure that the QoS requirements, with the primary focus on the age of data and backup frequency, are always met. There are multiple ways of ensuring this. Selecting the appropriate restarting parameters or implementing a timer mechanism with an optimal trigger are two possible ways to do this. However, some backup service providers, may still demand a stronger control over the age of data to deliver high QoS. Therefore in this chapter we study a new queueing model for backup server. The key modification is that we explicitly a timer within the model that was missing in Chapter 2.

Performance measure of Queueing model	**Service quality of Data storage**
Queue Size	Backlog size
Probability server busy	Power consumption
Number of new connections	Cost paid to service provider
Number of slots since last backup	Age of the data

Figure 4.1: Relating performance measures of queueing model with QoS of data storage

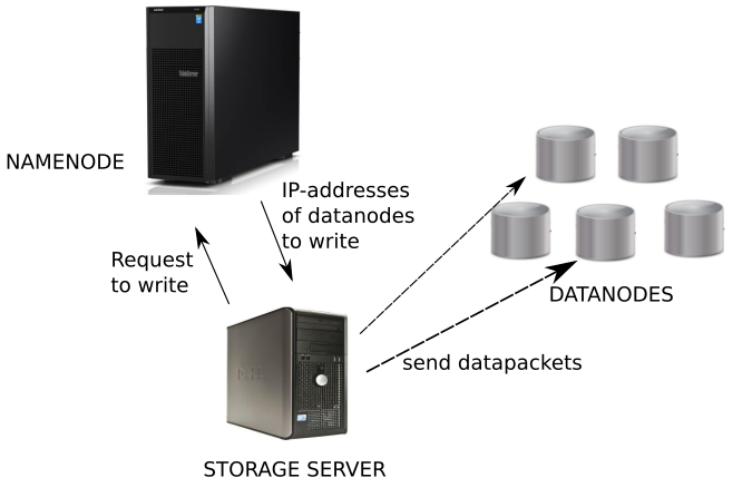


Figure 4.2: Data storage mechanism

We model the storage process as a discrete-time batch service queueing model with sleep mode. This model is characterized using 3 parameters: c, the capacity of the storage server, l, the backlog restarting threshold, and τ restarting threshold for maximum time between the completion of a storage and the start of a new storage. The τ parameter is precisely used to model the timer mechanism. To keep the model applicable to a wide range of use cases, the service time of data packets and sleep epochs of the server are assumed to follow a general distribution. Similar to the model studied in chapter 2, the data which needs to be stored on the cloud infrastructure arrives at the storage server in the form of data packets. We model the number of arrivals of these data packets in a single slot as a general distribution and independent across each slot.

We discussed the challenges faced during the setup of backup operations in chapter 1. For the convenience of the reader, we will briefly reiterate some of those challenges. A system administrator has four major challenges while designing the data storage operations

1. Data backlog should stay low so that the number of data packets waiting for storage in the backlog should not exceed the available system buffer.

2. Waiting time of data packets should stay low as the data packets that are not yet stored can be lost.

3. Storage server should be in sleep mode as long as possible to keep the power consumption and cloud resource usage low.

4. Establish as few connections as possible to cloud infrastructure as the user is charged for number of connections made to namenode.

In order to ensure that challenges (1) & (2) are met, the administrator needs to perform storage operations as frequently as possible. However, increasing the frequency of storage operations is not good for challenges (3) & (4). Therefore, there is a clear trade-off involved in selecting this storage frequency. By modeling these storage operations as a queueing model, we aim to compute the necessary performance measures and this trade-off.

4.2 Model description

Newly generated data packets are not immediately stored, but wait before they are uploaded to the cloud infrastructure. We model newly generated data packets as packets arriving to a queueing system. Waiting before storage is modelled by waiting in the queue of the queueing model until being served by a server. Service corresponds to being stored on cloud infrastructure.

The number of data packets generated during consecutive slots is assumed to constitute an independent and identically distributed sequence $\{A_k; k \geq 1\}$ where A_k denotes the number of newly generated data packets during the slot k. The common random variable of the sequence $\{A_k; k \geq 1\}$ is denoted by A and its mass function and probability generating function respectively by $a(n)$ and $A(z)$. Note that A has a general distribution, i.e., we do not impose a particular distribution to the applicability of the model. This is exactly the same as the model we described in section 2.1.

A common strategy to tackle the trade-off explained in the chapter 2 is to only start the storage when enough packets are present, i.e., waiting for storage; but at the same time, imposing a limit on the maximum number of slots since the last storage. This is modelled as thresholds as follows; when the storage server becomes available it starts a new storage operation if the number of packets present is at least l or if the time since the completion of the previous storage is at least τ.

Once the storage is activated, multiple batch operations are performed till backlog gets empty. Since, for efficiency purpose, packets are not transmitted individually, but in batches of, say, c packets, the storage service is modelled as a series of batch services of c packets (except if less than c packets are present for the batch, then a smaller batch is served). Furthermore, in order to not start storage too frequently, the threshold l is not too small in practice and typically larger than c. Hence, it is safe to assume that $l \geq c$. Further, we assume that $A(0) > 0$, i.e., the probability that zero packets arrive in a slot is non-zero otherwise the storage server would never go into a sleep mode.

Remark: The exhaustive policy is frequently used in storage processes

and cloud service providers charge each new connection (new storage initiation), for example [8] and [11] can charge as high as \$0.01 per 1,000 requests. The number of storage operations in general are significantly higher than the number of total files eventually written. As already explained in section 1.2.1, this is because the packets stored on the cloud infrastructure are immutable. That is, even if there is a small change to a file, which is divided into several data packets for storage, all the data packets and their redundant copies have to be rewritten. Therefore, we incorporate exhaustive service policy in the model of cloud storage. Hence, after having served a batch of packets, the server serves another batch of packets unless there are no packets any more. Packets arriving during storage are thus also served during that same storage cycle. Checking each slot whether a new storage could be initiated would be too energy consuming. Hence, if the server decides not to start the storage, the server temporarily goes into sleep mode. Incorporating sleep mode is a common strategy in telecommunications to save energy for example [76], [77], [78], [79], [80], [81].

In order not to restrict the applicability of the model, the length of a sleep period denoted by T, can have any distribution. The generating function of T is denoted by $T(z)$. Note that the special case $T(z) = z$ corresponds to checking each slot. Hence, the case of no sleep mode is also included. The lengths of distinct sleep epochs are assumed to be independent. This part of the model is also present in the model studied in chapter 2.

The time to transmit (serve) a batch of m packets ($m \leq c$) is denoted by G_m and its probability generating function by $G_m(z)$. Again, to not restrict the applicability, no restrictions are imposed on the distribution of G_m. A summary of all the notations used in the analysis is given in the table 1.

4.3 Analysis

The analysis of this model will follow a similar path as we have seen in section 2.2. However, the analysis of the system initiation opportunities is more involved than seen in section 2.2.4.

We begin by writing down the discrete Markov chain for (Q_k, W_k, R_k). We observe the system at a random slot k and observe the quantities Q_k, W_k and R_k. Using these quantities we can write the following relations for the slot $k + 1$

- If there is more than one slot remaining in the current epoch, the system sees only new arrivals. During sleep periods, the number of slots since previous storage also increases.
- If the system is at the end of service of the current batch and the backlog

Table 1: List of notations used in the analysis	
c	capacity of the batch server
l	starting threshold of backlog size for batch service to restart when server wakes up from a sleep mode.
τ	starting threshold of number of slots since last storage for batch service to restart when server wakes up from a sleep mode.
T	defines the distribution of any random sleep period with the generating function $T(z)$.
Q_k	the number of packets in queue at the slot boundary k i.e. excluding the packets in service.
W_k	the number of slots since the last storage at slot boundary k. $W_k = 0$ implies that server is on during slot k while $W_k = j > 0$ implies that no storage has taken place in j slots (including the current slot).
A_k	the number of arrivals in the slot k. Its distribution is given by the generating function $A(z)$.
R_k	the remaining service time or the remaining sleep time in the current epoch depending on whether a service or sleep mode is going on.
G_m	the service time of a batch containing m packets. Its distribution is given by the generating function $G_m(z)$. Moreover, $G_0 \equiv T$.
$\eta(a, b)$	minimum$\{a, b\}$.
ρ	$A'(1)G_c'(1)/c$ is the load of the system
$I_0(x)$	1 iff $x = 0$ 0 otherwise

is not empty, it starts serving a new batch in the next slot(with max c in a single batch). This is due to the exhaustive service policy. Further, if the backlog is empty, the server goes into a sleep mode.

- If the server is at the end of a sleep epoch and finds that at least τ slots have passed since the last storage, it starts the service immediately. However, if the backlog is empty, it just resets the number of slots since last storage to 1 and goes into another sleep mode.
- If the server is at the end of a sleep epoch and finds that the threshold τ has not been breached, it decides to start a new batch service if the backlog is more than l packets.

Using the relations above, we start by writing the vector $(Q_{k+1}, W_{k+1}, R_{k+1})$ in terms of (Q_k, W_k, R_k).

$(Q_{k+1}, W_{k+1}, R_{k+1}) =$

- If $R_k > 1, W_k = 0$, (ongoing service)

$$(Q_k + A_k, 0, R_k - 1). \tag{4.1}$$

- If $R_k > 1, W_k > 0$, (ongoing sleep mode)

$$(Q_k + A_k, W_k + 1, R_k - 1). \tag{4.2}$$

- If $R_k = 1, W_k = 0$ OR $R_k = 1, W_k \geq \tau$, (τ threshold breached)

$$(Q_k + A_k - \eta(Q_k + A_k, c), I_0(\eta(Q_k + A_k, c)), G_{\eta(Q_k+A_k,c)}). \tag{4.3}$$

- If $R_k = 1, 0 < W_k < \tau, Q_k + A_k \geq l$, ($l$ threshold breached)

$$(Q_k + A_k - c, 0, G_c). \tag{4.4}$$

- If $R_k = 1, 0 < W_k < \tau, Q_k + A_k < l$, (no threshold breached)

$$(Q_k + A_k, W_k + 1, T). \tag{4.5}$$

Define the joint generating function,

$$V_k(z, y, x) = E(z^{Q_k} y^{W_k} x^{R_k - 1}).$$

Further, we use the notation

$$E(z^X\{\text{condition}\}) = E(z^X|\{\text{condition}\})P(\text{condition}).$$

Using relations eqn (4.1-4.5), one can write

$$V_{k+1}(z, y, x) = E\Big(z^{Q_k + A_k} x^{R_k - 2}\{R_k > 1, W_k = 0\}\Big)$$

$$
\begin{aligned}
&+ E\Big(z^{Q_k+A_k} y^{W_k+1} x^{R_k-2}\{R_k > 1, W_k > 0\}\Big) \\
&+ E\Big(z^{Q_k+A_k-\eta(Q_k+A_k,c)} y^{I_0(\eta(Q_k+A_k,c))} x^{G_{\eta(Q_k+A_k,c)}-1} \\
&\qquad \{R_k = 1, W_k = 0 \text{ OR } R_k = 1, W_k \geq \tau\}\Big) \\
&+ E\Big(z^{Q_k+A_k-c} x^{G_c-1}\{R_k = 1, 0 < W_k < \tau, Q_k + A_k \geq l\} \\
&+ z^{Q_k+A_k} y^{W_k+1} x^{G_0-1}\{R_k = 1, 0 < W_k < \tau, Q_k + A_k < l\}\Big).
\end{aligned} \tag{4.6}
$$

Following [68], the system is stable when $\rho < 1$. We look at the steady state of the system. For this, we further define

$$V(z, y, x) = \lim_{k\to\infty} V_k(z, y, x), \tag{4.7}$$

$$d_j(m) = \lim_{k\to\infty} P(Q_k + A_k = m, W_k = j, R_k = 1), \qquad j < \tau \tag{4.8}$$

$$d_\tau(m) = \lim_{k\to\infty} P(Q_k + A_k = m, W_k \geq \tau, R_k = 1). \tag{4.9}$$

Namely, $d_j(m)$ represents the stationary probability of having m packets in the backlog, and the number of slots since last storage being equal to j and the server being at the end of a sleep or storage period. For $j = \tau$ the number of slots since last storage is greater than or equal to τ.

