國立臺灣大學電機資訊學院電機工程學研究所

碩士論文

Graduate Institute of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

以最大概略估計為基礎之捉放法模型估算伺服器數量

Estimation on Server Population with MLE-based CMR (Capture-Mark-Recapture)

翁瑋襄

Wei-Shiang Wung

指導教授: 黃寶儀 博士

Advisor: Polly Huang Ph.D.

中華民國 111 年 6 月

June, 2022

doi:10.6342/NTU202201298

Acknowledgements

從研究的開頭到論文撰寫完成，這段期間經歷的許多考驗，要感謝很多人的協助與鼓勵，才讓我逐一克服困難走到現在。

首先非常感謝指導教授 Polly Huang，從一開始大學部的專題研究，每當我們遇到問題時指引我們我解決問題的方向，到後來我們升上研究所，Polly 逐漸放手讓我們自己探索研究方法，甚至是最後讓我們獨立完成論文，我覺得自己在這兩年半的訓練中成長非常多。Polly 每次跟我們談話的內容不僅僅是研究方面，也會給予我們一些人生啟發和未來發展建議，讓我們對未來有更好的規劃與夢想。

另外也要謝謝中研院的陳伶志教授與陽明交通大學的林靖茹教授，兩位老師給予許多深度且建設性的建議，幫助我的論文邏輯架構更加完整。

這邊也要感謝實驗室的同學許誠與黃珮欣。我跟許誠是合作的夥伴，我們平時討論研究方向與論文架構，也是相互勉勵的好戰友；珮欣則是給予我許多報告上的建議。有了他們的協助與參與，讓我的碩士生活留下深刻回憶。

最後感謝從小到大一路支持我的父母，沒有他們無怨無悔的付出，就不會有現在的我。

doi:10.6342/NTU202201298

中文摘要

影音服務的品質與順暢度取決於內容傳遞網路 (CDN) 的規模與設備完善程度。近年來因為影音服務需求的成長，為了因應客戶的需求，內容傳遞網路中的伺服器數量也顯著的增加。因為 Twitch 廣泛的影音應用，我們認為長時間持續續的探究其內容傳遞網路架構是一個重要課題。發想於內容傳遞網路中的伺服器數量的新增淘汰和動物群體出生死亡行為的相似性，我們認為可以套用野生群體數量估算的捉放法，透過每次少量的網路流量取樣以達到估算整體內容傳遞網路的伺服器數量。在我們過去發表的 AINTEC 論文 (2021) 中，Cormack-Jolly-Seber (CJS) 模型能夠相對準確的估算出伺服器總數。

然而，傳統的 CJS 模型的機率假設仍有很多的限制。因為其需要較多採樣間距估算才能收斂，導致這個模型僅限於離線的估算。此外，這個模型假設所有個體被抓捕和生存的機率都是一樣的，這個假設並不符合 Twitch 的內容傳遞網路中伺服器的服務型態。因此，我們引入了考慮異質性、以最大概似估計為基礎的 CJS 捉放法模型來解決這兩個議題。這個模型不僅可以賦予每一台伺服器不同的參數設定，還會以最大概略估計一次性估算 CJS 機率模型中的所有參數。不過每一伺服器都有相對應的參數會導致整個機率模型過於複雜，我們因此使用分群法按照提供服務的模式將伺服器分群，透過讓同一群伺服器共用參數以達到減少模型的參數數量。我們使用 2021 年五月蒐集的資料集做測試，發現以最大概似估計

doi:10.6342/NTU202201298

為基礎的 CJS 模型的確在在線估算中有較好的表現，而異質性和伺服器分群在實驗中並未有效提升估算準確率，我們透過檢視各分群的估算結果詳細分析其中原因。

關鍵字： Twitch，捉放法，伺服器數量估計，最大概略估計，異質性

doi:10.6342/NTU202201298

Abstract

The quality and continuity of the video services such as Twitch depend on the scale and well-being of their content distribution networks (CDNs). Due to the growing demand for video services, server numbers in the CDNs have rapidly increased to feed videos to the clients. Given the widespread use of Twitch, we find continuous survey of its CDN an important subject of study. Inspired by Capture-Mark-Recapture(CMR), a methodology widely used to estimate animal population, we developed a system to continuously observe its CDN size (i.e., the number of servers) with lightweight probing. According to our previous research in AINTEC, the Cormack-Jolly-Seber (CJS) model can estimate the CDN size at each sample time with relatively low errors.

Nevertheless, the assumptions of the traditional CJS model are still restrictive. Due to its long converging period, the model can only estimate server population offline. Besides, it assumes that all servers share the same capturing and survival rates, which does not meet the server patterns in Twitch's CDN. Therefore, we introduce the Maximum-Likelihood-

doi:10.6342/NTU202201298

Estimation-based (MLE) CJS model with heterogeneity to address these two issues. It not only allows different parameters for each server but also co-estimates all parameters in the CJS probability model. The resulting MLE model is too complicated, and thus we try server clustering to reduce the parameter space. Using a data set collected in May 2021, we find the MLE-based CJS indeed performs better in online estimation. Heterogeneity and server clustering, on the other hand, do not improve the estimation accuracy. For these worse results, we identify the detailed reasons with the estimation results in each group.

Keywords: Twitch, Capture-Mark-Recapture, Server Population Estimation, Maximum Likelihood Estimation, Heterogeneity

doi:10.6342/NTU202201298

Contents

doi:10.6342/NTU202201298

doi:10.6342/NTU202201298

doi:10.6342/NTU202201298

List of Figures

doi:10.6342/NTU202201298

doi:10.6342/NTU202201298

List of Tables

doi:10.6342/NTU202201298

Chapter 1

Introduction

Internet video accounts for more than 80% of global traffic today. Among the contributors, video on demand (VoD) dominates. Live video, although lower in traffic volume today, is rapidly growing. Amid the pandemic, the demand for video services has skyrocketed. A 45% growth is observed from March to April 2020 [8]. Among all Internet services contributing to this prosperity, Twitch dominates and is responsible for 43.6% of the live video traffic. In fact, the amount of traffic Twitch generates is just behind Netflix, Apple and Google [15]. The viewership of the *eSports* events on Twitch often overtakes the viewership of the traditional, physical sports events. The *League of Legends World Championship* finals, for example, brought in 99.6M viewers, surpassing the 98M viewership of *Super Bowl* in 2018[25]. Twitch, as a major contributor to the global traffic and a near monopoly of the live video streaming market, is an important subject of study.

Similar to many global-scale Internet services today, Twitch operates on a content distribution network (CDN). In a live session, a video stream is uploaded in real time and duplicated to one or more servers on the CDN. The viewers, in the meantime, are directed

doi:10.6342/NTU202201298

to one of these servers to fetch the stream. Given the stringent bandwidth and delay requirement, the size and distribution of the CDN determine the quality and availability of the service. They are hence a critical element in the system design.

Deng et al. [13] are the first to crawl and discover Twitch's CDN. By repeatedly requesting the video channels available, they found a total of 783 servers scattered around North America, Europe, Asia, and Oceania in early 2016. Similar effort has been reported for other video distribution systems such as Hulu [12], YouTube [16], Netflix [3], and Periscope [14]. These works, however, are one-time effort. With a user base growing at about 120% per year, Twitch's CDN is a moving target. A one-time measurement study satisfies the need to know but does not lead to better understanding or management of the system. Towards a better view of the service's evolution and the ability to detect outages or mis-configurations, we see the need to survey Twitch's CDN infrastructure continuously.

Taking a detailed scan of a service CDN is costly. Each scan involves repeated probes to the service usher, i.e., the managing server at a higher level that determines the video server per viewer request. To observe the CDN infrastructure continuously, a naive way is to trigger such a scan time after time. This will impose a significant amount of traffic and workload on a regular basis. The more frequent we scan, the higher the overhead is. In addition, the overhead per scan increases over time as well. In [13], the authors recorded over 700k requests to cover all video channels in 2016. The number of channels in 2021 is 600% higher. It is quite clear that naive approach will not scale surveying a growing system.

To better scale the measurement effort, we resonate with these works [20, 33]. Both of them proposed to detect anomalies in Internet-scale services in two phases. The first phase is lightweight and tracks only indicators of anomalies. When an anomaly is anticipated, the second phase is triggered to perform a detailed scan. The approach fits well to the survey of Internet-scale CDNs. In that, one can estimate the CDN size first and trickle down to a full scan when a significant level of size change is detected. CDN size might not be the only indicator for the purpose. It is nonetheless plausible as a sudden decease or increase in CDN size indicates an outage or expansion in the deployment.

The challenge lies in achieving accurate CDN size estimation with lightweight probing. In our previous study [39], we drew similarity surveying animal population in a natural environment and server population in a CDN. We found that Cormack-Jolly-Seber (CJS) [9, 21, 36], a well-known CMR methodology, is efficient to estimate server population with low traffic overhead. In addition, we identified that the best parameter combination with 1 hour sampling duration and 24 hours sample spacing can fit the US data set in 2020 well with an error rate lower than 10%.

However, the traditional CJS model is still restrictive. It assumes that all individuals share the same survival and capture probability. Apparently, the assumption is too simple because not all servers share the same service schedules. Besides, since the traditional CJS estimates population based on the assumption of equality for future capturing probability, it requires a longer time period for convergence. Hence, computations of the traditional CJS can be done offline only.

In this study, we expand our previous work [39], introducing Maximum-Likelihood-Estimation-based CJS model with heterogeneity [29, 30] to handle these two issues. The estimations of both the traditional and MLE-based CJS will converge to similar accuracy under the assumption of identical capturing and survival rates for all servers. The main

doi:10.6342/NTU202201298

difference of these two models lies in the estimation process. The MLE-based CJS model co-estimates all parameters of the CJS probability model, yet estimations of the traditional CJS come from an equality of future capturing probability at each sample point. Thus, even though the MLE-based model still relies on capturing history and future records for population estimation, provided the same amount of data, it approaches a similar or higher level of accuracy. Furthermore, the model with heterogeneity allows individuals with different capturing probability and survival rate. By assigning a pair of survival and capture rate to each server, the estimation accuracy may be improved. However, the resulting MLE problem will be of extremely high dimension and may not converge. To reduce the parameter space, we can cluster the servers into several groups. Servers in each group sharing some similarities are assigned the same survival rate and capturing probability. As a result, server clustering can not only maintain the estimation accuracy but also reduce the computational overhead.

Using two data sets collected in November 2019 and May 2021, we first compare the two variations of CMR model assuming identical capturing probability and survival rate for all servers. Additionally, we try a variety of clustering strategies such as IP prefix, server working frequencies, and K-means to improve the estimation results. The major findings in this study are that: (1) MLE-based model can estimate server population with a higher or similar level of accuracy provided the same amount of observations (2) Estimate server population with heterogeneity (3) Identify the detailed reasons to explain why clustering-based estimations lead to worse results.

The remainder of the thesis is organized as follows. Our previous study in AINTEC 2021, an R package for MLE-based CJS model, and K-means are briefly summarized in

doi:10.6342/NTU202201298

Chapter 2. A short introduction to Twitch and a general evaluation of the US data set collected by Wang[38] are provided in **Chapter 3**. Next, we compare the traditional and MLE-based CJS models through offline, online, and delayed server population estimations in **Chapter 4**. Then, we further use different server clustering strategies for improving the estimation accuracy and evaluate the server clusters in detail in **Chapter 5**. A final conclusion is given in **Chapter 6**.

doi:10.6342/NTU202201298

Chapter 2

Related Work

The challenge to tackle in this study is how we monitor the server population continuously with as little probing traffic as possible. We find the problem very similar to that of surveying animal population in the wild. In that, frequent and exhaustive probes are costly and disturbing to the ecosystem we aim at preserving, not to mention that one can never be sure of the true population. Previous study has shown that the use of Capture, Mark, and Recapture (CMR) [22] for server population estimation leads to relatively accurate results. We relaxed the assumption of CMR model and applied Maximum Likelihood Estimation (MLE) to do further evaluations on Twitch servers. In addition, clustering Twitch servers based on server show time gives insight to Twitch's strategies for server assignments.