Using eqns (4.7) -(4.9), eqn (4.6) can be rewritten as

$$
\begin{aligned}
V(z, y, x) = {} & \frac{A(z)}{x}\Big(V(z, 0, x) - V(z, 0, 0)\Big) + \frac{A(z)y}{x}\Big(V(z, y, x) \\
& - V(z, y, 0) - V(z, 0, x) + V(z, 0, 0)\Big) \\
& + \frac{yT(x)}{x}\Big(d_0(0) + d_\tau(0)\Big) + \sum_{m=1}^{c-1} \frac{G_m(x)}{x}\Big(d_0(m) + d_\tau(m)\Big) \\
& + \sum_{m=c}^{\infty} \frac{z^{m-c} G_c(x)}{x}\Big(d_0(m) + d_\tau(m)\Big) \\
& + \sum_{j=1}^{\tau-1}\sum_{m=l}^{\infty} \frac{z^{m-c} G_c(x)}{x} d_j(m) + \sum_{m=0}^{l-1}\sum_{j=1}^{\tau-1} \frac{z^m y^{j+1} T(x)}{x} d_j(m).
\end{aligned} \tag{4.10}
$$

One can observe that

$$\lim_{k\to\infty} \sum_{m=0}^{\infty}\sum_{j=0}^{\tau} z^m P(Q_k + A_k = m, W_k = j, R_k = 1) = A(z)V(z, 1, 0). \tag{4.11}$$

Using eqn (4.11), eqn (4.10) can be rewritten as

$$\begin{aligned}V(z,y,x)\Big(1-\frac{A(z)y}{x}\Big) &= \frac{A(z)(1-y)}{x}\Big(V(z,0,x)-V(z,0,0)\Big)\\ &\quad-\frac{A(z)y}{x}V(z,y,0)+\frac{A(z)G_c(x)}{xz^c}V(z,1,0)\\ &\quad+\Big(\frac{yT(x)}{x}-\frac{G_c(x)}{xz^c}\Big)\Big(d_0(0)+d_\tau(0)\Big)\\ &\quad+\sum_{m=1}^{c-1}\frac{z^cG_m(x)-z^mG_c(x)}{xz^c}\Big(d_0(m)+d_\tau(m)\Big)\\ &\quad+\sum_{m=0}^{l-1}\sum_{j=1}^{\tau-1}\frac{z^{m+c}y^{j+1}T(x)-z^mG_c(x)}{xz^c}d_j(m). \quad (4.12)\end{aligned}$$

This equation contains $\tau\times l+c$ unknowns namely $\mathbb{U}=\big\{d_0(m):m=0\ldots c-1\big\}\bigcup\big\{d_j(m):m=0\ldots l-1,j=1\ldots\tau\big\}$. We will show that $V(z,1,0)$, $V(z,0,x)$, $V(z,0,0)$, $V(z,y,0)$ can be expressed as linear combinations of the unknowns $\mathbb{U}$. Therefore, $V(z,y,x)$ can be expressed explicitly using the model parameters and unknowns $\mathbb{U}$.

Substituting $y=1$ followed by $x=A(z)$ in eqn (4.12) we get

$$\begin{aligned}A(z)V(z,1,0)\Big(z^c-G_c(A(z))\Big) &= \Big(z^cT(A(z))-G_c(A(z))\Big)\big(d_0(0)+d_\tau(0)\big)\\ &\quad+\sum_{m=1}^{c-1}\Big(z^cG_m(A(z))-z^mG_c(A(z))\Big)\Big(d_0(m)+d_\tau(m)\Big)\\ &\quad+\sum_{m=0}^{l-1}\sum_{j=1}^{\tau-1}\Big(z^{m+c}T(A(z))-z^mG_c(A(z))\Big)d_j(m). \quad (4.13)\end{aligned}$$

Substituting $y=0$ in eqn (4.12) gives

$$\begin{aligned}V(z,0,x)(x-A(z)) &= -A(z)V(z,0,0)-\frac{G_c(x)}{z^c}\Big(d_0(0)+d_\tau(0)\Big)\\ &\quad+\sum_{m=1}^{c-1}\Big(G_m(x)-\frac{z^mG_c(x)}{z^c}\Big)\Big(d_0(m)+d_\tau(m)\Big)\\ &\quad+A(z)V(z,1,0)\frac{G_c(x)}{z^c}-\frac{G_c(x)}{z^c}\sum_{j=1}^{\tau-1}\sum_{m=0}^{l-1}z^md_j(m). \quad (4.14)\end{aligned}$$

Further, substituting $x=A(z)$ in eqn (4.14) gives us

$$A(z)V(z,0,0)=-\frac{G_c(A(z))}{z^c}\Big(d_0(0)+d_\tau(0)\Big)+A(z)V(z,1,0)\frac{G_c(A(z))}{z^c}$$

$$+\sum_{m=1}^{c-1}\Big(G_m(A(z))-\frac{z^mG_c(A(z))}{z^c}\Big)\Big(d_0(m)+d_\tau(m)\Big)$$
$$-\frac{G_c(A(z))}{z^c}\sum_{j=1}^{\tau-1}\sum_{m=0}^{l-1}z^md_j(m). \tag{4.15}$$

Finally, $V(z,y,0)$ can be computed by substituting $x=A(z)y$ in (4.12) which gives us

$$\begin{aligned}yA(z)V(z,y,0)&=A(z)(1-y)\Big(V(z,0,yA(z))-V(z,0,0)\Big)\\&+\Big(yT(yA(z))-\frac{G_c(yA(z))}{z^c}\Big)\Big(d_0(0)+d_\tau(0)\Big)\\&+\sum_{m=1}^{c-1}\Big(G_m(yA(z))-\frac{z^mG_c(yA(z))}{z^c}\Big)\Big(d_0(m)+d_\tau(m)\Big)\\&+\frac{A(z)G_c(yA(z))}{z^c}V(z,1,0)\\&+\sum_{m=0}^{l-1}\sum_{j=1}^{\tau-1}\Big(z^my^{j+1}T(yA(z))-\frac{z^mG_c(yA(z))}{z^c}\Big)d_j(m),\end{aligned}$$

which on using eqn (4.13) - (4.15) gives us

$$\begin{aligned}yA(z)V(z,y,0)&=yA(z)V(z,0,0)+yT(yA(z))\Big(d_0(0)+d_\tau(0)\Big)\\&+\sum_{m=0}^{l-1}\sum_{j=1}^{\tau-1}z^my^{j+1}T(yA(z))d_j(m).\end{aligned} \tag{4.16}$$

Substituting eqn (4.13)-(4.16) in eqn (4.12), one can write

$$\begin{aligned}&V(z,y,x)(x-yA(z))\\&=y\big(T(x)-T(yA(z))\big)\big(d_0(0)+d_\tau(0)\big)+\Big[\frac{x-yA(z)}{x-A(z)}\Big]\times\\&\Big[-z^{-c}\Big(G_c(x)-G_c(A(z))\Big)\Big(d_0(0)+d_\tau(0)\Big)+\sum_{m=1}^{c-1}\Big(G_m(x)\\&-G_m(A(z))-z^{m-c}\big(G_c(x)-G_c(A(z))\big)\Big)\Big(d_0(m)+d_\tau(m)\Big)\\&-\Big(G_c(x)-G_c(A(z))\Big)\sum_{j=1}^{\tau-1}\sum_{m=0}^{l-1}z^{m-c}d_j(m)\Big]\\&+\frac{\big(x-yA(z)\big)A(z)\big(G_c(x)-G_c(A(z))\big)}{(x-A(z))z^c}V(z,1,0)\end{aligned}$$

$$+\sum_{m=0}^{l-1}\sum_{j=1}^{\tau-1} z^m y^{j+1}\Big(T(x)-T(yA(z))\Big)d_j(m), \tag{4.17}$$

where $V(z,1,0)$ is given by eqn (4.13). Therefore, we have shown that the generating function $V(z,y,x)$ can be expressed as a linear combination of the unknowns $\mathbb{U}$ and the model parameters. Now, we will compute relations between unknowns $\mathbb{U}$ and form a solvable system of linear equations. Once these unknowns have been computed, the generating function would be completely expressible using the model parameters. To achieve this we use the results from [69].

4.3.1 Zeros of $z^c - G_c(A(z))$

In chapter 2, section 2.2.3, we proved that $z^c - G_c(A(z))$ has $c-1$ zeros inside the complex region $\{z : |z| \leq 1, z \neq 1, z \in \mathbb{C}\}$. These zeros can be used in eqn (4.13) giving us $c-1$ linear equations in the unknowns $\mathbb{U}$.

4.3.2 Normalization condition

The normalization condition of probability generating functions, as already seen in section 2.2.6, gives us $V(1,1,1) = 1$. Therefore, by applying L'hôpital rule in eqn (4.17) and using the normalization condition we get

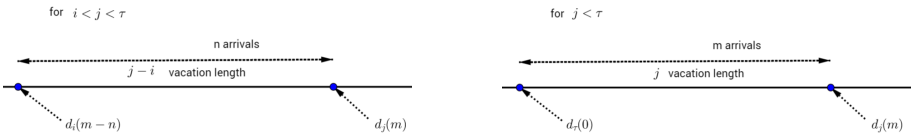
$$cT'(1)\Big(d_0(0)+d_\tau(0)\Big) + \sum_{m=1}^{c-1}\Big(cG'_m(1) - mG'_c(1)\Big)\Big(d_0(m)+d_\tau(m)\Big) + \sum_{j=1}^{\tau-1}\sum_{m=0}^{l-1} cT'(1)d_j(m) = c - G'_c(1)A'(1). \tag{4.18}$$

Therefore, using the zeros of $z^c - G_c(A(z))$ the normalization condition, eqn (4.18), we already have c linear equations in unknowns $\mathbb{U}$. However, the number of unknowns in $\mathbb{U}$ is equal to $\tau \times l + c$. To find more relations, we look at service initiation opportunities.

4.3.3 Service initiation opportunities

Service initiation opportunities are defined as the slot boundaries on which the service/sleep periods start and end. We look at a service initiation opportunity at the end of a sleep period and relate it to the previous service initiation opportunity. We looked at service initiation opportunities for the model studied in chapter 2 in section 2.2.4. In that model, the restart of service was dependent only on the number of packets in the backlog. However, here it also depends on the number of slots since the last backup which gives us more cases to consider.

Note that if the service is not initiated at the end of a sleep period, another sleep period would follow. Our main aim is to make some observations



(a) Previous epoch $d_i(m), i < j$ (b) Previous epoch $d_\tau(0)$

Figure 4.3: Recursive relation for Case I

and write recursive relations between $d_j(m), j > 0$. These relations can be divided into two cases.

CASE I: $j < \tau$

During the steady state, let us assume that the storage server is at the end of the sleep mode and slot number $= k$ i.e. $R_{k-1} = 1$. Further assume that the backlog size $Q_k = m$, and the number of slots since the last storage $W_{k-1} = j$. The probability that the server is in this state is equal to $d_j(m)$ where $j < \tau$. Say the previous period ended at slot f, it has to be one of the following :

1. a service period with no backlog at the end i.e. $Q_f = 0, W_{f-1} = 0, R_{f-1} = 1$. The probability the server is in this state is equal to $d_0(0)$,
2. a sleep period at the end of which more than τ slots had elapsed without storage and no backlog i.e. $Q_f = 0, W_{f-1} \geq \tau, R_{f-1} = 1$. The probability of the server being in this state is equal to $d_\tau(0)$,
3. a sleep period with less than j slots since last storage i.e. $Q_f = n, W_{f-1} = i, R_{f-1} = 1$ where $i < j, n \leq m$.

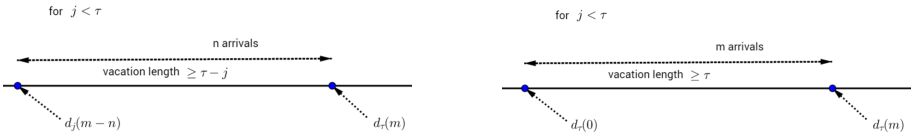
See Figure 4.3 for an illustration of the recursive relation. It is important to note that at the end of the previous service initiation opportunity, the backlog can not be more than l packets otherwise the current period would not be a sleep period. Therefore for $m < l$, we can write

$$d_j(m) = \sum_{i=1}^{j-1} \sum_{n=0}^{m} d_i(n) a(m-n, j-i) t(j-i) + \big(d_0(0) + d_\tau(0)\big) a(m, j) t(j), \tag{4.19}$$

where $a(n, s)$ is the probability of n arrivals in s slots and $t(s)$ is the probability that the length of sleep epoch is s slots.

CASE II: $j = \tau$

During the steady state, let us assume that the storage server is at the end of the sleep mode and the slot number is equal to k , i.e., $R_{k-1} = 1$. Further assume that the backlog size Q_k is equal to m and the number of slots since



(a) Previous epoch $d_j(m), j < \tau$ (b) Previous epoch $d_\tau(0)$

Figure 4.4: Recursive relation for Case II

the last storage W_{k-1} is equal to τ. The probability that the server is in this state is equal to $d_\tau(m)$. Say the previous period ended at slot f, it has to be one of the following :

1. a sleep period with j slots since the last storage where $j < \tau$, i.e., $Q_f = n, W_{f-1} = j, R_{f-1} = 1$. The probability that the server is in this state is equal to $d_j(n)$,

2. a sleep period with more than τ slots since the last storage and no backlog i.e. $Q_f = 0$, $W_{f-1} \geq \tau, R_{f-1} = 1$. The probability that the server is in this state is equal to $d_\tau(0)$,

3. a service period which ends with zero packets in backlog, i.e., $Q_f = 0$, $W_{f-1} = 0, R_{f-1} = 1$. The probability that the server is in this state is equal to $d_0(0)$.

See Figure 4.4 for an illustration of the recursive relations. Therefore for $m < l$, we have

$$d_\tau(m) = \sum_{j=1}^{\tau-1} \sum_{n=0}^{m} \sum_{s=\tau-j}^{\infty} d_j(n)a(m-n,s)t(s) + \sum_{s=\tau}^{\infty} \big(d_\tau(0) + d_0(0)\big)a(m,s)t(s). \tag{4.20}$$

Together, these two cases give us $\tau \times l$ new equations. Combining the results obtained in section 4.3.1 and section 4.3.2, we have a solvable linear system of equations with $(c + \tau \times l)$ unknowns. Therefore, by solving this linear system, one can exactly compute the generating function $V(z, y, x)$ in terms of the model parameters.

Computational complexity

For exact computation of the generating function $V(z, y, x)$, one needs to solve a system of linear equations with $(c+\tau \times l)$ unknowns for a general distributions of A, T and G_m, $m \leq c$. During the analysis of the model studied in chapter 2, we only had to solve for $l+c-1$ unknowns, see section 2.2.3. In contrast, here we have to solve for a large number of unknowns which increases the computational complexity of this model. However, since T, the distribution of the sleep epoch

of the storage server, is fixed by the system administrator, it should have a reasonably simple distribution. This reduces the order of the system of linear equations significantly. That is, if we know that $P(T > p) = 1$, the length of each sleep epoch is more than p. Therefore, from eqn (4.19), $d_j(m) = 0$ for all $j \leq p$. The order of system of linear equations is reduced to $c + (\tau - p) \times l$.

4.4 Performance measures

In this section, we look at generating functions of some performance measures of this queueing system which are expressed in $V(z, y, x)$. These include the QoS measures of the data backup policy described in section 1.2.1. Since we have computed a closed form expression for this generating function in terms of the model parameters and known constants $\mathbb{U}$, the expressions of performance measures can be easily computed by substitution. Further, it is very straightforward to compute the moments of the performance measures from their generating functions. For example, if $X(z)$ is a pgf of a random variable X,

$$\begin{aligned} E(X) &= \left.\frac{\partial X(z)}{\partial z}\right|_{z=1}, \\ Var(X) &= \left.\frac{\partial^2 X(z)}{\partial z^2}\right|_{z=1} + \left.\frac{\partial X(z)}{\partial z}\right|_{z=1} - \left(\left.\frac{\partial X(z)}{\partial z}\right|_{z=1}\right)^2, \\ E(X^n) &= \left.\frac{\partial^n X(z)}{\partial z^n}\right|_{z=1}. \end{aligned}$$

4.4.1 Backlog size

By the definition of the generating function $V(z, y, x)$, the generating function of the backlog size is given by $V(z, 1, 1)$.

4.4.2 Queue content during sleep mode

When the system is in sleep mode, $W_k > 0$. Therefore, the generating function for the queue content during sleep mode is given by

$$\frac{V(z, 1, 1) - V(z, 0, 1)}{1 - V(1, 0, 1)}. \tag{4.21}$$

4.4.3 Probability of a new connection

A new connection is established when a new service is initiated after a sleep epoch. Therefore,

$$P(\text{ New connection }) = \lim_{k \to \infty} P(W_{k+1} = 0, W_k > 0)$$

$$
\begin{aligned}
&= \lim_{k\to\infty} \sum_{j=1}^{\infty} Pr\big(W_{k+1}=0, W_k=j, R_k=1\big) \\
&= \lim_{k\to\infty} \sum_{j=1}^{\tau-1} \sum_{m=l}^{\infty} Pr\big(Q_k+A_k=m, W_k=j, W_{k+1}=0, R_k=1\big) \\
&\quad + \lim_{k\to\infty} \sum_{m=1}^{\infty} Pr\big(Q_k+A_k=m, W_k\geq\tau, W_{k+1}=0, R_k=1\big) \\
&= \sum_{m=l}^{\infty} \sum_{j=1}^{\tau-1} d_j(m) + \sum_{m=1}^{\infty} d_\tau(m) \\
&= V(1,1,0) - V(1,0,0) - \sum_{m=0}^{l-1} \sum_{j=1}^{\tau-1} d_j(m) - d_\tau(0) \\
&= d_0(0).
\end{aligned}
$$

Note that $d_0(0) = \lim_{k\to\infty} P(Q_k + A_k = 0, W_k = 0, R_k = 1)$. That $d_0(0)$ is equals the probability that we are in the last slot of a storage period. Between two consecutive storage periods, there is always a backup-off period. Therefore, in steady state the P(new connection) equals the probability that we are in the last slot of a storage period.

4.4.4 Probability that server is busy

During service periods, the number of slots since the last storage remains zero. Therefore the probability that the server is busy is given by

$$
\begin{aligned}
P(\text{ Server Busy }) &= \lim_{k\to\infty} P(W_k=0) = V(1,0,1) \\
&= \left(\frac{G_c'(1)T'(1)A'(1)}{c-G_c'(1)A'(1)}\right)\big(d_0(0)+d_\tau(0)\big) \\
&+ \sum_{m=1}^{c-1} \left(G_m'(1) + \frac{G_c'(1)\big(G_m'(1)A'(1)-m\big)}{c-G_c'(1)A'(1)}\right)\big(d_0(m)+d_\tau(m)\big) \\
&+ \sum_{m=0}^{l-1} \sum_{j=1}^{\tau-1} \left(\frac{G_c'(1)T'(1)A'(1)}{c-G_c'(1)A'(1)}\right) d_j(m). \qquad (4.22)
\end{aligned}
$$

This quantity is the probability that a in randomly selected slot, the backup server is busy. A higher value of this probability would imply that the server is active during more slots. Therefore, it captures the power consumption of the storage server. Moreover, to keep the backup operations running, the backup server communicates actively with the cloud infrastructure, this quantity also captures the duration for which connections to the cloud infrastructure are kept alive.

4.4.5 Number of slots since last storage

By the definition of the generating function $V(z, y, x)$, the generating function of the number of slots since the last storage is given by

$$\begin{aligned} V(1,y,1) = &\left(\frac{y(1-T(y))}{1-y} + \frac{G_c'(1)T'(1)A'(1)}{c - G_c'(1)A'(1)}\right)\big(d_0(0) + d_\tau(0)\big) \\ &+ \sum_{m=1}^{c-1}\left(G_m'(1) + \frac{G_c'(1)\big(G_m'(1)A'(1) - m\big)}{c - G_c'(1)A'(1)}\right)\big(d_0(m) + d_\tau(m)\big) \\ &+ \sum_{m=0}^{l-1}\sum_{j=1}^{\tau-1}\left(\frac{y^{j+1}(1-T(y))}{1-y} + \frac{G_c'(1)T'(1)A'(1)}{c - G_c'(1)A'(1)}\right)d_j(m). \end{aligned} \tag{4.23}$$

This is an important quantity since it gives us the distribution of how long the data storage operations would be stopped. As described in chapter 1, this gives us the distribution of the limits defined by RPO. In chapters 2 and 3, we were able to compute only the first moment and the PGF for an approximate measure of this quantity.

4.5 Numerical example

In this section, we consider a storage system whose parameters are inspired by our experience with storage systems and some proprietary data. This example is very similar to the one studied in section 2.5. To evaluate the performance, we evaluate the first moments of the measures defined above.

Storage system

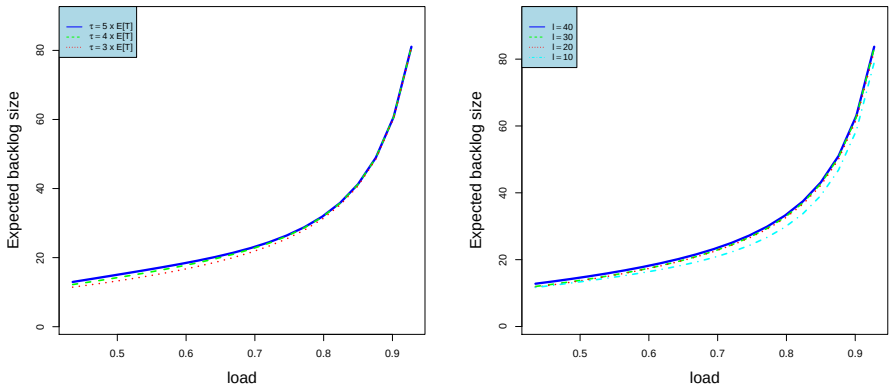
The newly generated data packets arrive at the storage server at a rate between 20 and 60 MB/min. The storage server can serve at a maximum rate of 60 MB/min. Data packets are served in batches of maximum size of 10 packets. The storage server enters sleep mode if there is nothing to serve at the end where the length of sleep mode is 1.5 minutes on average. The storage server resumes operations after the sleep if there are more than 20 data packets waiting in backlog or more than 5 mins have elapsed since the past storage operation. Each data packet is, on an average, 10 MB is size.

Queueing model

We assume that each slot is of 10 seconds length. Therefore, model parameters can be evaluated as

- The distribution of number of arrivals in a single slot is a mix of Poisson and Power law so that the distribution has a heavy tail i.e.

$$A(z) = p \times e^{\lambda(z-1)} + (1-p) \times \frac{\zeta_\gamma(z)}{\zeta_\gamma(1)},$$



(a) Dependence on τ (b) Dependence on l

Figure 4.5: Expected backlog size with increasing system load

$$\zeta_\gamma(z) = \sum_{k=r}^{\infty} \frac{z^k}{k^\gamma}.$$

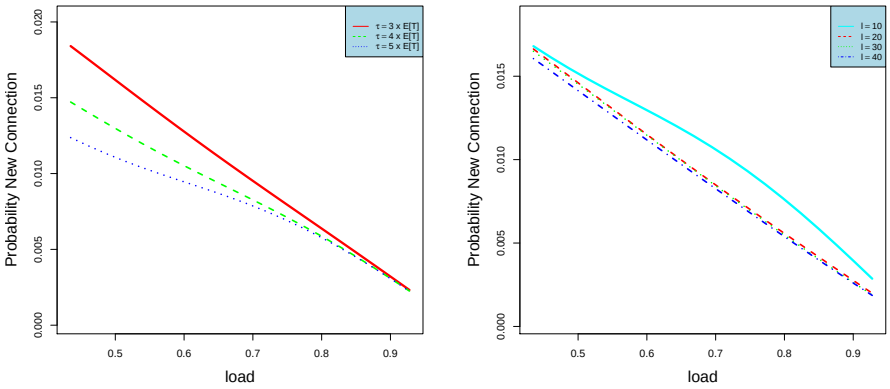
- The length of each sleep period has a uniform distribution between 7 and 11 slots, i.e. $T(z) = \frac{1}{5}\sum_{i=7}^{11} z^i$ and $E(T) = 9$.
- The batch server capacity $c = 10$.
- The backlog starting threshold $l = 20$.
- The number of slots since last storage starting threshold $\tau = 30$.
- Service time of a batch of size m has a uniform distribution i.e. $G_m(z) \equiv [m, m+2]$.

In our numerical analysis we show the dependence of the first moment of our performance measures on different values of l and τ for varying system load. These results are illustrated in Figures 4.5 - 4.8.

4.5.1 Observations

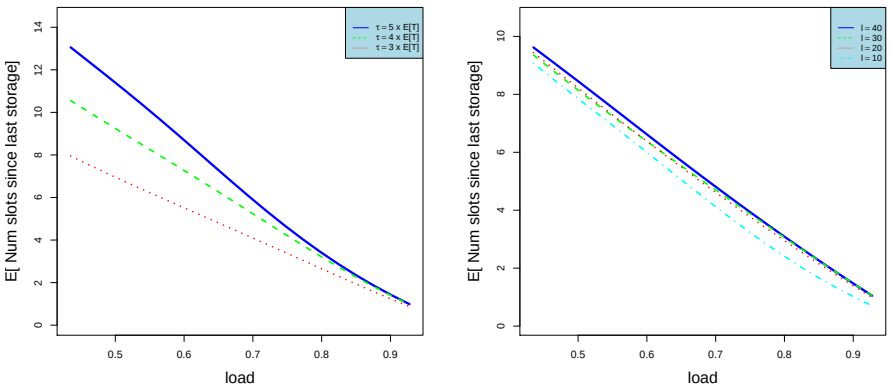
The results illustrated in Figure 4.5 - 4.8 show that the performance measures are dependent on the choice of storage operations parameters. When the system is under low or moderate load, we observe the following

1. from Figures 4.5-4.7, smaller restarting thresholds are better to have a lower expected backlog size. While higher restarting thresholds result in smaller probability of a new connection and longer time since the last storage.



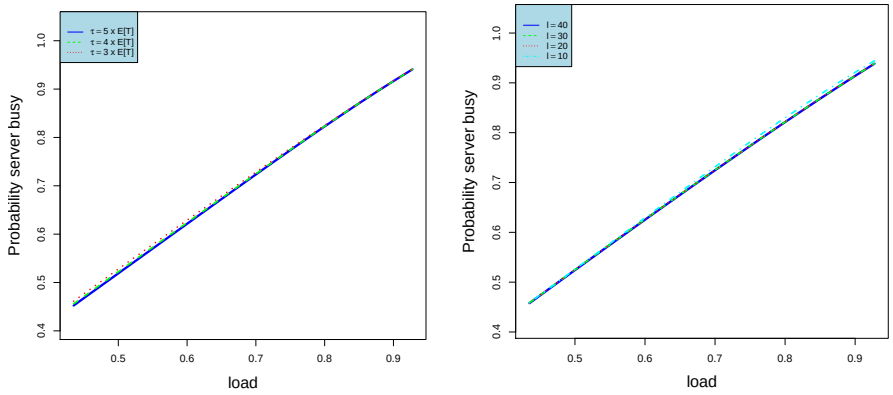
(a) Dependence on τ (b) Dependence on l

Figure 4.6: Probability of new connection with increasing system load



(a) Dependence on τ (b) Dependence on l

Figure 4.7: Expected number of slots since last upload with increasing system load



(a) Dependence on τ (b) Dependence on l

Figure 4.8: Probability server busy with increasing system load

2. from Figure 4.8, a higher thresholds results in lower probability of server being busy. It is important to note that even a slight change in this measure translates to equivalent percentage change in the consumption of power and cloud resources. Therefore, even an improvement of 0.1% may be substantial.

When the load reaches the capacity of the system, the choice of parameters does not have a significant impact for any of the defined performance measures. This happens because the system would rarely go into sleep mode when the load is very high and therefore the restarting thresholds do not play a large role.