2.1 Lincoln-Peterson Model

Lincoln-Petersen (LP) [24, 28] is the simplest of the CMR methods. To begin with, one would capture a few animals, mark, and release them back to the wild. The proportion of the marked animals at this point will be the number of animals captured (C) over the

doi:10.6342/NTU202201298

entire population (N). To close the deal, one would capture again. The proportion of the marked animals should equal the number of marked animals (M) over the number of animals captured in the 2nd round (R). Knowing C, M and R, one derives the animal population N as Eq. (2.1).

$$N = \frac{RC}{M} \tag{2.1}$$

The LP method works for a close population and assumes: (1) the animal population in between 2 captures does not change and (2) the probability of the animals being captured is independent and identical (i.i.d) over time. These can be too strong for CDN discovery, where the server population is likely changing between crawling events and the chance of discovering a server is not i.i.d, knowing the server allocation is biased to its proximity to the client. Next, we introduce an open population CMR method that relaxes the assumptions.

2.2 Cormack-Jolly-Seber Model

Cormack-Jolly-Seber (CJS) [9, 21, 36] is designed to estimate an open population. In that, an animal might stay alive with a varying survival rate over time, i.e., the population can be dynamic. An animal might be captured with a varying probability over time as well, i.e., the chance of an animal being captured does not need to be uniform, nor identical. The way it works is to capture and release the animals continuously. With the capturing history, it co-estimates the population, survival rate, and capturing probability at every capture. The population at capture t (N_t) is calculated as Eq. (2.2), where M_t is the

doi:10.6342/NTU202201298

number of marked animals and PM_t the proportion of marked animals.

$$N_t = \frac{M_t}{PM_t} \tag{2.2}$$

In the interest of space, we present the most intuitive derivation [9] of the key term M_t , and divert the readers to [36] for details. Consider the probability of marked animals being caught again in the future are identical for both marked animals released after the t th capture (every animals released are marked) and marked animals not captured in the t th capture. We can derive Eq. (2.3).

$$\frac{R_t}{CN_t} = \frac{Z_t}{M_t–CM_t} \tag{2.3}$$

CN_t is the number of animals captured in round t and CM_t the number of marked animals captured in round t. R_t is the number of animals, captured in round t, being recaptured in the future. Z_t is the number of marked animals not captured in round t, but recaptured in the future. By manipulating the terms in Eq. (2.3), we come to M_t and PM_t as Eq. (2.4) and Eq. (2.5).

$$M_t = \frac{(CNt) * Zt}{R_t} + CMt \tag{2.4}$$

$$PM_t = \frac{CM_t}{CN_t} \tag{2.5}$$

Note the two terms, R_t and Z_t . They account for the chance of the animals being

doi:10.6342/NTU202201298

recaptured in the future, which depends on whether they will survive to a future time and the chance of them being recaptured at the time. The two terms are essentially compounds of survival rates and capturing probabilities into the future. That is why, in CJS, the population estimations in the past are often adjusted as they are regressed to fit the new data. As the new data are added, the value of R_t and Z_t are affected. In the meantime, the population estimations in the past are adjusted. CJS is unique in that it takes into account data in the past and the future. Eq. (2.4), Eq. (2.5) produce a biased estimation of the population. The tendency is to overestimate and the bias can be large for small samples (e.g., animals that are hard to capture or close to extinction), and the following extension [34] is often applied to mitigate the bias.

$$M_t = \frac{(CN_t + 1) * Z_t}{R_t + 1} + CM_t \tag{2.6}$$

$$PMt = \frac{CM_t + 1}{CN_t + 1} \tag{2.7}$$

2.3 Estimation on Server Population with CJS Model

In the previous research [39], the best way to observe a CDN infrastructure is to scan the CDN network in detail. However, the CDN discovery is costly and probing overheads are too high. Thus, we would like to propose a system that we can supervise the CDN in real time with lightweight probing overheads in the AINTEC paper. The challenge lies in achieving accurate CDN size estimation with only a few samples. To this end, we drew similarity surveying animal population in a natural environment and server population in

doi:10.6342/NTU202201298

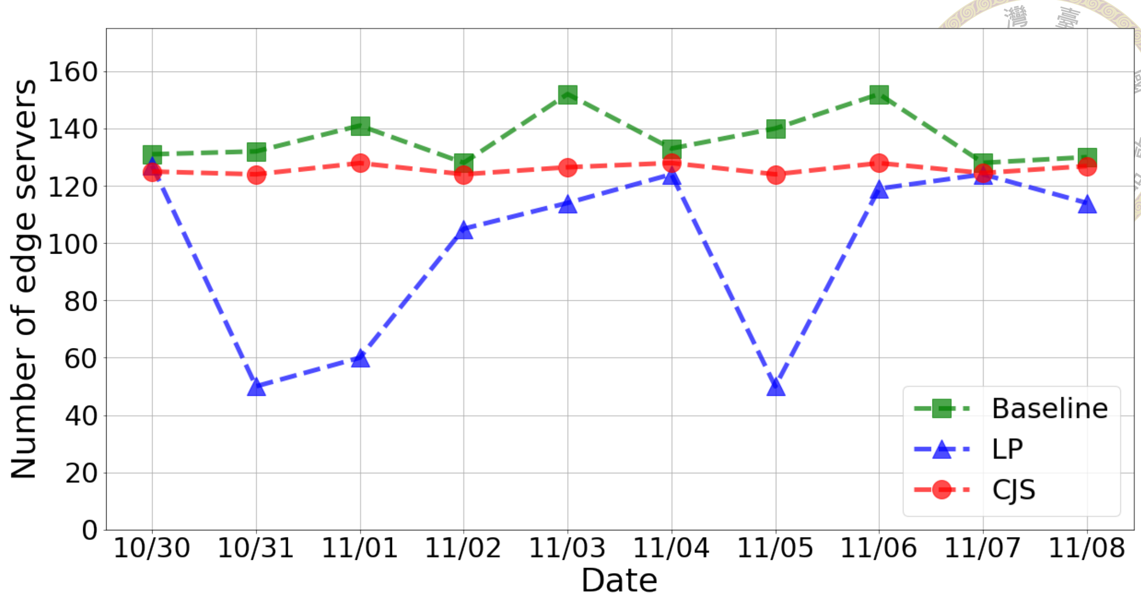


Figure 2.1: CDN Population Estimation: CJS vs. LP in 2019

a CDN. That points us to CMR, a well-known methodology in ecological statistics. The literature suggests that by marking the animals captured previously and examining the probability the marked animals being captured again, one can infer the population with little capturing effort. In our previous study, we experimented and compared two CMR models, i.e., the Lincoln-Petersen (LP) and the Cormac-Jolly-Seber (CJS) model. The former is the simplest of all CMR models. It is also the most restrictive by assuming the population is closed. That means the population does not change between two captures. The latter is more relaxed and allows estimation of an open population.

We took the West-US data set collected in Nov 2019 and assumed an 1 hour sampling duration. Since there is no public Twitch data set, we define the server population observed in our data set as the baseline of comparison. For LP, two samples were taken at 8am and 4pm each day. For CJS, we only sampled at 4pm each day which is 50% less probing traffic than LP. We can see in **Figure 2.1** that CJS tracks the baseline better than LP. In short, CJS is the better model both in terms of accuracy and overhead. We therefore focus further discussions on CJS.

doi:10.6342/NTU202201298

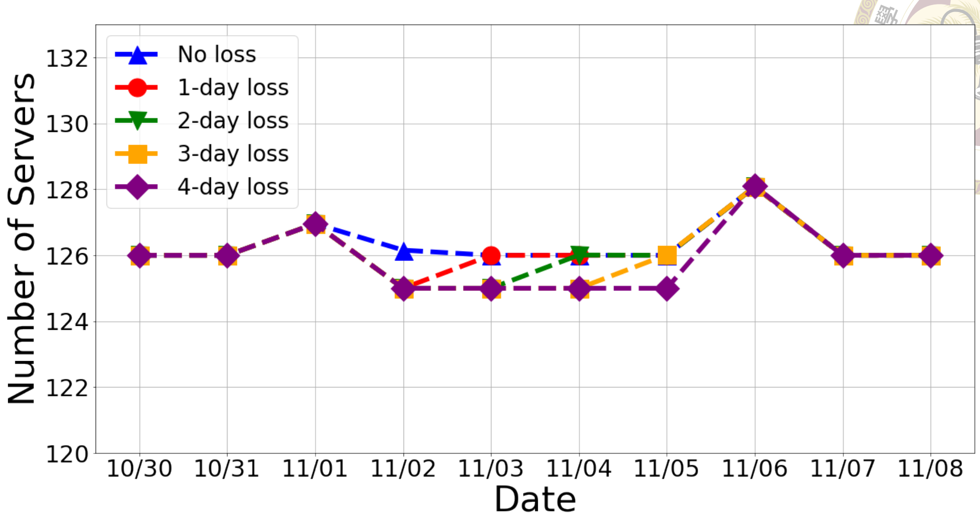


Figure 2.2: Effects of Having Probing Failures

To estimate the server population, multiple captures are required. The duration of each capture (D) and the time spacing between the captures (S) will influence the probing overhead and the estimation accuracy. In fact, the trade-off between overhead and estimation error holds true under all parameter combinations we have observed. Hence, we would like to look for the parameters combination that minimizes the probing overhead under a certain estimation error budget. We consider combinations of D and S where the estimation error rate is less than 10%. With the best setting for D equals to 1 and S equals to 24, we find a 7.25% average estimation error in the West-US data set. Besides, provided an estimation error bound, we can identify as well the best parameter combination to minimize the probing traffic. Furthermore, we did an experiment to test the model robustness to probing failure by removing data entries for 1 to 4 days. As shown in **Figure. 2.2**, the influence is localized within the duration of the outage as well. As soon as the data re-enter the repository, the estimation accuracy recovers rapidly.

doi:10.6342/NTU202201298

2.4 MLE-based CJS Model

The CJS model mentioned above is still restrictive. The assumption of the survival and capture rate being the same across individual animals is not yet relaxed. Hence, we would like to introduce the Maximum-Likelihood-Estimation (MLE)-based model with heterogeneity [29, 30] to tackle this issue. Assuming an open population is a multinomial distribution model, the MLE-based CJS model estimate capturing probability and survival rate at each sample point. For estimating Twitch's CDN size, we can treat the probability a server is consistent for providing service as the survival rate and the probability an online server is discovered in a sample as capturing probability. Considering these two factors, we are able to build the corresponding multinomial probability model and apply the MLE-based CJS to estimate Twitch's server population.

Generally, individuals vary in capturing probability and survival rate. To solve the problem, a flexible framework of MLE-based models allowing individual heterogeneity was proposed [34]. For each individual, its capturing history can be transformed to a probability formula with capturing probability and survival rate. By multiplying the probability formulas of all individuals, a multinomial distribution probability model is constructed. Considering heterogeneity, two factors, individual and sample time, will influence the value of capturing probability and survival rate. This model assumes that the logit scales of capturing probability and survival rate can be expressed as summation of the mean value, the terms related to individual and sample time respectively and a covariance term of individual and sample time. Hence, the probability model is transformed into logit scale. Given the constraints of variances and covariance, MLE will iteratively search the

doi:10.6342/NTU202201298

values of capturing probability and survival rate with lowest error rate. The final step is to transform the outputs back to normal scale. Further introductions about MLE-based CJS model are in these references.

However, the resulting MLE problem is of extremely high dimension and may not converge. The Pollock-Robson model [5, 31], on the other hand, exploits similarity in a group of animals and/or within a time period. Thus, we attempt to divide these animals into several groups. Animals in the same group share similar behaviors and properties. Assuming capturing probability and survival rate are all different for each group and at each sample, a complicated CJS probability model is built. Then, maximum likelihood estimation [7, 10, 26, 32] is applied to search the values of capturing probability and survival rate that leads to the lowest error rate. The model quality is evaluated by Akaike information criterion (AIC) [1]. AIC is positively correlated to the number of parameters in the model and negatively correlated to the maximum value of the likelihood function. Thus, lower AIC value usually reflects the goodness of model fitting and the simplicity of model structure.

Many R packages implemented the framework with individual heterogeneity for analysis of capture-recapture data. We applied one of these packages called ‘marked‘ to our study. ‘marked‘ [23] published in 2013 is a relatively new package, and the running efficiency of ‘marked‘ is better than other old packages. In addition, ‘marked‘ is an open source software. This enables users to fully understand the CJS model structures. Another R package called ‘OpenCR‘[2, 11, 30, 35] is an implementation of the MLE-based CJS model as well. The estimation results of these two packages are almost the same. Hence, we mainly focus on server number estimations with ‘marked‘ in this study.

doi:10.6342/NTU202201298

2.5 K-means Clustering

To reduce the parameter space of the MLE model, we need to clusters the animals first. K-means [17] is a very effective unsupervised clustering method for this task.

Given the number of clusters K, K-means algorithm divides N individuals X into K disjoint clusters C. For the ith cluster, it can be described by its mean (a.k.a centroid) μ_i. The goal of the K-Means algorithm is to find the mean value μ_i of each group that minimize the inertia, which is defined as: $\sum_{i=0}^{N} \min_{\mu_j \in C}(\|x_i - \mu_j\|^2)$.

K-means contains three main steps. First of all, it randomly chooses K initial centroids. The second step assigns each individual to the nearest centroid. Third, compute the mean value of all the individuals in each cluster as the new centroid. Step two and step three will be repeated iteratively until the difference between old and new centroids is less than a threshold.

In this study, clustering servers with K-means enables further discussions on the service schedules of Twitch's servers. The API of K-means clustering from scikit-learn [6, 27] is used for server clustering in **Chapter 5**.

doi:10.6342/NTU202201298

Chapter 3

Data Set

The target of our study is Twitch. It is one of the many live video services existing. In this chapter, we will briefly introduce Twitch's CDN, data collection, and data analysis.

3.1 Introduction to Twitch

Unlike VoD, live video services such as Twitch appeal to a different market. Launched officially in 2011, Twitch has created a new entertainment industry targeting gamers. A good proportion of the revenue comes from hosting the *eSports* events. As of August 2021 [37], Twitch has recorded 6.5M concurrent viewers and 233K concurrent channels at its peak.

For each viewing request, the client initiates a connection to Twitch's proprietary load balancer (Usher), which returns an index file containing the locations of the playlist server corresponding to different video qualities. As depicted in **Figure 3.1**, by contacting the playlist server, the client acquires a video playlist (.m3u8 file) consisting of URLs in

doi:10.6342/NTU202201298

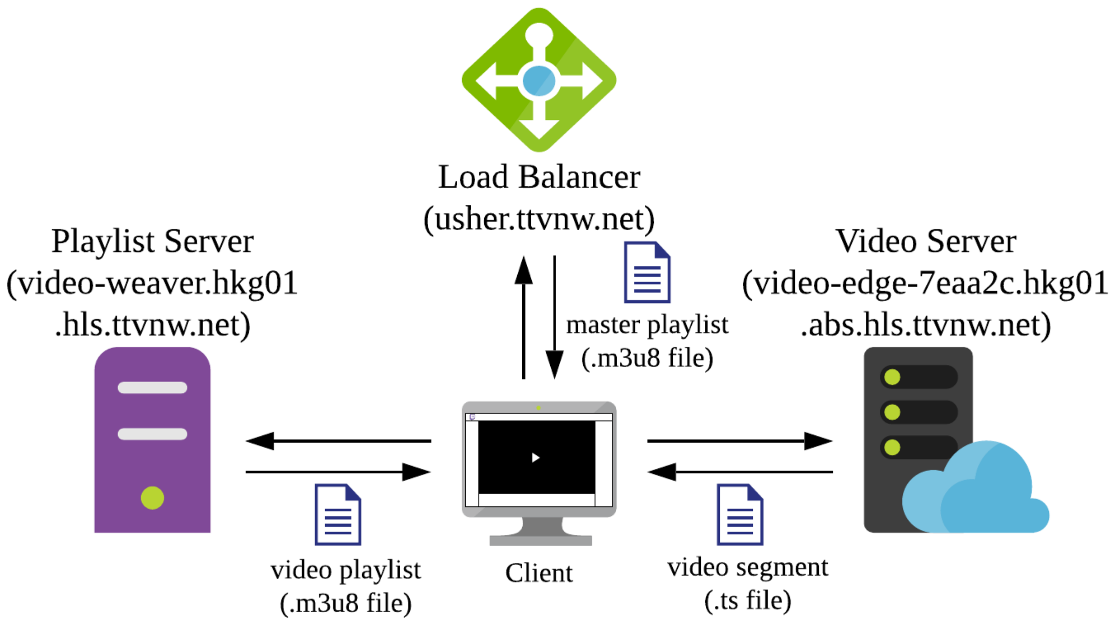


Figure 3.1: 3-Way Redirection Video Lookup

sequential order, each pointing to the content server of a video segment (.ts file).

The best knowledge for Twitch's CDN was published in 2017 by Deng et al. [13]. By repeatedly requesting videos of all the channels from widespread locations for 5 months in early 2016, the authors discovered that Twitch' s CDN was centralized. The outgest servers were clustered in 4 groups and physically located in US, Switzerland, Hong Kong, and Sydney. The 4 clusters were accessed primarily by viewers in North America, Europe, Asia, and Oceania. The sizes were 360, 257, 119, and 47, respectively.

Inferred from the data was the outgest server allocation policy. First of all, the viewers were mainly redirected to outgest servers in the same region. Asian viewers however were more often redirected to non-Asian regions due to poor peering to the Asian cluster at the time. Secondly, a popular channel may be hosted by multiple outgest servers. These edge servers can span multiple regions to allow access from viewers of geographical spread. Lastly, the number of outgest servers assigned to a channel correlated loosely to the number of viewers in the channel. These findings suggested a dual preference to

doi:10.6342/NTU202201298

delay and load.

3.2 Data Collection

Borrowing Twitch's API, Wang [38] implemented the crawler by using Node.js and concurrently crawled via a number of VPNs spanning west and east coast of US to discover the US cluster in Twitch's CDN. This was achieved by running the Docker engine on a local machine. The crawler instances ran containerized above the system kernel. The approach was less resource-intensive while being able to isolate applications and ensure that they work uniformly across containers. The crawler started by acquiring the list of active streams and the number of viewers per stream. To avoid being flagged as an abusive user, Wang limited the scope and the number of streams he crawled. Note that the problem in this work is not Twitch CDN Discovery. Thus, the data set does not need to be complete. The crawler only targeted the top streams that account for 80% of the viewers. Next, the crawler requested each of the top streams and stored the hostname and IP address of each video servers discovered. Note that the video server hostname is embedded in the .m3u8 file. Such a round of requests was repeated continuously at the frequency of every 20 minutes in May 2021. This data set contains dense and repeated rounds of requests. The number of unique IP addresses observed is considered the best knowledge server population. It serves also as the baseline that we compare the estimates from the two sampling mechanisms with.

doi:10.6342/NTU202201298

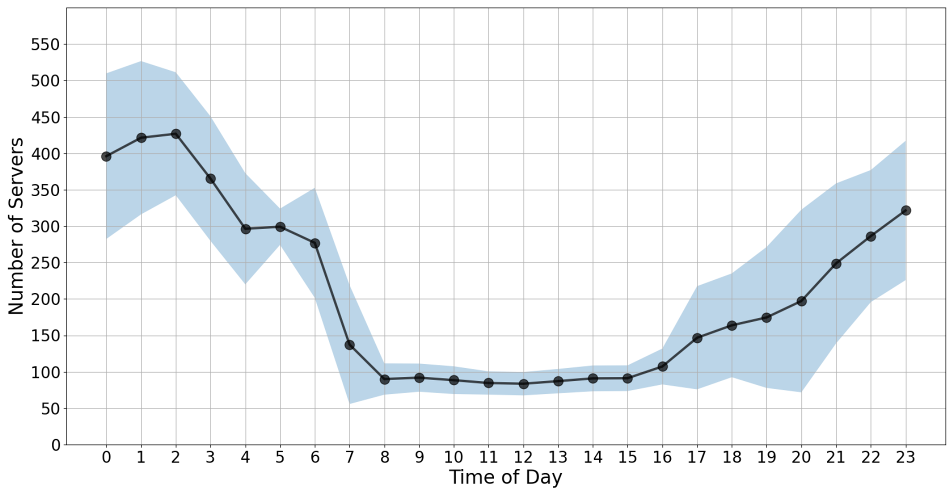


Figure 3.2: Average server count by hour in US

3.3 Data Analysis

3.3.1 Hourly Server Count

Depicted in **Figure 3.2** is the average hourly server count. The blue shadow indicates the value range of adding or subtracting one standard deviation. Since the data set was collected through VPN, it is presented by UTC+8 (Taiwan) time zone. In the figure, the server count peaks around 0-2 am and stays low from 8am to 6pm.

The servers are observed by repeated requests to the streams watched by 80% of the viewers. In general, the more streams there are the more servers we observe. As the number of active streamers may be higher during certain hours of a day, the number of streams we request and therefore the number of servers we observe can vary as well. Furthermore, as shown in [13], the number of servers allocated per stream correlates to the number of viewers. These are to say the number of unique servers observed in each hour indicates how active the streamers and viewers are in the hour.

doi:10.6342/NTU202201298

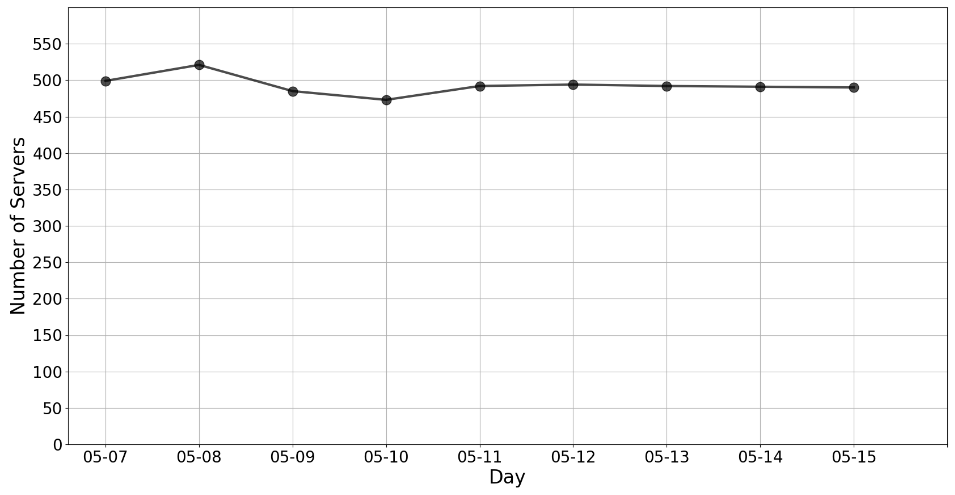


Figure 3.3: Daily Server Counts in US

The time zone of mainland United States covers from UTC-5 to UTC-7. Thus, the 0-2 am peak in UTC+8 time zone is roughly noon in the US. This time period is usually people's leisure and personal time. It is not surprising to see more users logging on to Twitch for entertainment. On the other hand, 8am to 6pm in UTC+8 time zone corresponds to the sleeping time in the US. Fewer viewers lead to fewer servers on service. Hence, the server numbers stay low at this period of time.

3.3.2 Daily Server Count

Figure 3.3 shows the daily server counts. The number ranges from 473 to 521 in the US cluster in May 2021. Compared with fewer than 170 servers in Nov 2019, the daily service demand grew rapidly from 2019 to 2021. This shows that a large number of viewers have joined Twitch in recent years. In addition, the standard deviation of the daily count is 12.02, which accounts for only 2.5% in the cluster. This shows that the daily server count is relatively stable in the US cluster.

doi:10.6342/NTU202201298

Chapter 4

Preliminary Experiment

In this chapter, we aim to compare the estimation results from the traditional CJS model and the MLE-based one. All servers are assumed to have the same capturing probability and survival rate. But these two parameters may change from time to time. We first introduce how to apply marked on our data set and estimate population in 4.1. Then we will compare the offline estimation results from both models in 4.2. At last, we want to extend the two methods in online estimation and compare the results with further discussions.

4.1 Methodology

As mentioned in **Section 2.4**, 'marked' is an R package to estimate the capturing probability and survival rate of an open population [23]. To estimate server population with 'marked', several operations need to be done. Considering execution time and difficulty in operation, we only use 'marked' in R to co-estimate the capturing probability and survival rate with the MLE-based model. We preprocess data and do subsequent compu-

doi:10.6342/NTU202201298

tations with server population with Python instead.