4.5.2 Comparison with model of chapters 2 and 3

In our model in chapters 2 and 3 we have modeled backup processes as a queueing model with probabilistic restarting rules. In that model, the batch server goes into a vacation if there is nothing to serve. When the vacation ends, it checks *only* the number of packets in the backlog. If there are more than l packets in the backlog, the service is resumed immediately. However, if there are i packets in the backlog, $i < l$, the backup service is resumed with probability α_i where the α_i s are defined by the user.

In contrast to those chapters, our focus here is to model the time between the completion of storage and restart of next storage. By modeling this time, we model the recovery point objective (RPO) limits. Moreover, in chapters 2 and 3, using the model and analysis, we computed the mean value and PGF of the distribution of a measure which is an approximation of the RPO limits. However, in this chapter we are able to compute the exact distribution of these RPO limits. Finally, we include a timer such that if the RPO is larger than a

value τ, we start a new storage even if the number of packets is lower than l.

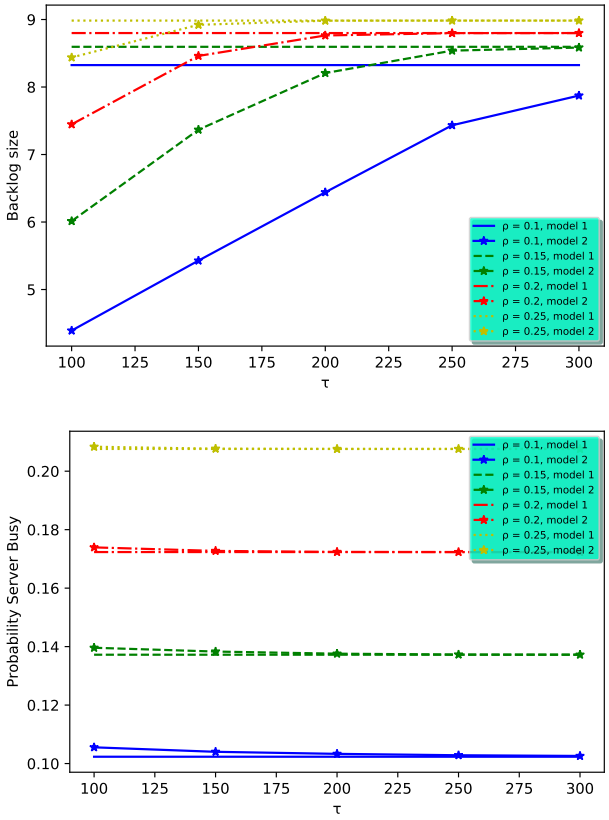
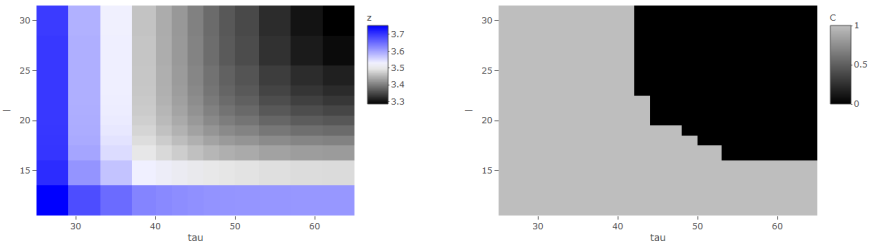


Figure 4.9: Illustration that as τ increases, the performance measures of our model (model 2) converges to results obtained in [1] (model 1).

The queueing model studied in this chapter and the model studied in chapters 2 and 3 are equivalent if we have $\alpha_i = 0, \forall i < l$ and $\tau \to \infty$. That is, with no probabilistic restarts the model described in chapter 2 is equivalent to this model without a timer ($\tau = \infty$). In figure 4.9, we plot the mean backlog size and the probability server is busy at random slot with increasing value of τ. It illustrates that as τ increases, the performance measures converge to the performance measures give by model in chapter 2 with $\alpha_i = 0, \forall i < l$. The convergence is faster for higher loads since the server goes into fewer vacations for higher loads. The difference between the performance measure values for the two models highlights the importance of correctly modeling τ, especially for the mean backlog size. That is, by introducing τ in our model, we are able to more accurately capture the behavior for storage services with a strict limit on time since the last storage operation.



(a) Cost function (b) Feasible region of l and τ shaded grey

Figure 4.10: Cost function and feasible region

4.5.3 Cost minimization

In this section we define the cost function of a user which captures the monetary cost of doing storage operations similar to what we did in section 2.5.3. The expected QoS as defined as constraints. We then look for the optimal values of parameters l and τ which minimize the cost. We use the same parameters as defined in the beginning of this section with load $\rho = 0.6$. Define the cost function as

$$C(l, \tau) = e^{\text{Prob server busy}} + e^{50 \times \text{Prob New Service}}. \tag{4.24}$$

Further, the QoS measures are given by the following constraints

$$\begin{aligned} E(\text{Backlog Size}) &\leq 22 \text{ packets}, \\ E(\text{Number of slots since last storage}) &\leq 8 \text{ slots}. \end{aligned} \tag{4.25}$$

View Figure 4.10 for the change in the cost function, eqn (4.24), of the user and the feasible region given by the constraints, eqn (4.25). We find that $\tau = 47$ and $l = 19$ are the optimal thresholds for this user.

4.5.4 Final insights

While designing data storage processes, one is mainly concerned about the challenges described in the introduction. These challenges can be divided into two broad categories i.e. cost of performing storage and the probability of data loss. The level of sensitivity of the measures in these categories, on the storage parameters is very different. Given a feasible target of the performance measures, one can compute the required storage parameters of the storage system. From our observation and analysis in the previous sections, we observe the following

1. Expected backlog size and probability that the server is busy are numerically less sensitive to the storage parameters as compared to probability of a new connection and expected number of slots since last storage.

However, for a user, a small change in less numerically sensitive measures (say probability that server is busy) may be more costly than a larger change in more numerically sensitive measure(say expected number of slots since last storage).

2. These parameters have an impact only when the system is under moderate or low load.

3. A trade-off can be achieved between the counteracting performance measures by selecting optimal storage parameters.

4.6 Summary

In this chapter we have modeled cloud data storage processes as an exhaustive batch service queueing model with sleep mode and thresholds. By properly selecting the threshold of backlog size and time since the last storage, we can ensure a good balance between risk of data loss and cost of doing storage operations. We compute some key performance measures of the queueing system which help us to numerically compute the performance of an equivalent storage system. We consider a numerical example where we illustrate the dependence of these performance measures on the storage parameters. These computations are useful to address the trade-off between doing data storage frequently enough and reducing the resource usage. We then illustrate the use of these performance measures to write a cost function of a user with additional QoS constraints. The cost function and constraints are then used to select the optimal restarting threshold parameters.

5

Virtual Reality headsets

In this chapter, we analyze the streaming of 360^0 videos on VR headsets using a wireless channel. Current capabilities of wireless communication under 4G do not support enough bandwidth to be able to handle wireless streaming on VR headsets. Therefore, the streaming services are dependent on a wired Ethernet connection. The same cable is used to supply the power to the headset (see Figure 5.1). However, with the growth of 5G wireless standards, see [82], it will be possible to provide the services without a wired connection. Wireless transmission is desired for two reasons:

- Being connected to a wire results in a degraded immersive experience as it limits the free movement.
- The wires can also be dangerous as one can easily trip on them as their vision is completely occupied by the headsets.

Streaming completely wireless comes with a couple of challenges. First, wireless communication is less reliable; 5G has more intermittent behavior due to shadowing etc. Secondly, in the absence of any cable, the headset would have to be powered by a battery. Therefore, the consumption of battery power, which is linked to the number of times a wireless antenna is used, has to be minimal. To tackle these challenges, we need to maintain a buffer of the

Figure 5.1: Oculus rift

correct tiles by pre-fetching when the wireless channel quality is good. When the channel quality is bad, this buffer can mitigate the impact and maintain the high standards of QoE of the users.

In normal videos, when video needs to be buffered, the complete view of the video is downloaded in the best possible resolution permitted by the available bandwidth. Therefore, the buffer length is simply the length of video in the buffer which can be played back. However, in 360^0 videos only the portion of the video that is in the view of the user is transmitted. This portion of the video is called the Field of View (FoV) and accounts for less than 15% of the total video (see Figure 5.2). Such transmission reduces the bandwidth requirements which makes the high quality FoV transmission possible.

The buffering mechanism for 360^0 videos is therefore much more challenging than for a simple video. Not only do we need to account for the network interruptions, we also need to take into account the user head movements to build a useful buffer. Additionally, if the user head moves in an unanticipated direction, the buffer needs to be rebuilt. Since the device is supported by a battery and a wireless network connection, we need to ensure that both these resources are used reasonably.

Providing such an immersive experience comes with a lot of challenges. Since the video outside the FoV is transmitted at the lowest quality, there is a risk that if the user moves outside the anticipated FoV, a low quality fallback video is displayed. Many papers have tried to optimize the different components of streaming 360^0 videos. For example, [83] aim to minimize the average transmission rate by addressing the trade-off between creating projections at the VR headsets versus doing it at the edge node. [84] look at a similar trade-off with delay constraints by utilizing computation and buffering resources at the VR device. [85] optimize the buffer use by storing video in multiple bit rates.

A few papers have analysed the transmission of 360^0 videos on the headset. [4] use linear regression to predict and download the FoV of the user up to two seconds and download tiles of this predicted FoV. [86] implement a methodology where the tiles in the FoV of user are streamed at the highest possible quality and at the lowest for tiles outside this view with buffer length limited to two seconds. [87] maintain an additional low quality buffer of the whole 360^0 video to avoid scenario in which nothing is shown to the user. [5] investigate a multi-tier approach of streaming on VR headsets, where the impact of bad network conditions is mitigated by falling back on a WiFi connection. All the studies use simulations to analyze the performance and compare with other strategies. While simulation studies may have some advantages, performing them can be a time consuming exercise. On the other hand, our model is able to provide a precise table of download decisions the headset must take,

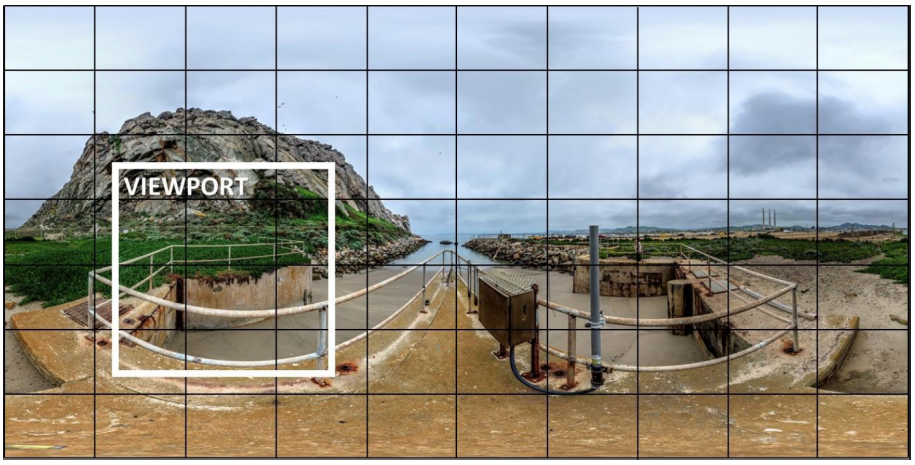


Figure 5.2: The FoV of a user, see [2] for more details.

encompassing all the system scenarios. Moreover, this model also guarantees that the long run cost of performing the task would be minimal. More precisely we use Markov Decision Process (MDP) to get the decision table and then use simulation to evaluate the performance under different conditions.

In this chapter, we use the patterns of head movement of the user and the information about the most viewed parts of the video to build the buffer which has FoV in the highest quality. We model this as an MDP which helps us decide which parts of the video should be buffered. For example, if enough high quality tiles are not available for the upcoming video segment, the headset attempts to use as much bandwidth that it is permitted to use in the given channel quality. However, if the upcoming segments of FoV are available in high quality, tiles are downloaded only if the channel quality is very good, as downloading in such conditions uses fewer resources. More precisely, downloading decisions are taken to build a buffer in a way such that the trade-off between high QoE and resource usage is optimized. We show how the buffering mechanism translates to MDP parameters. The MDP approach leads to a downloading policy that ensures a proper trade-off between QoE and resource usage.

Our contribution is therefore:

- We develop an MDP based streaming approach which includes user head movement prediction as well as variation in the channel quality,
- The mechanism allows the flexibility to model different video types and users to accurately measure and optimize the cost and QoE of any specific user,

- We perform experiments to collect and analyse the user head movement data and use it to evaluate the MDP based buffering policy,
- We show that a higher quality of FoV can be achieved using our approach at a lower expected cost.
- Unlike other methods, using the MDP model, the buffer decisions can be computed in a few minutes.

We start with an overview of streaming of videos in Section 5.1 where we provide the required details of streaming process on VR headsets. We discuss the modelling assumptions in Section 5.2. This is followed by an overview of MDP formulation of the problem in Section 5.3 where we show how we model the primary features of VR streaming of 360^0 videos. In Section 5.4, we give a brief overview of the experiment in which we collect the data of user head movement. In the Sections 5.5 - 5.7 we evaluate the performance of this buffering approach by comparing it with other common approaches.

5.1 Video streaming on VR headsets

360^0 videos are divided into small tiles which can be downloaded separately and then stitched together after the transmission. Since only the FoV tiles are primarily required, those tiles are rendered in the highest resolution possible. The tiles outside the FoV are transmitted at the lowest resolution as a fall back mechanism. Therefore, any buffering mechanism needs to predict and account for the FoV of the user to build an appropriate buffer. However, if the user makes an unanticipated movement, this buffer needs to be rebuilt.