4.1.1 Data Preprocessing

The original data collected by Wang is in a json format. For each live stream, it contains the streamer ID, start time, end time, region, list of transactions with time records and the IPs of video servers, and other stream information. In this study, we would like to use these transaction data to estimate server population over time.

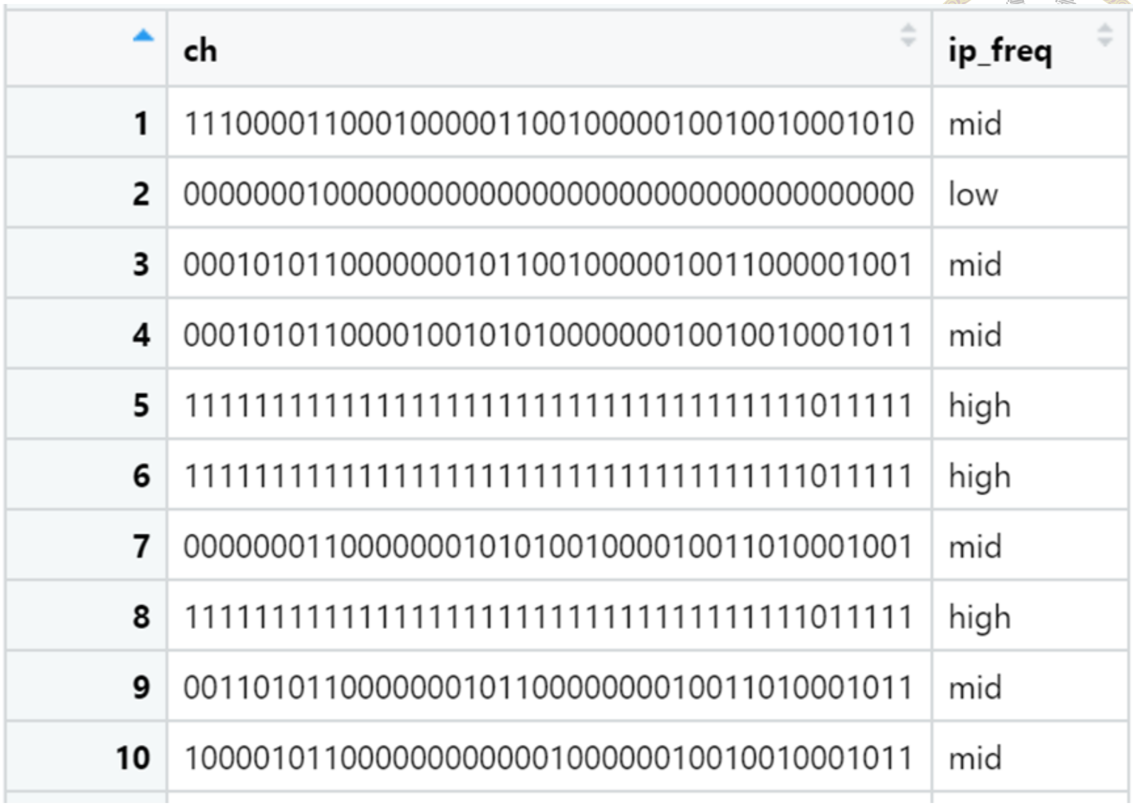
To fit the input format of ‘marked‘, we transform data into the following format with Python. Each row represents a capturing history of an IP address. ‘Ch‘ is a required feature showing the capturing records. 1 means the individual showing up in the sample; otherwise, it will be 0. An additional label can be added for data groups. **Figure 4.1** is an example of the input format.

4.1.2 Capturing and Survival Rate Estimation with Marked

First of all, given the capturing records of each server as the input data, the function ‘process.data‘ initializes several variables which are specific to the CJS model. Group labels, time intervals between samples, and animal initial ages can be defined in this function. In our cases, we will specify the individual label if we consider heterogeneity; otherwise, we will leave the three arguments blank. **Figure 4.2** is a screenshot example of the output data structure from ‘process.data‘.

Second, the function ‘make.design.data‘ creates a list of design data frame for model fitting. To be more specific, each parameter, such as capturing probability p and survival

doi:10.6342/NTU202201298



	ch	ip_freq
1	1110000011000100000110010000010010010001010	mid
2	00000001000000000000000000000000000000000	low
3	0001010110000000101100100000100110000001001	mid
4	0001010110000100101010000000100100100001011	mid
5	11111111111111111111111111111111111011111	high
6	11111111111111111111111111111111111011111	high
7	0000000110000000101010010000100110100001001	mid
8	11111111111111111111111111111111111011111	high
9	0011010110000000101100000000100110100001011	mid
10	1000010110000000000000100000010010010001011	mid

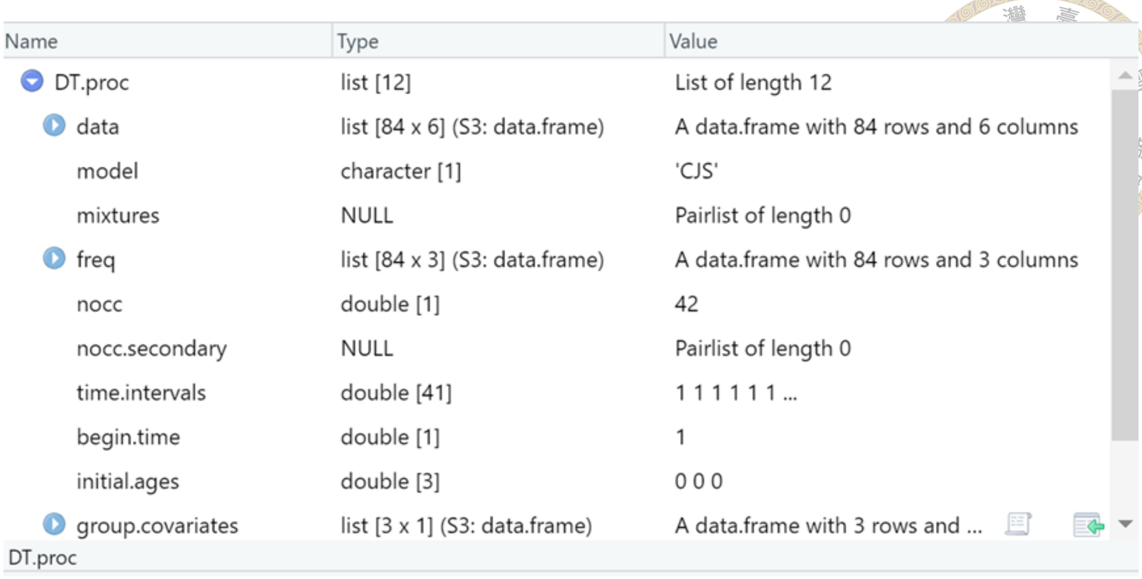
Figure 4.1: An Example of the Marked Input Format

rate Φ for CJS, has a corresponding data frame in the output list. **Figure 4.3** is a screenshot example of the output data structure from ‘make.design.data‘.

Next, we can use the ‘crm‘ function to fit a CJS model with the design data frame. Its goal is to optimize the term $1 + \frac{1}{e^{-Xb}}$ with Maximum Likelihood Estimation where X is the design matrix produced by ‘make.design.data‘ and b is the vector of parameters. After hundreds or thousands of iteration, the ‘crm‘ function prints lists of p and Φ values with lowest AIC. Since the MLE computation are on a logit scale, we need to change the estimation results from ‘crm‘ to the real values. Here we use ‘predict‘ function to do the transformation and we can get the real values of p and Φ.

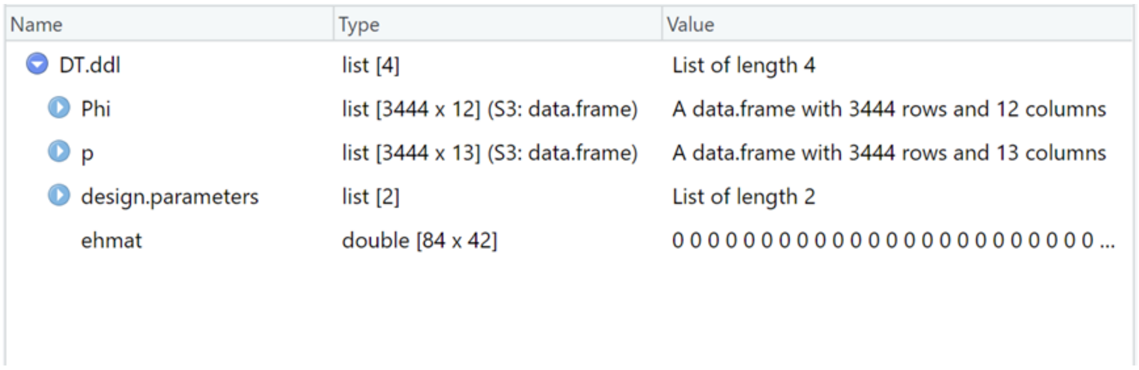
The output notations are shown as follows: ϕ_i is the survival rate from the ith sample to the $i+1$th sample. p_i is the capturing probability of the ith sample.

doi:10.6342/NTU202201298



```
Name                  Type                                 Value
DT.proc               list [12]                            List of length 12
  data                list [84 x 6] (S3: data.frame)       A data.frame with 84 rows and 6 columns
  model               character [1]                        'CJS'
  mixtures            NULL                                 Pairlist of length 0
  freq                list [84 x 3] (S3: data.frame)       A data.frame with 84 rows and 3 columns
  nocc                double [1]                           42
  nocc.secondary      NULL                                 Pairlist of length 0
  time.intervals      double [41]                          1 1 1 1 1 1 ...
  begin.time          double [1]                           1
  initial.ages        double [3]                           0 0 0
  group.covariates    list [3 x 1] (S3: data.frame)        A data.frame with 3 rows and ...
DT.proc
```

Figure 4.2: An Example of the Output from 'process.data'



```
Name                  Type                                 Value
DT.ddl                list [4]                             List of length 4
  Phi                 list [3444 x 12] (S3: data.frame)    A data.frame with 3444 rows and 12 columns
  p                   list [3444 x 13] (S3: data.frame)    A data.frame with 3444 rows and 13 columns
  design.parameters   list [2]                             List of length 2
  ehmat               double [84 x 42]                     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
```

Figure 4.3: An Example of the Output from 'make.design.data'

$$1 \xrightarrow{\Phi_1} \underset{p_2}{2} \xrightarrow{\Phi_2} \underset{p_3}{3} \xrightarrow{\Phi_3} \underset{p_4}{4} \xrightarrow{\Phi_4} \underset{p_5}{5} \xrightarrow{\Phi_5} \underset{p_6}{6} \xrightarrow{\Phi_6} \underset{p_7}{7} \tag{4.1}$$

Noted that the capturing probability at the first sample and the survival rate at the last sample are not provided by ‘marked‘. Since there is no reference point before the first sample, the capturing probability cannot be estimated. As for the last sample, no future record is given to estimate the survival rate.

doi:10.6342/NTU202201298

4.1.3 Server Population Estimation with Capturing Probability

With the numbers of servers and the capturing probability values estimated by 'marked', we can estimate the server population at each sample point. It is straightforward that server population (N_t) multiplied by capturing probability (PM_t) is the numbers of servers being captured (M_t). Therefore, the server population (N_t) at a specific sample time can be derived by dividing numbers of servers in the sample (M_t) by the capturing probability (PM_t), which can be referred as Eq. (2.2).

4.2 Offline Estimation

We first assume identical capturing probability and survival rate among all servers and compare the offline estimations on server population with the traditional CJS model and MLE-based CJS model. We evaluate the performances of these two models with an old 2019 data set used in our AINTEC paper [39] and a new data set from 5/07 to 5/15 in 2021. Wang [38] collected these two Twitch data sets with different methods. The 2019 data set was collected through Azure virtual machines, and the 2021 data set was collected by VPN and local docker containers. Hence, the time zones are different in these two data sets. In addition, Twitch rapidly expanded its CDN in these years. The server numbers grew from 197 in 2019 to 537 in 2021. The server assignment policies might also change as well, leading to different server patterns.

According to the AINTEC paper [39], we select the best parameter combination for the 2020 US data set to do the comparisons. In other words, the sampling duration and

doi:10.6342/NTU202201298

the inter-sample spacing are set to 1 hour and 24 hours respectively. Since CJS estimates capturing probability and survival rate based on the history records and future data, the first and last day estimation results may not converge yet. Hence, we only evaluate the average error rates of the estimations without these two days.

Sample Time	Traditional CJS Model	MLE-based CJS Model
0-1 am	0.4055	0.4071
6-7 am	0.2292	0.3205
0-1 pm	0.0725	0.0736
6-7 pm	0.2998	0.3044

Table 4.1: Average Estimation Error Rates for Two CJS Models with 2019 US data set

Sample Time	Traditional CJS Model	MLE-based CJS Model
0-1 am	0.0119	0.0082
6-7 am	0.3538	0.3538
0-1 pm	0.8170	0.8164
6-7 pm	0.5559	0.5581

Table 4.2: Average Estimation Error Rates for Two CJS Models with 2021 US data set

Table 4.1 and 4.2 show the average error rates for the traditional CJS model and the MLE-based CJS model with 2019 and 2021 US data set respectively. The high error rate of the MLE estimation results sampled at 6-7am in the 2019 data set is caused by the varying server numbers at the sample time. It indirectly contributes to the bad performance since MLE model co-estimate all parameters and cannot fit well with dramatic change of server numbers. Apart from the exception, the estimation results from these two models are relatively similar. Both share very closed error rates at four different sample time. Generally, there is no obvious difference between these two models under offline server population estimations.

However, sample time has a great impact on the estimation accuracy. The results can be associated with **Figure 3.2**, which shows the average server count by hour in the US data set. In this figure, more than 300 servers appear only from 23 pm to 5am, yet

doi:10.6342/NTU202201298

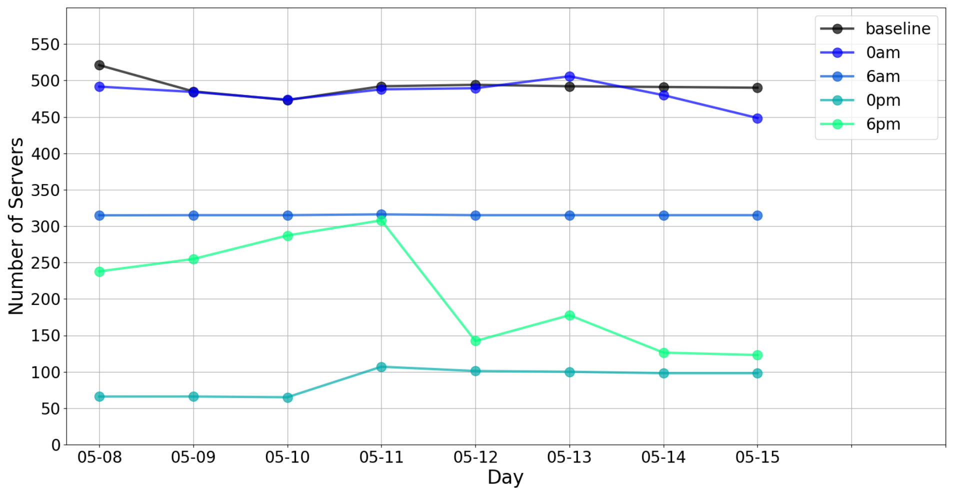


Figure 4.4: Offline Server Population Estimations at Four Sample Time with Traditional CJS Model

the number of servers remains low which are roughly 100 servers from 8am to 4pm in each day. Twitch's server allocation policy may directly influence the estimation results. Thus, the MLE server population estimation sampled at 0am with error rate 0.82% is more accurate than the results in other three sample time. The estimations from the traditional CJS have similar results as well.

Figure 4.4 and 4.5 respectively give closer looks to the server population estimations in each day with the traditional and MLE-based CJS models. The estimations on server population from both CJS models remain relatively stable for all of these sample time. We then speculate that these servers are divided into several groups. Twitch may have an internal policy to assign these groups of servers with different service schedules. Therefore, we would like to try some reasonable clustering methods and see if these server clusters lead to estimation results with lower error rates. The results and further discussions will be presented in **Chapter 5**.

doi:10.6342/NTU202201298

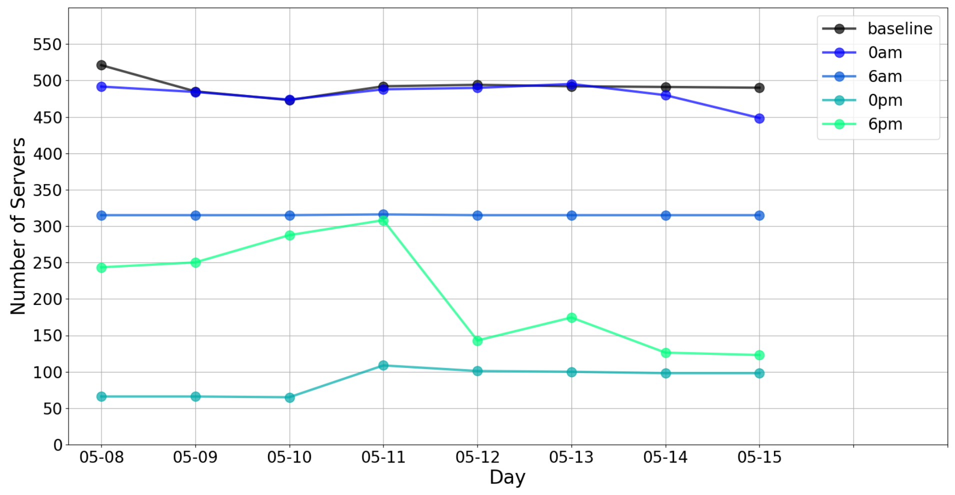


Figure 4.5: Offline Server Population Estimations at Four Sample Time with MLE-based CJS Model

4.3 Online Estimation

An important extension of server population estimation is to know the numbers of servers in real time. In this section we would like to evaluate the online estimation performances of these two CJS models.

In real time estimation, both CJS models do the estimations only based on the history and the current sample. For example, if we would like to estimate the server population on the fifth day, which is the blue block in **Figure 4.6**, we only make estimations based on the capturing history in the previous four days and the current sample on the fifth day. Similar computations are repeated to estimate the real time server numbers in each day. According to the experiment for real time estimation, the error rates of the traditional and MLE-based CJS models are 15.02% and 22.61% respectively. The real-time estimation results in each day are shown in **Figure 4.7**. The results indicate that both CJS models cannot converge without future data.

doi:10.6342/NTU202201298

Figure 4.6: Schematic Diagram of Real Time Estimation

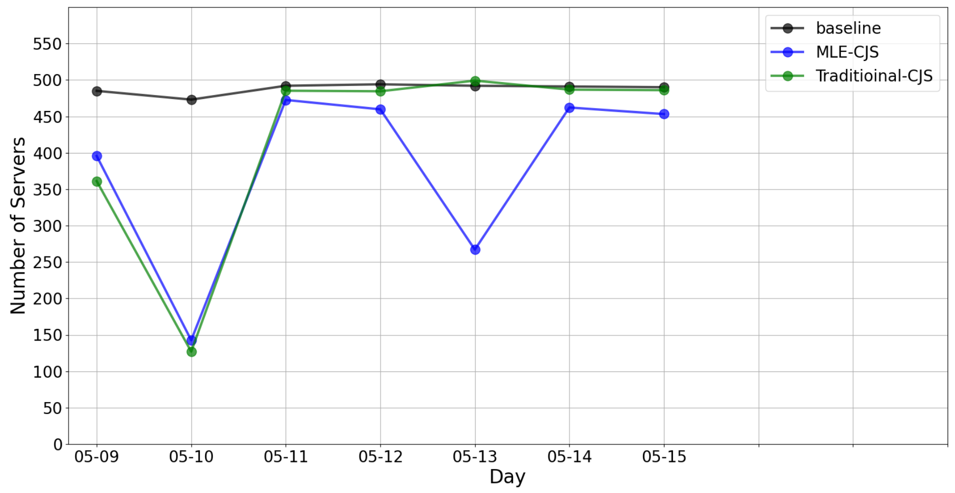


Figure 4.7: Real Time Estimation Results

For the traditional CJS model, we can recall the equations in **Section 2.2**. Both R_t and Z_t account for the chance of the individuals being recaptured in the future. Without the future data, R_t and Z_t are zeroes. These two terms influence the value of M_t in Eq. (2.6), indirectly causing inaccuracy estimation for N_t. For the MLE-based model, it co-estimate the capturing probability and survival rate for each sample point. However, the survival rate at the last sample point can not be estimated without future data. This indirectly influences the estimation of capturing probability at the last sample point, leading to the real time estimations with high error rates.

doi:10.6342/NTU202201298

4.4 Delayed Estimation

It is a big challenge for CJS model to do online estimations because it requires the history records and the future data to estimate the server population at a specific sample point. Sometimes online estimations may not converge, thus leading to inaccurate results. On the other hand, estimations with one-day delay may be acceptable for most applications. Considering this situation, we will compare these two CJS models for real time estimation and estimation with one-day delay. We concern if MLE-based CJS model with more complicated computations has shorter convergence time.

In estimation with one day delay, both CJS models do the estimations based on the history, the current sample, and the future data of one more sample. For example, if we would like to estimate the server population in the fifth day, which is the blue block in **Figure 4.8**, we make estimations based on the capturing history in the previous four days, the current sample in the fifth day, and the future data in the sixth day. By repeating the estimation process in each day, we can derive the server population at each sample point. This online estimation method significantly relies on the convergence speed of CJS models. Based on the experiment with 2021 US data set, the estimation error rates for the traditional and MLE-based CJS models are 5.04% and 3.64% respectively. The estimation results with one day delay are shown in **Figure 4.9**. These results show that the MLE-based CJS model has a shorter convergence period.

The results may be directly associated with the model structures. As mentioned in **Section 2.2**, traditional CJS model estimate server population highly relies on R_t and Z_t. These two terms directly influence the probability individuals are captured in the

doi:10.6342/NTU202201298

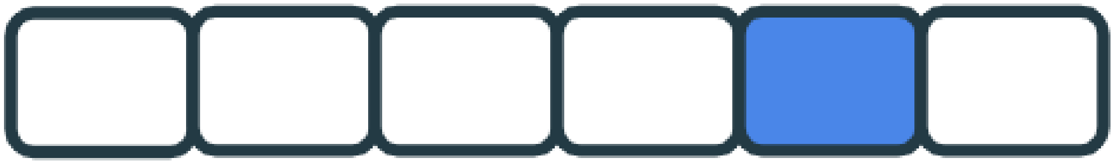


Figure 4.8: Schematic Diagram of Estimation with One Day Delay

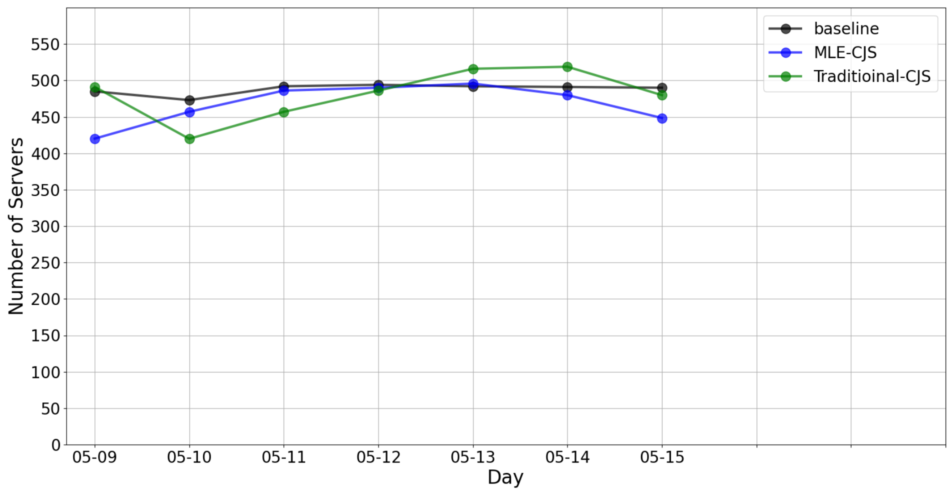


Figure 4.9: Estimation Results with One Day Delay

future. With future data for only one sample, the values of R_t and Z_t cannot be accurately estimated. On the other hand, MLE-based CJS model is able to compute both the capturing probability and the survival rate at the sample point with one sample delay. Therefore, the server population can be derived by the capturing probability with lower error rates.