Video is transmitted in short segments of a fraction of second. That is, if a normal video is buffered, the transmission is done to add short segments of the video to build the buffer. Similarly, in 360^0 videos, the video is split in short segments in time. Therefore, when a tile is downloaded a short segment of that portion of video is downloaded. In this chapter we assume that the segments are half a second long.

Buffering mechanism is case of 360^0 videos is challenging, because we need to predict the FoV of the user. If incorrect tiles are downloaded, it leads to higher costs without any improvement to QoE. When buffering, we have two types of information available about the user interest: short predictions and long predictions.

5.1.1 Short term prediction

At any point in time, we have access to the head position of the user and the angular momentum of their head in three independent directions (see Figure 5.3). Using this information, it is possible to predict the head position

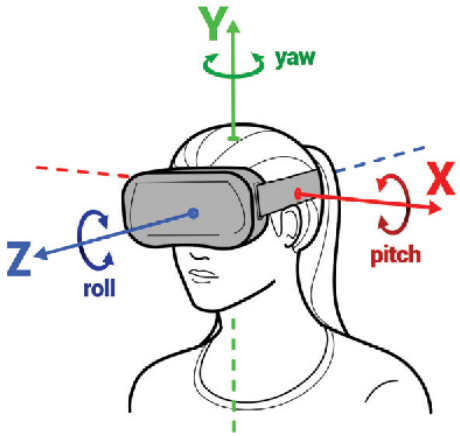


Figure 5.3: 360^0 video viewing direction (from [3])

in the next one or two seconds with very high accuracy. [88] show that accurate predictions are possible up to half or one seconds with accuracy of more than 92%. More methods exist for such predictions. For example, [89] describe a gravitational predictor and use of AI-techniques to predict the interesting part of the video. [90] use supervised learning to predict the eye position.

In our chapter, we assume these predictions are represented as three-dimensional distribution function and are known beforehand. That is, $P(\theta_r, \theta_p, \theta_y)$ is the probability that the head moves in a time slot by θ_r, θ_p and θ_y in the roll, pitch and yaw respectively (see Figure 5.3). Note that these predictions are user dependent and are only accurate for one or two seconds. Therefore, we can only use them to build the buffer of the first few segments. Moreover, these predictions are independent of the current head position and momentum.

In Figure 5.4, we illustrate these distributions that we get from analyzing the data of our experiments. We have two different videos viewed by 27 participants (more details to follow in Section 5.4). We illustrate the distribution of the degree of head movement in each direction for different cases. We have picked up three special cases to highlight the extreme cases of head movement that we observed for these users. We will use these scenarios to evaluate the performance of our buffering mechanism in the upcoming sections.

5.1.2 Long term prediction

If we observe different users over the same video segments, there is a decent amount of overlap in their viewing patterns (see Figure 5.5). That is, different

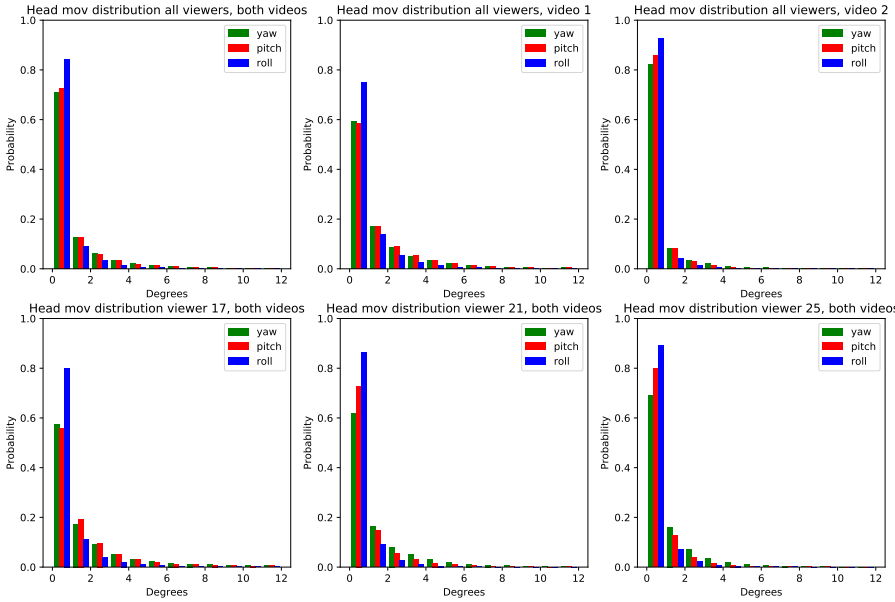


Figure 5.4: Probability distributions of the user head movement from the experiment detailed in Section 5.4

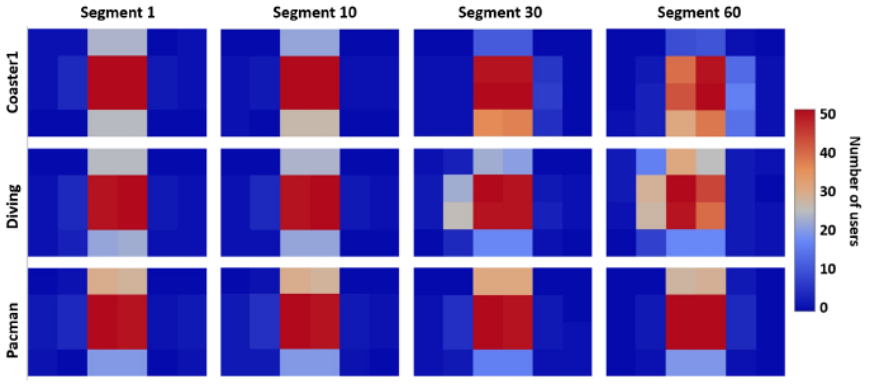


Figure 5.5: Heatmap of user's FoV pattern (from [4])

users find similar areas of the videos interesting and their FoV is in that general direction. [91] provide a data-set of head movements of viewers watching 360^0 videos. [92] analyze the traces of viewing data of more than 150 users and conclude that the viewer's focus follows a similar pattern over all users using the data shared in [93]. [94] use traces of head movement of users from 19 VR videos to predict the future FoV position of the user using deep learning.

5.1.3 Tile importance

For each segment, using short or long term prediction, for each tile we know the probability of it being in the FoV. The magnitude of these probabilities defines the sequence in which tiles are downloaded. In other words, these probabilities define the importance of each tile to create the user FoV. Therefore, to know the status of a future segment in the buffer, we only need to know the number of tiles downloaded in the buffer for that segment. We do not need to know the location of those tiles within the segment. When a user moves the head, the importance of tiles changes, which makes only a fraction of tiles in the buffer useful.

5.2 Modelling assumptions

In the following sections, we will model the buffering process as a discrete-time Markov decision process. That is, time is divided into slots (of a fraction of a second) and the decisions to buffer are taking at the beginning of a slot. We assume that the channel quality (CQ) is a Markov process, described by probability transition matrix P_{CQ}. It is assumed that the channel quality remains constant during a single video segment. The aim is to perform buffer operations at the beginning of these slots when the channel quality is good, such that the FoV is available for future segments in a high quality. The algorithm optimizes a weighted average of QoE and resource usage.

We will make the following assumptions:

- The whole 360^0 video is always available in the lowest quality. We do not consider this transmission as a part of our problem as it doesn't require much bandwidth.
- Tiles of the best quality are transmitted to improve the quality of the FoV. Our focus is on the transmission of these high quality tiles.
- The user behaviour is known from the beginning, i.e., the distribution of the navigation pattern $P(\theta_r, \theta_p, \theta_y)$ is known.

We divide the buffer into two parts (see figure 5.6)

- The short-term buffer which is limited to the first two segments

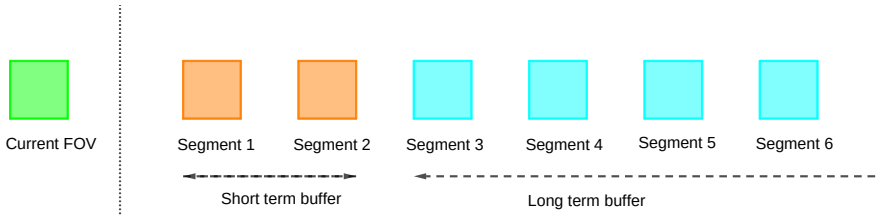


Figure 5.6: Buffer divided into short term and long term buffer

- The long-term buffer which comprises all the segments after the short term buffer.

The head movements only influence the first two segments and are always given the higher priority. Moreover, we enforce that for long-term buffer segments, no more than M tiles are downloaded per segment. This is done to ensure that only reasonable number of tiles are downloaded using long term predictions as they are not user specific.

5.3 MDP formulation for buffering mechanism

A Markov decision process has four major components: the state space, the probability transition matrix, the action space and the reward function. To capture the buffering mechanism, we need to define the situation of the buffer and channel in the form of a state. This state should include the information of the number of tiles in the buffer, the time left in the current segment being played as well the condition of the channel quality. Let us denote the state space as $\mathbb{S}$, the current slot number is denoted by k and the system is in state s_1 at the starting of this slot.

At the beginning of each slot, a buffer decision can be chosen out of all possible actions denoted by action set $\mathbb{A}$. The validity of these actions depends on the state of the system. Let us assume that the action chosen is denoted by a at the beginning of slot k.

The system transitions between states as new tiles are downloaded and channel quality changes. This transition is also dependent on the action a chosen at the beginning of the slot. The matrix $P(s_2|s_1, a)$ defines the probability that the system enters state s_2, *given* that it was in state s_1 and the action a was chosen.

The system is rewarded for being in any given state, where the states which ensure that the QoE is good have a higher reward. This influences the decision of the buffering process to move the system closer to the high reward states. This reward function is defined as $R(a, s_1)$, that is, $R(a, s_1)$ is the

amount of reward received if an action a is chosen if the system is in state s_1. Precisely, it is a weighted sum of the quality of the predicted FoV defined by s_1 and the direct cost of performing the action a.

As the system transitions between these states and takes actions, it collects rewards. Therefore, for any given state, we can define the expected utility for each state of a given policy π as:

$$V^{\pi}(s) = R(\pi(s), s) + \sum_{s'} \gamma P(s'|\pi(s), s) V^{\pi}(s')$$

where γ is the discount factor.

Therefore, using the framework of the Markov decision processes (MDPs), we want to arrive at the map of actions which would lead to the maximum expected total reward of the system. This is because these set of actions will influence the system to move to states with higher reward which are representative of high QoE and low expected cost. Such a map of action space is called the *optimal policy*. We will use π^* to define such an optimal policy where

$$\pi^*(s) = a^* \tag{5.1}$$

for each state s it gives the optimal action a^*.

5.3.1 State space

The state space is defined by

$$\begin{aligned} \mathbb{S} = \{&(n_1, n_2, L, N, R, CQ) \\ &0 \leq n_1 \leq T, 0 \leq n_2 \leq T, 0 \leq L \leq L_{max}, \\ &0 \leq N \leq M, 1 \leq R \leq Q, 0 \leq CQ \leq CQ_{max}\}, \end{aligned} \tag{5.2}$$

where n_1 is the number of tiles in the first buffer segment, n_2 is the number of tiles in the second buffer segment, L is the number of long-term segments which have already been downloaded and N is the number of tiles in the current long term segment which needs more tiles. R is the number of slots left in the current video segment being played at the headset and CQ is the channel quality during the slot. At the end of the video segment in the headset, the tiles in the first buffer segment are used to build the FoV of the user. By storing only the number of tiles in each segment and not including their position, we are able to ensure that the state space is not too large which results in a good performance of policy iteration discussed in paragraph 5.3.5.

5.3.2 Action space

For any given state of the system, there is a set of feasible actions from which one must be chosen. The set of all such possible actions forms the action space

$\mathbb{A}$. We define actions using a tuple (bs, n), where bs is the buffer segment in which the tiles are downloaded and n is the number of tiles downloaded. For the short term buffer, $bs = 1$ or 2, while for the long term buffer, $bs = 3$. That is,

$$\mathbb{A} = \{(0,0)\} \cup \{(bs, n) : bs = 1, \ldots 3, n = 1, \ldots T\}. \tag{5.3}$$

Not all actions may be valid in a given state. For example, if the channel quality $CQ = 0$, no tiles can be downloaded. Therefore, any action (bs, n) with $bs > 0$ is not valid when $CQ = 0$.

5.3.3 Probability transition matrix

The system transitions between states based on the following factors:

- The action chosen in the previous slot: for instance, if more tiles are downloaded in the second segment, system would transition to a state with more tiles in that segment.
- User head movements: for instance, if the user moves their head a few degrees more than anticipated, some of the tiles in the buffer would become useless. That is, out of n_1 and n_2 tiles in the buffer, only a fraction of them would be useful. Therefore, the system would move to a state with smaller n_1 and n_2 depending on the magnitude of movement.
- Channel quality: the channel quality, alone, is assumed to follow a Markov process. This transition is determined by the probability transition matrix P_{CQ}.
- Remaining time in the current segment of the video: when the current segment of headset is over, the first segment of the buffer is used to create the FoV of the user. Which means that in the next slot, n_1 is replaced by n_2 and n_2 is replaced by M or N, depending on the condition of long term buffer segments.

5.3.4 Reward function

In a given state s, if action a is chosen, it would lead to a reward of $R(a, s)$. This reward is a weighted sum of two quantities:

- *Cost of downloading*: downloading tiles in a channel state incurs a cost which increases as the channel quality degrades. Therefore, the algorithm is penalized higher if tiles are downloaded in a worse channel quality.
- *FoV quality*: a state defines the quality of the predicted FoVs. Therefore, the algorithm is penalised if the buffer of the predicted FoVs is not good. Moreover, the first buffer segment is prioritized over the second buffer segment which has a higher priority over long term segments. Therefore, the penalty for FoV unavailability decreases with higher segment number.

Notation List	
s_1, s_2, s	used to denote states
$\mathbb{S}$	the state space
a	used to denote actions
$\mathbb{A}$	the action space
n_1, n_2	used to denote the number of tiles in segment one and two
T	maximum number of tiles that can be downloaded in a segment
CQ	the channel quality
CQ_{max}	highest channel quality state possible
P_{CQ}	probability transition matrix of channel quality
N	number of tiles in the current long-term buffer segment
M	maximum number of tiles that can be downloaded in any long-term segment
R	remaining number of slots in the current video segment
Q	length of each video segment
L	number of long-term segments downloaded in buffer
L_{max}	maximum number of long-term segments that can be downloaded in buffer
(bs, n)	action which means n tiles are download in segment bs
$R(a, s)$	reward gained by taking an action a in state s
$P(s_2\|a, s_1)$	probability that their is a state change to s_2 given that action a is taken while system is in state s_1
γ	discount factor

By choosing such penalties/rewards we are able to prioritise the tiles in the short-term segments. Moreover, these downloads also take into account the cost of downloading in a given channel quality.

5.3.5 Policy iteration

Policy iteration consists of two sequential phases, policy evaluation and policy improvement [95]. In policy evaluation, we start with a random policy π and evaluate the policy using the Belmann equation. We have already discussed this in detail in section 1.5.4. The optimal policy π^* we get after completion of policy evaluation, optimises a weighted sum of costs and QoE.

5.4 User head movement experiment

We collected a data set of user behaviour and navigation patterns while viewing immersive media. We used the four sequences in the 8i voxelized full bodies data set [96] for the experiment. Each sequence was clipped at 5 seconds long and was rendered at a constant 30 frames per second. The playback was looped until the participant asked to move ahead to the next sequence. The sequences were rendered in the Unity game engine and participants were asked to wear an Oculus Rift CV1 head mounted display to view the content. They were asked to sit on a swivel chair placed at the centre of the room and were allowed to inspect the scene using only head movements. We rotated and scaled the sequences so they appeared in front of the participants and were speaking or moving in their general direction. At every rendered frame we logged the position and orientation of the viewport by tracking the camera object in the scene. We capture the viewport position using X,Y,Z coordinates in the world coordinate system and the orientation using three Euler angles.

5.5 Algorithm parameters

To view the parameters and the exact code, have a look at the github repository[1] .

FoV tile distribution

As expected, downloading more tiles improves the quality of FoV of the user. The model can handle any general cumulative probability distribution, $F(x)$, of the utility of tiles. Here we will use a uniform distribution for utility, i.e., $F(x) = Cx$, where C is a constant.

[1]The code is available at `https://github.com/saxe405/buffer`

Head movements

We use the distributions obtained from the analysis of head movement discussed in section 5.4. We have selected six particular scenarios, namely:

- *W3_ all* : distribution using all the data.
- *W3_ V1* : distribution using the data of video one.
- *W3_ V2* : distribution using the data of video two.
- *W3_ P17* : distribution using the data of viewer 17.
- *W3_ P21* : distribution using the data of viewer 21.
- *W3_ P25* : distribution using the data of viewer 25.

Channel quality

The authors of [5] use 5G traces to measure the performance of a multi-tier approach of streaming on VR headsets. These traces suggest that using three states Markov process would be enough to capture the variation in the channel quality. The model can however support any general channel quality matrix.

Long term predictions

Our model requires only the parameter M to be defined for long term predictions. A high value of M may result in wastage of resources, while using a very small value defeats the purpose of using long term predictions. We use $M = 80\%$ of the maximum number of tiles we need to complete a FoV.

5.6 Performance measures

We compute two main performance measures:

- *FoV quality/QoE*: as high quality tiles are downloaded, the quality of FoV improves. This measure quantifies the quality of the FoV by looking at the buffer at the end of the video segment being played on the headset.

$$\text{FoV quality} = \frac{1}{K}\sum_{k=1}^{K} F(s_k.n_1 \big| \text{segment ends at slot } k), \tag{5.4}$$

  where s_k is the state of the system at the end of slot k and $F(.)$ is the FoV tile distribution defined in section 5.5 and K is the length of the simulation.

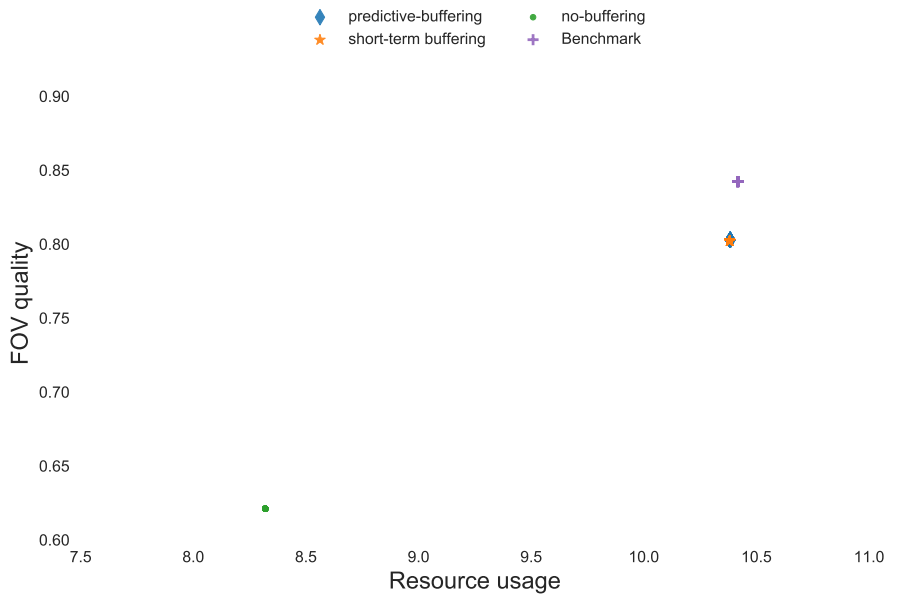


Figure 5.7: Scatter plot of QoE vs cost for different weights in the reward function. There is only a single point for streaming because it doesn't use the reward function.

- *Cost of streaming*: Cost of downloading new tiles depends on the channel conditions. The idea is to improve the FoV quality at the least additional cost. This measure captures the cost of streaming using any policy.

$$\text{Cost} = \frac{1}{K}\sum_{k=1}^{K} \text{Num\_tiles\_downloaded}(\pi(s_k)) * Cost(s_k.CQ), \quad (5.5)$$

  where π is the chosen policy which gives us the map of actions. In case of predictive buffering, we use the optimal policy obtained by policy iteration defined in section 5.3. s_k the state of the system at the end of slot k and $Cost(CQ)$ is the cost of downloading a single tile in the channel quality CQ.

To numerically analyze the performance, we compare three streaming policies[1] and a benchmark

- *No-buffering*: In this methodology, the headset does not plan any buffer segments. While the current segment is played out, the tiles are downloaded for just the upcoming segment. This strategy is the greedy approach which does not consider the variation in the channel quality and the cost of downloading in a bad channel state. This strategy serves as a good benchmark to evaluate the impact of including channel conditions and user behavior into the streaming strategy on the VR headset.

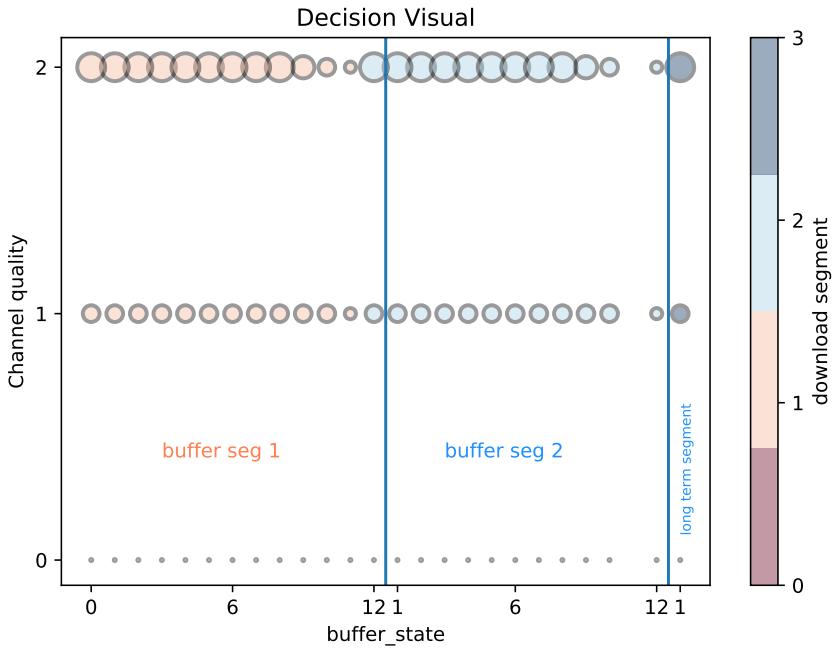
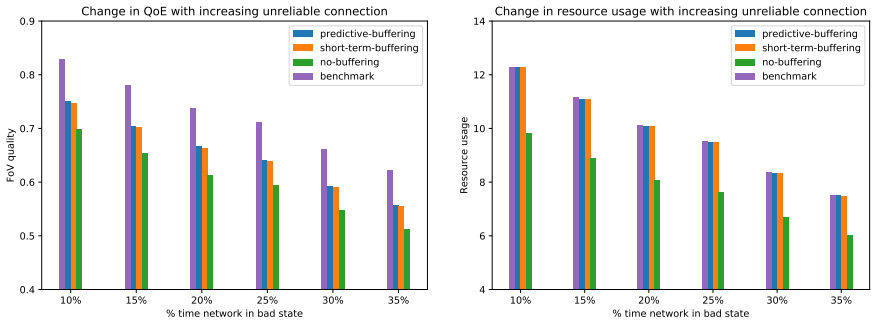


Figure 5.8: Illustration of decisions taken by predictive-buffering in different states. State variables R and L are excluded to reduce the number of dimensions.

- *Predictive-buffering*: this is the approach which uses the optimal MDP policy that optimizes the weighted sum of resource usage and QoE. Both short-term and long-term segments are maintained in the buffer with short-term segments receiving a higher priority over long-term segments.
- *Short-term buffering*: In this methodology, only a short-term buffer is maintained. That is, only the head movement statistics are utilized to download the tiles. This approach is basically identical to the previous approach (predictive buffering) with $L_{max} = 0$ as no tiles are downloaded for long-term segments.
- *Benchmark* : To compute the impact of network interruptions alone, we need to isolate the impact of the prediction error of the head movement on the performance. Therefore, we introduce a benchmark scenario in which we assume that the head movement prediction has no error. More precisely, this metric is equivalent to *Predictive-buffering* with 100% accurate head movement prediction. By comparing this metric with others, we can see the impact of user head movement. Moreover, we are able to compare the performance of *Predictive-buffering* with the best performance under the same networking conditions.



(a) Change in QoE

(b) Change in cost

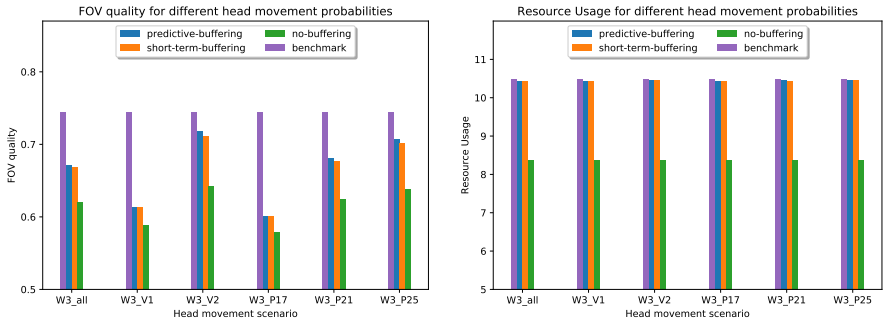
Figure 5.9: Impact of channel unreliability i.e. the amount of time spent in state 0.

5.7 Numerical results

In this section, we briefly discuss the performance of the three streaming policies described in the previous section.

- In Figure 5.7, we show scatter plot of resource usage and quality of FoV. Since there is no buffering mechanism in place for 'no-buffering' scenario, we have a single point. However, for 'short-term buffering' and 'predictive-buffering', we can have different weights for cost of downloading and FoV quality. Therefore, we have multiple points for these scenarios, each defined by a different relative weight of cost of downloading and FoV quality. Using predictive-buffering or short term buffering, we get an improvement of FoV quality of more than 15%. The difference in the FoV quality from the benchmark is less than 5%. Since more high quality tiles are downloaded to maintain a buffer, the improvement in FoV quality is possible at an additional cost. Further, by using both long term and short term predictions, the quality of FoV is further improved without a noticeable increase in the average resource usage.