Comparing the real time estimation results in **Figure 4.7** and estimation results with one day delay in **Figure 4.9**, the MLE-based CJS estimations in 5/10 and 5/13 are much different. It is closely related to its model structure as well. In fact, the numbers of servers captured in these two days are relatively low. In real time estimation, the model cannot distinguish whether these servers move out or are just not captured in the sample. Hence, the estimated capturing probability may far from the truth. For the estimation with one day delay, since these servers appear in the next sample again, the CJS model can estimate the capturing probability at a specific sample based on the future record. As a result, the

doi:10.6342/NTU202201298

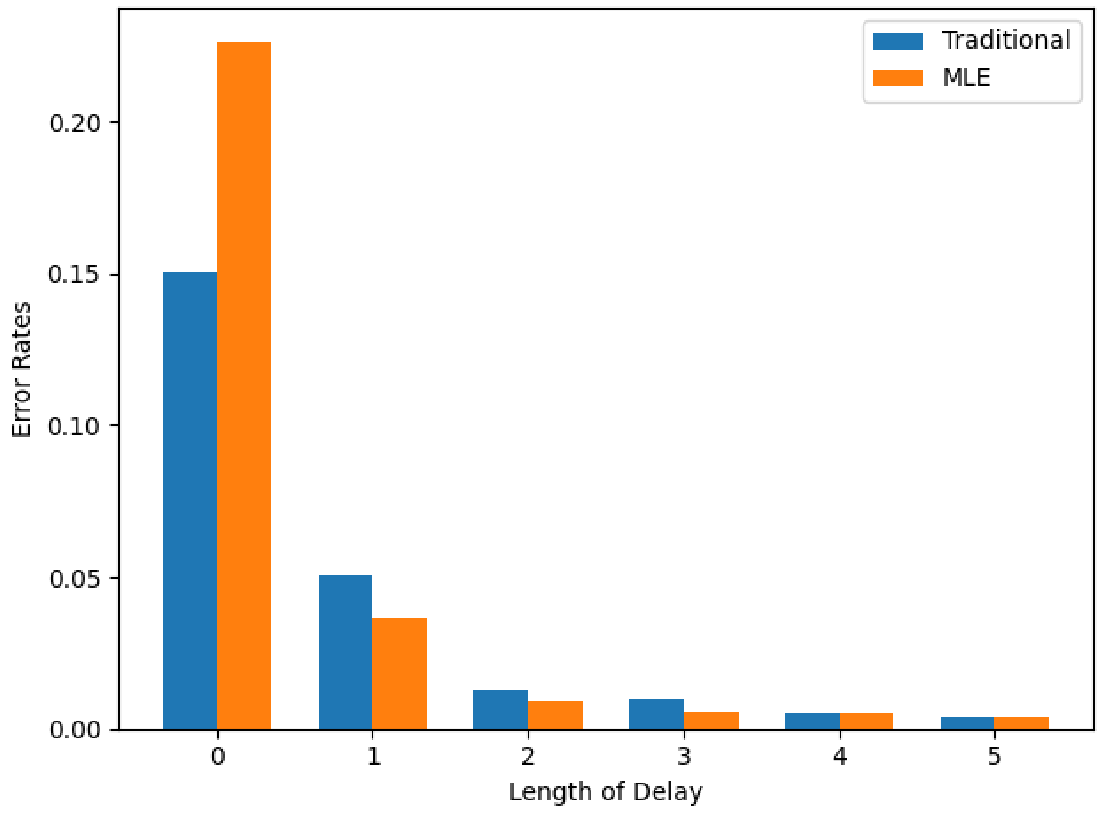


Figure 4.10: Average Estimation Error Rates with Delays

MLE-based model has a much better performance in 5/10 and 5/13.

4.5 Convergence Periods of the Two CJS Models

Figure 4.10 shows the average estimation error rates with different length of delays. Delay with zero sample equals to real time estimation. Likewise, delay with one sample represents the server population estimation delayed for one day. Comparing the average error rates of these two CJS model, we find that the MLE-based CJS model usually have a better performance given the same amount of future records. The results are related to the two CJS model structures, which have been discussed in **Section 4.4**. Moreover, as the estimation delay is longer, both server number estimations from these two models converge, and thus leading to lower estimation errors.

doi:10.6342/NTU202201298

Chapter 5

Clustering-based Experiment

We discover the hourly server count in **Section 3.3.1** seems to significantly change at different time. 0-2am is a peak while server numbers remain low. Besides, in the preliminary experiment, we find that the server population estimations remain stable at different sample time. We then speculate if these servers are divided into several groups according to Twitch's internal policy. Servers in the same groups may share similar patterns, yet servers in different groups have very different behaviors. Therefore, we try a couple of server clustering methods. We then assume that servers in the same group share the same capturing probability and survival rate. By estimating the population in each server group with MLE-based CJS models, we hope to improve the overall estimation accuracy. Furthermore, through different clustering strategies, we take a look at Twitch's internal policy for edge server assignments.

doi:10.6342/NTU202201298

5.1 Experiment Design

The clustering-based server population estimations contain two steps. First, servers are divided in to several groups in a reasonable way. Second, MLE-based CJS models will estimate capturing probability and the survival rate of each server group. Next, we can estimate server population with the capturing probability in each server group.

5.1.1 Server Clustering

The clustering strategies are highly related to the estimation results. Hence, we need to carefully evaluate if a server clustering strategy is reasonable. In this study, three clustering strategies are proposed.

5.1.1.1 Frequency of Occurrence

Figure 3.2 in **Section 3.3.1** shows that there is a peak during 0-2am with large variation and the server numbers stay low during 8am to 6pm with barely no variation. This may indicate that some servers are core servers. They are responsible for providing services all day long. In contrast to core servers, other servers are supportive. Some supportive servers only provide services when many clients are online and request for services every day, while other supportive servers only show up occasionally and may only serve clients when there are special activities on Twitch. Therefore, we come up with a naive clustering strategy: since we sample at 0-1am in each day (from 5/7 to 5/15, 9 days in total), if a server has ever showed up in more samples, it represents that this server almost stays online to serve clients. Here, we define frequency of occurrence as the number of samples

doi:10.6342/NTU202201298

(from 5/7 to 5/15, 9 samples in total) in which a server showed up for providing services. This parameter may be associated with the different server properties mentioned above. Therefore, we try to divide these servers into three groups according to their frequency of occurrence.

These servers are divided into three groups with a straightforward strategy as follows: Servers showing up more than 4 samples are labeled as servers with "high frequency." Servers showing up less than 2 samples are labeled as servers with "low frequency." Other remaining servers are labeled as servers with "mid frequency."

5.1.1.2 Server IP Prefix

Server IP are often related to their geo-location. Administrators are used to assign a group of servers with IPs sharing same prefixes. Therefore, servers with same IP prefix may have similar behaviors. In other words, servers in the same group will share the same capturing probability and survival rate, making the server population estimation more accurate. However, the difficulty is we do not know the exact server management policies. If the IP prefix is too short, the clustering will be too coarse. Servers in each group may not have similar behaviors. On the contrary, if the IP prefix is too long, the clustering will be too fine, thus leading to too many small groups. In case the number of servers are very low, the MLE-based CJS model may not fit and may even diverge. In this study, we try 16-bit and 24-bit prefix clustering.

doi:10.6342/NTU202201298

5.1.1.3 K-means Clustering with Transaction Numbers

Although IP prefixes are closely related to server behaviors, finer clustering may lead to divergence. Therefore, we would like to try another feature for server clustering. According to Wang's thesis [38], the crawler continuously sent requests to Twitch in order to discover all the server IPs providing services for the streaming channels. For each request, the Twitch's API returned only a server IP. Wang's crawling algorithm repeatedly sent requests until no new server IP showed up anymore. In other words, if a server IP were returned by Twitch's API with many times in the crawling process, it shows that this server was a core server and responsible for providing streaming services at the sample time. Here, we define the transaction number as the the server occurrences for multiple requests in a specific time period. By observing the transaction number in different time periods, it can form a multi-dimensional feature vector for each server. The transaction numbers of each server efficiently reflect server patterns. This parameter even provides more service information of each server compared with frequency of occurrence in **Section 5.1.1.1** and server IP prefix in **Section 5.1.1.2**. Since transaction numbers in different time periods are multi-dimensional vectors, K-means [17], an unsupervised clustering method, can be applied to cluster servers with the vectors of transaction numbers. In this way, servers in the same group share similar patterns of transaction numbers. Hence, it is more reasonable to assign servers in the same group with the same capturing and survival rates.

In our study, two-phase server clustering with K-means is used to classifying these servers. The first stage clustering strategy is developed by Hsu Cheng [19]. Clustering by transaction numbers in three different time periods in a day, servers in Twitch's CDN

doi:10.6342/NTU202201298

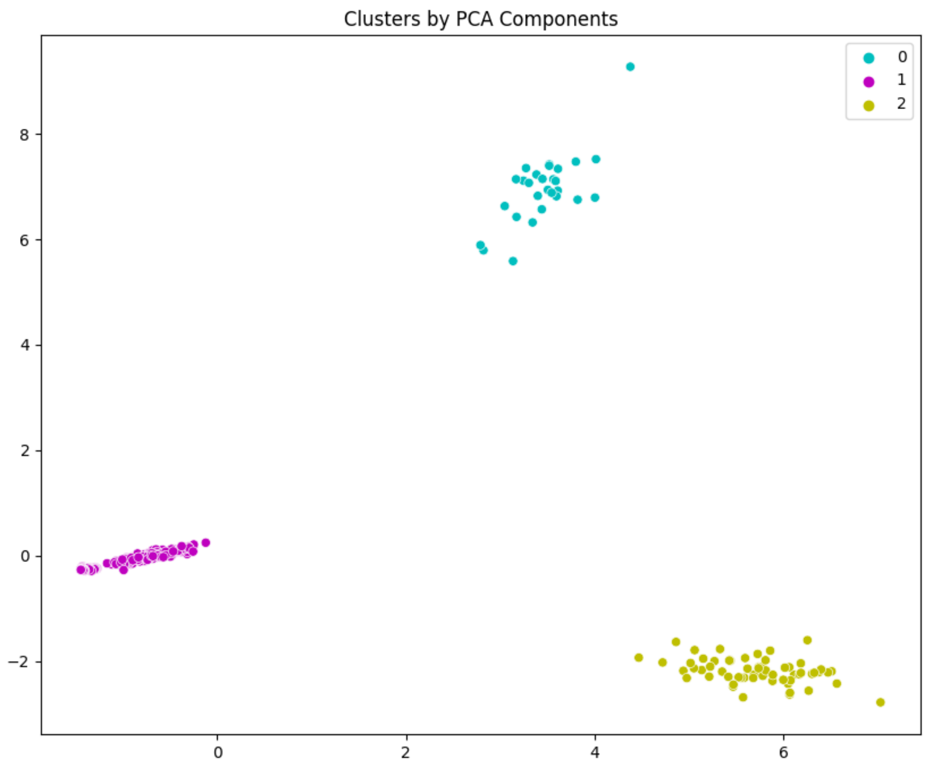


Figure 5.1: Clustering Results in the Second Phase

can be divided into three groups: orange, green, and blue. We will borrow this clustering result for server number estimation in **Section 5.2.3.1**.

There are 497 servers (more than 80%) in the blue server group. Nevertheless, servers in the blue server group occasionally appear the streaming systems and do not have a specific pattern. Thus, we attempt to do further evaluation within the blue group. This time we use the transaction numbers for an IP in each day (from 5/7 to 5/15, 9 days in total) as a 9-dimensional vector for clustering. For example, IP ‘52.223.227.169’ with transaction numbers [9283, 9765, 7412, 533, 731, 556, 706, 737] and IP ‘99.181.97.72’ with transaction numbers [125, 85, 4, 97, 76, 16, 57, 125] from 5/7 to 5/15. Then, we cluster servers in the blue group into three subgroups by K-means clustering. Using the vectors of the transaction numbers as the clustering feature, K-means randomly select 3

doi:10.6342/NTU202201298

vectors as the initial centroids. Then K-means will iteratively assign each server to one of the clusters based on its vector of transaction numbers and set the new centroid as the mean value of the transaction vectors in each cluster until the distance of the old and the new centroids in each cluster is less than a threshold. **Figure 5.1** shows the K-means clustering results with the transaction vectors projected onto a 2D surface.

In the second phase, we use the colors in **Figure 5.1** to represent the server groups. Servers in each cluster share similar patterns of transaction numbers and even have the same 24-bit IP prefixes in some clusters. There are 28 server IPs labeled as the cyan group. They have more transactions in the first few days and share the same 24-bit IP prefix '52.223.227'. Example: the server with IP '52.223.227.169' has the transaction records [9283, 9765, 7412, 533, 731, 556, 706, 737]. On the other hand, for the 61 servers labeled as the yellow group, they have more transactions in the last few days and share the same 24-bit IP prefix '192.16.65'. Example: the server with IP '192.16.65.56' has the transaction records [113, 25, 1677, 2484, 4408, 4540, 3230, 3063]. The remaining 408 servers (the magenta group) have few transactions, which shows that they seldom show up for providing services. Example: the server with IP '99.181.97.72' has the transaction record [125, 85, 4, 97, 76, 16, 57, 125].

5.1.2 Server Population Estimation

By dividing these servers into several groups, servers with similar patterns will share the same label. The label and capturing history for each individual forms the input for 'marked' for model fitting. Then 'marked' will co-estimate capturing probability and survival rate with the MLE-based CJS model. Next, with the capturing probability for

doi:10.6342/NTU202201298

each server group, we are able to estimate the server population in each group with the same method described in 4.1. By summing up the server population in each group, we obtain the overall server population at different sample time.

5.2 Experiment Results

We applied these three clustering strategies to server population in Twitch's CDN. The detailed observations of server clusters and estimation results are discussed in this section. Since we get the most accurate estimation sampling at 12 am in **Chapter 4**, we continue to run all the clustering-based experiments with the same sampling time.

5.2.1 Frequency of Occurrence

In the first part, we apply the frequency clustering method to the Twitch's dataset. Among the 538 servers in Twitch's CDN, there are 468 ”high frequency” servers, 45 ”mid frequency” servers, and 8 ”low frequency” servers. The overall server number estimation result in each day are shown in **Figure 5.2** with error rate 97%, which shows that the estimation does not converge.

Frequency	Server numbers	Average estimation error	Std
Total	538	0.9722	1.6218
High	468	0.0078	0.0101
Mid	45	0.1766	0.1574
Low	8	204.9497	402.5671
Not Appear	17	NA	NA

Table 5.1: Estimation Error Rates with Frequency Clustering

The average estimation error rates are shown in **Table 5.1**, and the estimation results of the three groups are shown in **Figure 5.3, 5.4, 5.5**, respectively. In fact, the estimation

doi:10.6342/NTU202201298

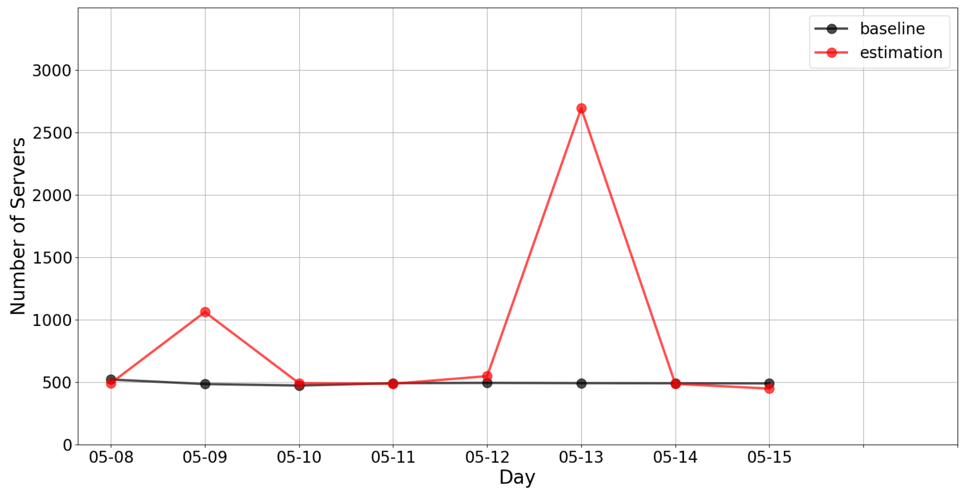


Figure 5.2: Server Number Estimation Result with Three Frequency Clusters

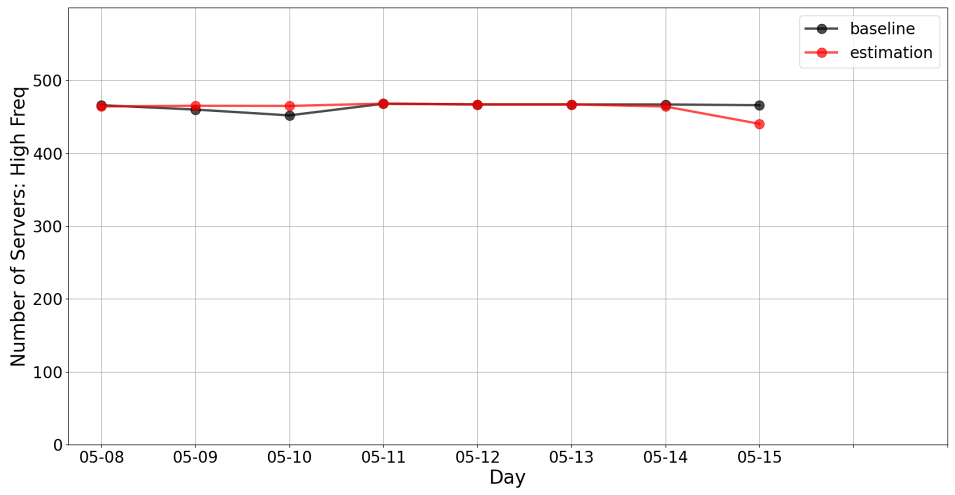


Figure 5.3: Server Number Estimation Result with ”High Frequency”

for ”high frequency” server group is relatively accurate due to the large population base and steady service patterns. On the other hand, the main reason for computation divergence is the low server numbers in ”mid frequency” and ”low frequency” server groups. These two server groups are too small, yet CJS are designed for estimating large animal population. In addition, the server numbers significantly vary from each sample point. These two factors make MLE estimation diverge, contributing to inaccurate capturing probability.

doi:10.6342/NTU202201298

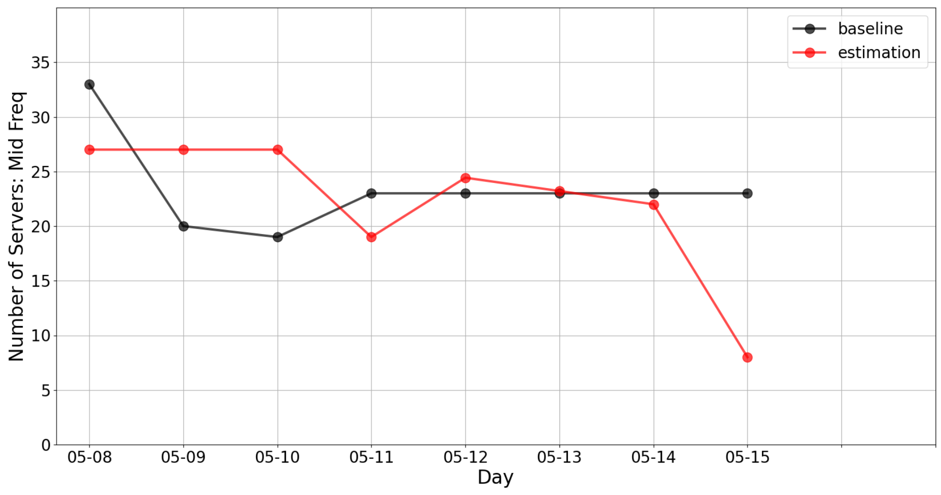


Figure 5.4: Server Number Estimation Result with ”Mid Frequency”

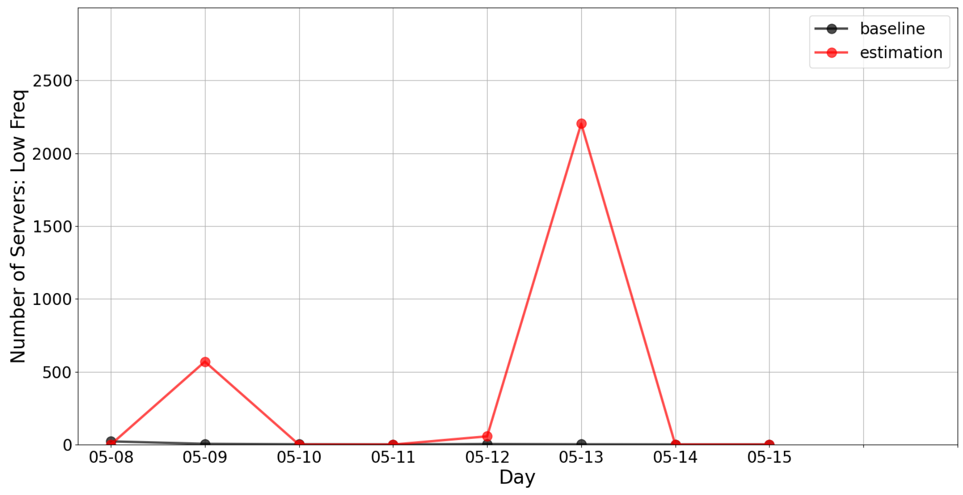


Figure 5.5: Server Number Estimation Result with ”Low Frequency”

Besides of computation divergence, the server clustering results with frequency clustering method are coarse. Most servers in ”mid frequency” and ”low frequency” do not share similar patterns. These cause higher variances in these two server groups, thus reducing the estimation accuracy. Furthermore, some servers do not show up during the samplings and are not classified into any of the server groups. These servers will not be considered into the server population by the CJS model. The reasons mentioned above all may lead to inaccurate estimation.

doi:10.6342/NTU202201298

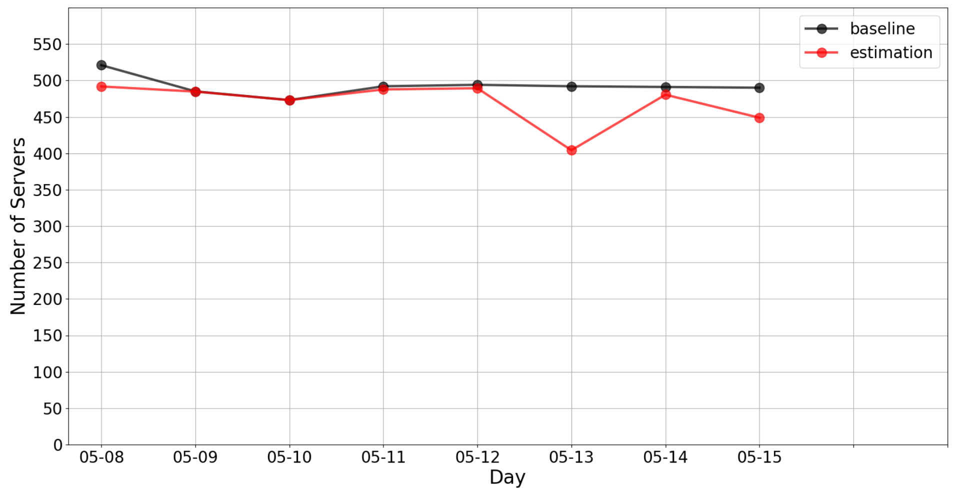


Figure 5.6: Server Number Estimation Result with Three 16-bit IP Prefix Clusters

5.2.2 Server IP Prefix

In this part, 16-bit and 24-bit IP prefixes are used to as the features for server clustering.