- In Figure 5.8 we provide an illustration of the decision taken by the predictive-buffering algorithm under different system conditions. The size of the bubble is defined by the number of tiles downloaded, while the colour of the bubble is defined by the segment in which these tiles are downloaded. Clearly, in a bad state (state 0), nothing is downloaded as the channel conditions do not allow it. In the average state (state 1), tiles in the first segment are prioritized. If the FoV of the first segment is complete, tiles of the second segment are downloaded. If both these segments do not need any more tiles, the network capacity is used to download tiles of long-term buffer segments. This holds in the best networking state (state 2) as well, however more tiles can be downloaded in a single slot.



(a) FoV Quality (b) Cost

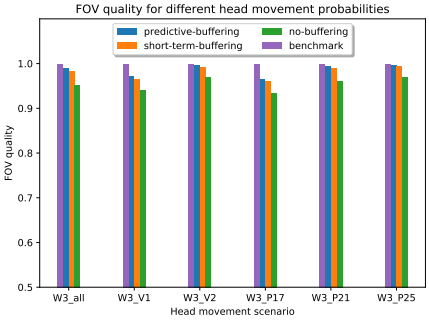
Figure 5.10: Performance for different head movement scenarios where the channel quality is modelled as a Markov chain which spends 20% of the time in state 0.

- In Figure 5.9a we illustrate the change in the quality of FoV as the network becomes increasing unreliable. By increasing unreliability, we mean that the fraction of time spent in bad state increases. In these illustrations, the channel stays in state 2 for 20% of the time and in state 1 for the remaining time. Using predictive-buffering, the performance is slightly better than using only short-term buffering. However, there is significant improvement in the quality as compared to the scenario of no-buffering. Moreover, there is about 5% difference from the benchmark, which is a result of the prediction error in the user head movement. This order of difference in the performance remains consistent with increasing unreliability of the channel.

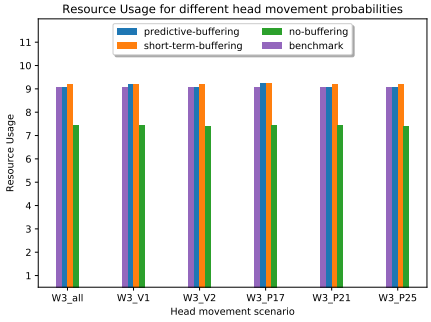
- In Figure 5.9b we illustrate the change in the resource usage as the network becomes increasingly unreliable. There is an additional cost of using predictive-buffering or short-term buffering as compared to no-buffering because more tiles are downloaded in those strategies. Since both the benchmark and predictive-buffering utilize the extra bandwidth to download tiles in the long-term buffer segments, they have a similar resource usage. Moreover, including long term predictions doesn't lead to any increment in the cost. That is, the difference between predictive-buffering and short-term buffering cost is negligible.

- In Figure 5.10 we look at the performance of our algorithm with different probability distributions of head movement. These distributions were computed from the experiment explained in section 5.4. We select these six scenarios since they capture the whole range of expected user behavior. In these figures, we assume that the channel spends 20% of the time in bad states. As illustrated in the figure, the performance is determined by how accurately we are able to predict the user FoV. For cases where the user has a higher tendency to move their head position, W3_V1 and W3_P17,

Performance along trace 2: a stable wireless 5G connection

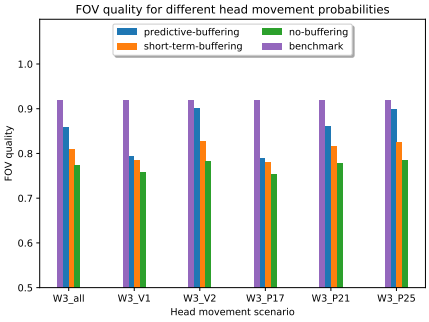


(a) FoV Quality

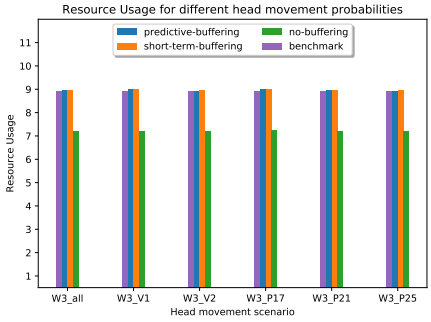


(b) Cost

Performance along trace 5: an unstable wireless 5G connection



(c) FoV Quality



(d) Cost

Figure 5.11: Performance along 5G traces published in [5]. The Markov model of channel quality is used to compute the decision table while the 5G trace is used in the simulation to evaluate the performance.

the performance is far from the benchmark. There is an improvement in performance with W3_all and W3_P21 since user head movement is more stable. And for W3_V2 and W3_P25, the performance is very close to the benchmark. It is important to note that there is always an advantage of buffering over no-buffering due to the improvement in the FoV quality. Moreover, since all the bandwidth is utilized to download tiles, the resource usage across all scenarios remains the same.

- In Figure 5.11, we look at the same head movement scenarios. The difference is that instead of using the Markov model for channel quality to simulate the performance, we use the actual traces of 5G published in [5]. Note that, we still use the Markov model of channel quality to solve the MDP using policy iteration. The trace number 2, used in Figures 5.11a-5.11b, corresponds to the trace of 5G with few network interruptions. While the trace number 5, used in Figures 5.11c and 5.11d, corresponds to a trace captured in the worst network condition in their area. Since these figures use the real world data of the head movement prediction as well as the channel quality, we can say with further conviction that using predictive-buffering, a higher FoV quality/QoE, closer to the benchmark is achieved at a small additional cost. Moreover, by including long term predictions we are able to reduce the average resource usage. This happens because the tiles are pre-fetched when networking is cheaper and therefore less tiles have to be downloaded when networking is expensive.

These results can be summarized as

1. Including short-term predictions leads to a substantial increase in the QoE of the user. This improvement is possible at an additional cost of resource usage.

2. There is a further improvement in the QoE by including long-term predictions in the model. Although the incremental improvement may not be as substantial as what we could achieve by using short-term predictions, this improvement comes without any significant additional cost. Further, in some scenarios, by including long-term predictions we are even able to reduce the average long term resource usage (Figure 5.11b).

3. Improvement in the accuracy of the head movement prediction can further help improve the QoE experience as we can get closer to the benchmark.

5.8 Summary

In this chapter, we have modeled the delivery of 360^0 videos on a VR headset over a wireless connection as an MDP problem. We perform an experiment where we show two videos to twenty seven participants to study the user behavior. Using the analysis of this data, we build a head movement prediction model and include it in the MDP formulation of the delivery of 360^0 videos. Further,

we model the intermittent behavior of the wireless channel as a Markov chain where each state represents the quality of the channel. We then solve this MDP using policy iteration which gives us the optimal policy that guarantees a high QoE for a reasonable resource usage by using their weighted average. Further, we illustrate the performance of our approach using traces of 5G connections as well as the head movement behaviour of some users.

6

Conclusion

In this dissertation, we have analyzed the performance of cloud storage processes using stochastic models. We also developed an MDP based streaming algorithm for 360^0 videos on VR headsets. In both these topics, we first identified the key features of these processes and the uncertainties associated with them. Using this information, we create stochastic models to gain more insight into their performance. The analysis of these models also helps in a cost effective and QoS aware scheduling of operations. We now conclude this dissertation with a summary of the main contributions.

In Chapter 2 we model data backup processes as a general queueing model. The backup server which performs the storage operations is modeled as a batch server with limited capacity. Additionally, this backup server goes into vacations when there are no data packets to serve. The service is restarted at the end of the vacation if the probabilistic restarting conditions are met. By analyzing this model, we are able to compute the QoS measures of a general backup process, namely the backlog size, age of data, probability of a new connection, and probability the backup server is busy. We further illustrate the importance of these QoS measures using a case study. In this study, we compute the optimal backup parameters of a storage system with a defined cost and QoS constraints. This computation is possible by framing this as an optimization problem with storage parameters as the variables. It is important to note that this model supports any general distribution of the number of arrivals in a slot as well as the distribution of service time. We therefore use a heavy tailed distribution of number of arrivals in the case study.

In Chapter 3 we focus on the analysis of the age of data of the model studied in Chapter 2. The age of data has a phase type distribution, therefore we can use some existing results of probability theory, Wald's equation, to compute the first moment. However, computing higher moments is not straightforward using such results. We are able to use some properties of the data backup process to exactly compute the PGF of the age of data. We use these results to compute the tail probabilities of the age of data and its

sensitivity on the data backup parameters. These results help in selecting appropriate storage parameters to ensure the tail probabilities are within the limits desired by the operator. Moreover, often data backup providers enforce a trigger mechanism to ensure the limit on the age of data. We compute the age of data as well as the backup frequency for such systems. We illustrate the use of these results to compute an optimal timer for such backup systems.

In Chapter 4 we introduce a new way of modeling storage systems. Some storage service providers have very high QoS guarantees and ensuring low age of data may be critical for delivering such service. This is enforced by including a threshold on the time since last backup within the model. We analyze this new model to get insights into the performance of these backup systems. We compare this model with the model studied in Chapters 2 and 3 and also illustrate the use of these performance measures using a case study similar to one we had in Chapter 2.

In Chapter 5, we introduce a new research topic, streaming of 360^0 videos on VR headsets. We have modeled the delivery of 360^0 videos on a VR headset over a wireless connection as an MDP problem. We perform an experiment where we show two videos to twenty seven participants to study the user behavior. Using the analysis of this data, we build a head movement prediction model and include it in the MDP formulation of the delivery of 360^0 videos. Further, we model the intermittent behavior of the wireless channel as a Markov chain where each state represents the quality of the channel. We then solve this MDP using policy iteration which gives us the optimal policy that guarantees a high QoE for a reasonable resource usage by using their weighted average. Further, we illustrate the performance of our approach using traces of 5G connections as well as the head movement behaviour of some real use cases selected from our experiment.

Bibliography

[1] A. Saxena, D. Claeys, H. Bruneel, B. Zhang, J. Walraevens, Modeling data backups as a batch-service queue with vacations and exhaustive policy, Computer Communications 128 (2018) 46 – 59.

[2] Tiledmedia, https://www.tiledmedia.com/index.php/technology/ (2019).

[3] F. Qian, B. Han, L. Ji, V. Gopalakrishnan, Optimizing 360 video delivery over cellular networks, in: Proceedings of the 5th Workshop on All Things Cellular, Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM, Association for Computing Machinery, 2016, pp. 1–6, 5th Workshop on All Things Cellular: Operations, Applications and Challenges, ATC 2016 ; Conference date: 03-10-2016 Through 07-10-2016. doi:10.1145/2980055.2980056.

[4] A. Mahzari, A. Taghavi Nasrabadi, A. Samiei, R. Prakash, Fov-aware edge caching for adaptive 360 video streaming, in: Proceedings of the 26th ACM International Conference on Multimedia, MM '18, ACM, New York, NY, USA, 2018, pp. 173–181.

[5] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, D. Dai, Multi-path multi-tier 360-degree video streaming in 5g networks, in: Proceedings of the 9th ACM Multimedia Systems Conference, MMSys '18, ACM, New York, NY, USA, 2018, pp. 162–173.

[6] Amazon, Amazon EC2 launched, https://aws.amazon.com/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2---beta/ (2006).

[7] S. Kumar, S. Manvi, G. Shyam, Resource management for infrastructure as a service (iaas) in cloud computing: A survey, Journal of Network and Computer Applications 41 (2014) 424 – 440.

[8] Amazon, Amazon web service s3, https://aws.amazon.com/s3/ (2017).

[9] Amazon, Aws posts $10b in q4 sales, dominates public cloud, https://www.sdxcentral.com/articles/news/aws-posts-10b-in-q4-sales-dominates-public-cloud/2020/01/ (2020).

[10] Microsoft, Microsoft azure storage, https://azure.microsoft.com/ (2017).

[11] IBM, Ibm cloud, https://www.ibm.com/cloud/ (2018).

[12] Amazon, $528.4 billion cloud services market: Global opportunities & strategies to 2022, https://www.businesswire.com/news/home/20190411005524/en/528.4-Billion-Cloud-Services-Market-Global-Opportunities (2019).

[13] Microsoft, SLA for Site Recovery, https://azure.microsoft.com/en-us/support/legal/sla/site-recovery/v1_0/ (2018).

[14] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems-The International Journal of Escience 25 (6) (2009) 599–616.

[15] Y. Zhang, J. Yao, H. Guan, Intelligent cloud resource management with deep reinforcement learning, IEEE Cloud Computing 4 (6) (2017) 60–69.

[16] K. Gai, M. Qiu, H. Zhao, Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing, Journal of Parallel and Distributed Computing 111 (2018) 126 – 135. doi:https://doi.org/10.1016/j.jpdc.2017.08.001.
URL http://www.sciencedirect.com/science/article/pii/S0743731517302319

[17] T. H. Nguyen, M. Forshaw, N. Thomas, Operating policies for energy efficient dynamic server allocation, Electronic Notes in Theoretical Computer Science 318 (2015) 159 – 177, twenty-ninth and thirtieth Annual UK Performance Engineering Workshops (UKPEW). doi:https://doi.org/10.1016/j.entcs.2015.10.025.
URL http://www.sciencedirect.com/science/article/pii/S1571066115000663

[18] K. Chen, W. M. Lang, K. Zheng, W. J. Ouyang, Research on the cloud storage security in big data era, Proceedings of the 2015 International Conference on Applied Science and Engineering Innovation 12 (2015) 659–664.

[19] V. Chang, G. Wills, A model to compare cloud and non-cloud storage of big data, Future Generation Computer Systems-the International Journal of Escience 57 (2016) 56–76.

[20] Backblaze, Backblaze, https://www.backblaze.com/blog/cloud-storage-durability-vs-availability/ (2020).

[21] Amazon, Amazon case studies, https://aws.amazon.com/solutions/case-studies/all/ (2018).

[22] Coursera, https://www.coursera.org/ (2020).

[23] Netflix, https://www.netflix.com/ (2020).

[24] Airbnb, https://www.airbnb.com/ (2020).

[25] https://www.spotify.com/ (2020).

[26] G. Electric, https://www.ge.com/ (2020).

[27] Hitachi, https://www.hitachi.com/ (2020).

[28] NASA, https://www.nasa.gov/ (2020).

[29] SAP, https://www.sap.com/index.html (2020).

[30] Ubisoft, https://www.ubisoft.com/en-us/ (2020).

[31] T. Coughlin, Market trends are driving digital storage choices for consumer devices [the art of storage], IEEE Consumer Electronics Magazine 6 (4) (2017) 133–136.

[32] Business-Wire, Cloud Storage Market - Forecasts, https://www.businesswire.com/news/home/20170614005856/en/92.48-Billion-Cloud-Storage-Market---Forecasts (2017).

[33] S. Shadroo, A. M. Rahmani, Systematic survey of big data and data mining in internet of things, Computer Networks 139 (2018) 19 – 47.

[34] D. Reinsel, J. Gantz, J. Rydning, Data age 2025: The evolution of data to life-critical don't focus on big data, https://www.seagate.com/our-story/data-age-2025/ (2017).

[35] Host-It, http://www.host-it.ie/online-backup/ (2018).

[36] Druva, https://www.druva.com/blog/understanding-rpo-and-rto/ (2018).

[37] M. Fiedler, T. Hossfeld, P. Tran-Gia, A generic quantitative relationship between quality of experience and quality of service, IEEE Network 24 (2) (2010) 36–41.

[38] L. Yang, W. K. Seah, Q. Yin, Improving fairness among tcp flows crossing wireless ad hoc and wired networks, in: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing, MobiHoc '03, Association for Computing Machinery, New York, NY, USA, 2003, p. 57–63.

[39] A. Brown, T. Green, Virtual reality: Low-cost tools and resources for the classroom, TechTrends 60 (5) (2016) 517–519.

[40] G. Schofield, G. Beale, N. Beale, M. Fell, D. Hadley, J. Hook, D. Murphy, J. Richards, L. Thresh, Viking vr: Designing a virtual reality experience for a museum, in: Proceedings of the 2018 Designing Interactive Systems Conference, DIS '18, ACM, New York, NY, USA, 2018, pp. 805–815.

[41] O. Tepper, H. Rudy, A. Lefkowitz, K. Weimer, S. Marks, C. Stern, E. Garfein, Mixed reality with hololens: Where virtual reality meets augmented reality in the operating room, Plastic and Reconstructive Surgery 140 (5) (2017) 1066–1070.

[42] J. C. P. Chan, H. Leung, J. K. T. Tang, T. Komura, A virtual reality dance training system using motion capture technology, IEEE Transactions on Learning Technologies 4 (2) (2011) 187–195.

[43] A. Elmezeny, N. Edenhofer, J. Wimmer, Immersive storytelling in 360-degree videos: An analysis of interplay between narrative and technical immersion, Journal For Virtual Worlds Research 11 (1).
URL https://jvwr-ojs-utexas.tdl.org/jvwr/index.php/jvwr/article/view/7298

[44] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, M. D. Silva, Vr is on the edge: How to deliver 360° videos in mobile networks, in: Proceedings of the Workshop on Virtual Reality and Augmented Reality Network, VR/AR Network '17, ACM, New York, NY, USA, 2017, pp. 30–35.

[45] P. Flajolet, R. Sedgewick, Analytic Combinatorics, 1st Edition, Cambridge University Press, New York, NY, USA, 2009.

[46] A. Saxena, D. Claeys, H. Bruneel, J. Walraevens, Modeling data backup as batch service with vacations, paper presented at INFORMS 19^{th} Applied Probability Society Conference, Northwestern University, Evanston, USA (2017).

[47] A. Saxena, D. Claeys, H. Bruneel, J. Walraevens, Analysis of the age of data in data backup systems, Computer Networks 160 (2019) 41–50.

[48] A. Saxena, D. Claeys, J. Walraevens, Analysis of Age of data in data backup services, paper presented at Stochmod, Lancaster, UK (2018).

[49] A. Saxena, D. Claeys, B. Zhang, J. Walraevens, Cloud data storage: A queueing model with thresholds, Annals of Operations Research (2019) 1572–9338.
URL https://doi.org/10.1007/s10479-019-03279-y

[50] A. Saxena, S. Subramanyam, P. Cesar, R. van der Mei, H. van den Berg, Efficient, QoE aware delivery of 360° videos on VR headsets over mobile links, in: Proceedings of the 13th EAI International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 156–163.

doi:10.1145/3388831.3388834.
URL https://doi.org/10.1145/3388831.3388834

[51] A. Saxena, S. Subramanyam, P. Cesar, R. van der Mei, H. van den Berg, QoE aware delivery of 360^0 videos on wireless VR headsets, IEEE Transactions on Network and Service Management, *Under Review*.

[52] F. Yu, Y. Wan, R. Tsaih, Quantitative quality estimation of cloud-based streaming services, Computer Communications 125 (2018) 24 – 37.

[53] P. M. Van de Ven, B. Zhang, A. Schörgendorfer, Distributed backup scheduling: Modeling and optimization, 2014 Proceedings IEEE Infocom (2014) 1644–1652.

[54] R. Xia, F. Machida, K. S. Trivedi, A Markov decision process approach for optimal data backup scheduling, 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, June 23-26, 2014 (2014) 660–665.

[55] B. T. Doshi, Queueing systems with vacations — a survey, Queueing Systems 1 (1) (1986) 29–66.

[56] T. Naishuo, Z. Z. George, Vacation Queueing Models, Springer US, 2006.

[57] K. Sikdar, U. C. Gupta, On the batch arrival batch service queue with finite buffer under server's vacation: $M^X/G^Y/1/N$ queue, Computers & Mathematics with Applications 56 (11) (2008) 2861–2873.

[58] R. Arumuganathan, S. Jeyakumar, Steady state analysis of a bulk queue with multiple vacations, setup times with n-policy and closedown times, Applied Mathematical Modelling 29 (10) (2005) 972–986.

[59] K. Sikdar, U. C. Gupta, Analytic and numerical aspects of batch service queues with single vacation, Computers & Operations Research 32 (4) (2005) 943–966.

[60] H. W. Lee, S. S. Lee, K. C. Chae, R. Nadarajan, On a batch service queue with single vacation, Applied Mathematical Modelling 16 (1) (1992) 36–42.

[61] H. Bruneel, B. G. Kim, Discrete-Time Models for Communication Systems Including ATM, Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[62] D. Chen, K. S. Trivedi, Optimization for condition-based maintenance with semi-markov decision process, Reliability Engineering & System Safety 90 (1) (2005) 25–29.

[63] D. Claeys, J.-P. Dorsman, A. Saxena, J. Walraevens, H. Bruneel, A queueing-theoretic analysis of the threshold-based exhaustive data-backup scheduling policy, in: Proceedings of the international conference on Numerical Analysis and Applied Mathematics (ICNAAM), Vol. 1863, 2016, p. 3.

[64] T. Fry, Data Processing, Elsevier Science, 2013.
URL https://books.google.be/books?id=4fz8BAAAQBAJ

[65] A. Zomaya, S. Sakr, Handbook of Big Data Technologies, Springer International Publishing, 2017.
URL https://books.google.be/books?id=SsQ2DgAAQBAJ

[66] D. Boullery, A. Schörgendorfer, P. Van de Ven, B. Zhang, Balanced distributed backup scheduling, US Patent 9,244,777 (Jan. 26 2016).
URL https://www.google.com/patents/US9244777

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

[68] W. B. Powell, P. Humblet, The bulk service queue with a general control strategy: Theoretical analysis and a new computational procedure, Oper. Res. 34 (2) (1986) 267–275.

[69] I. J. B. F. Adan, J. S. H. van Leeuwaarden, E. M. M. Winands, On the application of Rouche's theorem in queueing theory, Operations Research Letters 34 (3) (2006) 355–360.

[70] A. S. Alfa, Applied Discrete-Time Queues, 2nd Edition, Springer Publishing Company, Incorporated, 2015.

[71] N. L. Johnson, A proof of wald's theorem on cumulative sums, The Annals of Mathematical Statistics 30 (4) (1959) 1245–1247.
URL http://www.jstor.org/stable/2237468

[72] J. Liebeherr, A. Burchard, F. Ciucu, Delay bounds in communication networks with heavy-tailed and self-similar traffic, IEEE Transactions on Information Theory 58 (2) (2012) 1010–1024.

[73] TEKLINKS, Service Level Agreements, https://www.teklinks.com/wp-content/uploads/2017/01/Disaster-Recovery-SLA-20170109.pdf (2018).

[74] J. Abate, W. Whitt, Numerical inversion of probability generating functions, Operations Research Letters 12 (4) (1992) 245 – 251.

[75] J. Liebeherr, A. Burchard, F. Ciucu, Delay bounds in communication networks with heavy-tailed and self-similar traffic, IEEE Transactions on Information Theory 58 (2) (2012) 1010–1024.

[76] C. Cheng, J. Li, Y. Wang, An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing, Tsinghua Science and Technology 20 (1) (2015) 28–39.

[77] Z. Niu, X. Guo, S. Zhou, P. R. Kumar, Characterizing energy–delay tradeoff in hyper-cellular networks with base station sleeping control, IEEE Journal on Selected Areas in Communications 33 (4) (2015) 641–650.

[78] Shun-Ren Yang, Yi-Bing Lin, Modeling UMTS discontinuous reception mechanism, IEEE Transactions on Wireless Communications 4 (1) (2005) 312–319.

[79] I. Dimitriou, A modified vacation queueing model and its application on the discontinuous reception power saving mechanism in unreliable long term evolution networks, Performance Evaluation 77 (2014) 37 – 56.
URL `http://www.sciencedirect.com/science/article/pii/S0166531614000431`

[80] I. Dimitriou, Queueing analysis of the drx power saving mechanism in fault-tolerant 3gpp lte wireless networks, Annals of Operations Research 239 (2) (2016) 521–552.

[81] A. Gautam, G. Choudhury, S. Dharmaraja, Performance analysis of drx mechanism using batch arrival vacation queueing system with n-policy in lte-a networks, Annals of Telecommunications.

[82] M. Agiwal, A. Roy, N. Saxena, Next generation 5g wireless networks: A comprehensive survey, IEEE Communications Surveys Tutorials 18 (3) (2016) 1617–1655.

[83] Y. Sun, Z. Chen, M. Tao, H. Liu, Communication, computing and caching for mobile VR delivery: Modeling and trade-off, in: 2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018, 2018, pp. 1–6.

[84] X. Yang, Z. Chen, K. Li, Y. Sun, N. Liu, W. Xie, Y. Zhao, Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff, IEEE Access 6 (2018) 16665–16677.

[85] F. Wang, Z. Fei, J. Zheng, J. Wang, Qoe-aware mobile vr has cache management with coding helper, IEEE Access 6 (2018) 44556–44569.

[86] S. Petrangeli, V. Swaminathan, M. Hosseini, F. De Turck, An http/2-based adaptive streaming framework for 360 virtual reality videos, in: Proceedings of the 25th ACM International Conference on Multimedia, MM '17, ACM, New York, NY, USA, 2017, pp. 306–314.

[87] A. T. Nasrabadi, A. Mahzari, J. D. Beshay, R. Prakash, Adaptive 360-degree video streaming using scalable video coding, in: Proceedings of the 25th ACM International Conference on Multimedia, MM '17, ACM, New York, NY, USA, 2017, pp. 1689–1697.

[88] F. Qian, L. Ji, B. Han, V. Gopalakrishnan, Optimizing 360 video delivery over cellular networks, in: Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges, ATC '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 1–6. doi:10.1145/2980055.2980056.
URL https://doi.org/10.1145/2980055.2980056

[89] Oculus, https://code.fb.com/virtual-reality/enhancing-high-resolution-360-streaming-with-view-prediction/ (2018).

[90] T. Judd, K. Ehinger, F. Durand, A. Torralba, Learning to predict where humans look, in: 2009 IEEE 12th International Conference on Computer Vision, 2009, pp. 2106–2113.

[91] C. Wu, Z. Tan, Z. Wang, S. Yang, A dataset for exploring user behaviors in vr spherical video streaming, in: Proceedings of the 8th ACM on Multimedia Systems Conference, MMSys'17, ACM, New York, NY, USA, 2017, pp. 193–198.

[92] Y. Bao, T. Zhang, A. Pande, H. Wu, X. Liu, Motion-prediction-based multicast for 360-degree video transmissions, in: 2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2017, pp. 1–9.

[93] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, X. Liu, Shooting a moving target: Motion-prediction-based transmission for 360-degree videos, in: 2016 IEEE International Conference on Big Data (Big Data), 2016, pp. 1161–1170.

[94] X. Hou, S. Dey, J. Zhang, M. Budagavi, Predictive view generation to enable mobile 360-degree and vr experiences, in: Proceedings of the 2018 Morning Workshop on Virtual Reality and Augmented Reality Network, VR/AR Network '18, ACM, New York, NY, USA, 2018, pp. 20–26.

[95] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st Edition, John Wiley & Sons, Inc., USA, 1994.

[96] E. d'Eon, B. Harrison, T. Myers, P. A. Chou, 8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset, ISO/IEC JTC1/SC29 joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva (January 2017).