5.2.2.1 16-bit IP Prefix

16-bit IP prefix	Server numbers	Average estimation error	Std
Total	538	0.0365	0.0639
99.181	98	0.1704	0.3710
52.223	379	0.0111	0.0098
192.16	61	0.0015	0.0030

Table 5.2: Estimation Error Rates with 16-bit IP Prefix

Servers in Twitch's CDN are divided into three groups with 16-bit IP prefix clustering. Among all the servers, there are 98 servers with IP prefix "99.181", 379 servers with IP prefix "52.223", and 61 servers with IP prefix "192.16". The average estimation errors for each server group are listed in **Table 5.2**. The overall estimation error rate is 3.65%, which is worse than the estimation error rate 0.82% with no clustering. **Figure 5.6** shows

doi:10.6342/NTU202201298

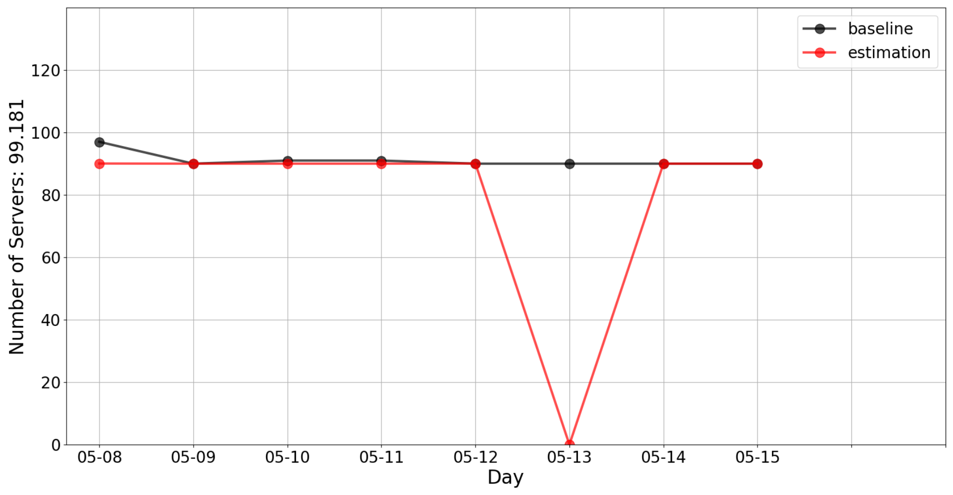


Figure 5.7: Server Number Estimation Result with IP Prefix ”99.181”

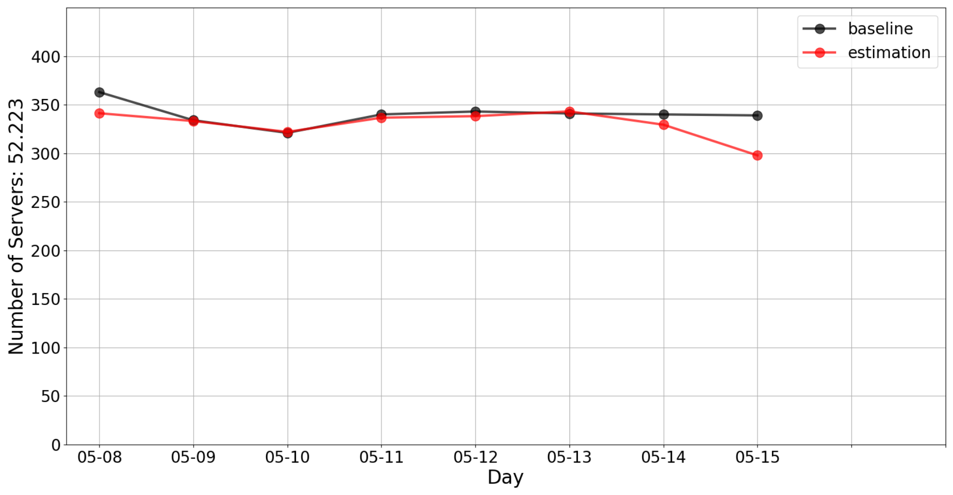


Figure 5.8: Server Number Estimation Result with IP Prefix ”52.223”

the total server number estimation result for each day by summing up the server number estimations in these three groups.

Server cluster ”99.181” is the one with highest estimation error rate. Due to measurement errors, the server number of the samples is zero in 5/13. Thus, according to the server estimation equation Eq. (2.2), the server population estimation was zero as well as shown in **Figure 5.7**. However, the server group with this IP prefix still appeared at other time in that day. That is why the estimation error is relatively high. As for the other

doi:10.6342/NTU202201298

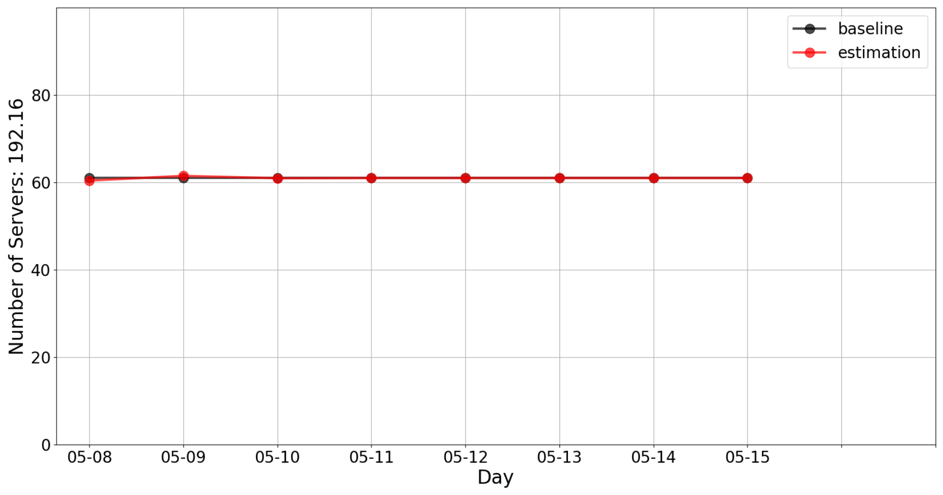


Figure 5.9: Server Number Estimation Result with IP Prefix ”192.16”

two clusters (**Figure 5.8**, **Figure 5.9**), the estimation results are relatively accurate. On the contrary, in the case without clustering, there are hundreds of servers caught at each sample. Even if the server numbers vary from each sample point, the MLE-based CJS model can still give relatively accurate capturing probabilities with low error.

Although the estimation results are relatively close to the baselines in each group, Cheng [19] still discovered that some servers with the same 16-bit IP prefixes have different service patterns. This shows that Twitch may manage these servers with more small groups, and make all servers in each group share similar patterns, rather than the three 16-bit-IP-prefix clusters. Therefore, we would like to try 24-bit IP prefix clustering instead.

5.2.2.2 24-bit IP Prefix

Next, we apply 24-bit IP prefix clustering on the Twitch data set. These servers are divided into 15 groups in total. By estimating the server number in each group and summing all these results, the estimation results are shown in **Figure 5.10**. The overall estimation error is 7.74%, which is still worse than the one with no clustering 0.82%. The

doi:10.6342/NTU202201298

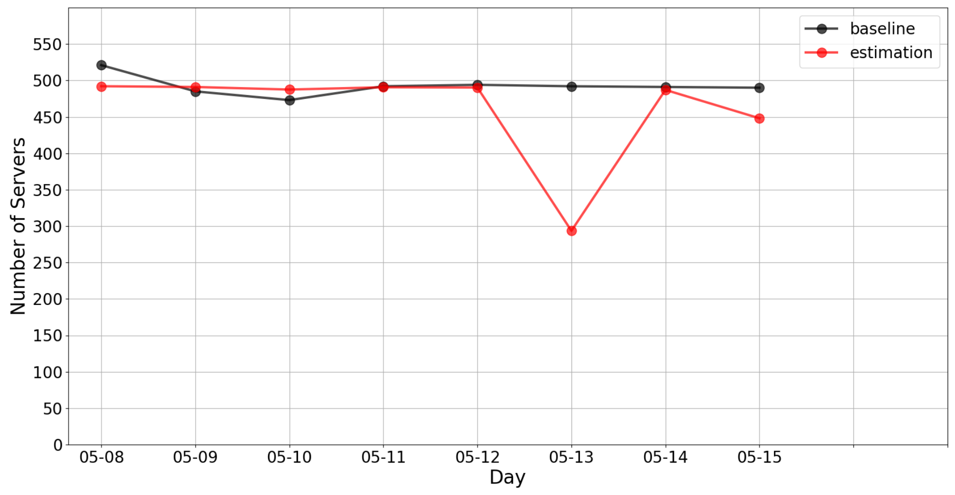


Figure 5.10: Server Number Estimation Result with Fifteen 24-bit IP Prefix Clusters

24-bit IP prefix and their corresponding server numbers and estimation errors are listed in **Table 5.3**.

24-bit IP prefix	Server numbers	Average estimation error	Std
Total	538	0.0774	0.1459
99.181.96	16	0.1667	0.3727
52.223.226	68	0.1667	0.3727
99.181.97	81	0.1709	0.3708
52.223.244	41	0.1667	0.3726
52.223.227	31	0	0
52.223.224	19	0	0
52.223.229	12	0	0
52.223.246	35	0.0049	0.0110
52.223.228	39	0	0
192.16.65	61	0.0014	0.0030
52.223.243	97	0.0732	0.0798
52.223.225	34	0.0058	0.0129
99.181.65	1	0.1250	0.3536
52.223.247	2	0.1250	0.3536
52.223.248	1	0.1250	0.3536

Table 5.3: Estimation Error Rates with 24-bit IP Prefix

It is worth noticing that the server groups with IP prefix ”99.181.96”, ”52.223.226”, ”99.181.97”, ”52.223.244” have relatively high estimation error rate because the server numbers of the sample in 5/13 are all zeros. However, servers with these IP prefixes

doi:10.6342/NTU202201298

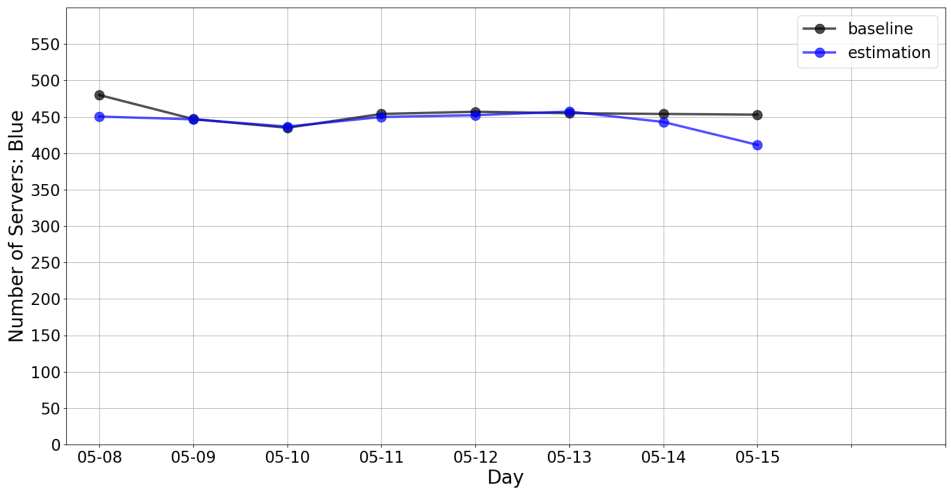


Figure 5.11: Server Number Estimation Result with the Blue Group

show up at other time period in that day, so the baseline shows that these servers were in service in 5/13. Furthermore, for servers with IP prefixes ”99.181.65”, ”52.223.247”, ” 52.223.248”, these groups contain only 1-2 servers. These 4 servers did not show up in any of the captures. Due to the finer IP prefix clustering, these servers were neglected by the CJS models. These two are main reasons of the relatively high error rates. Although servers in each group have similar behaviors, finer clusters may make the CJS model not fit well and magnify the estimation errors.

5.2.3 K-means Clustering with Transaction Numbers

We will evaluate the estimation results with the two phases K-means clustering.

5.2.3.1 Clustering by Transaction Numbers in Different Time Periods

Borrowing from Cheng’s clustering strategy [19], these servers can be divided into three groups: the blue group, the orange group, and the green group according to the Twitch’s servers by transaction numbers in three time periods in a day. The estimation

doi:10.6342/NTU202201298

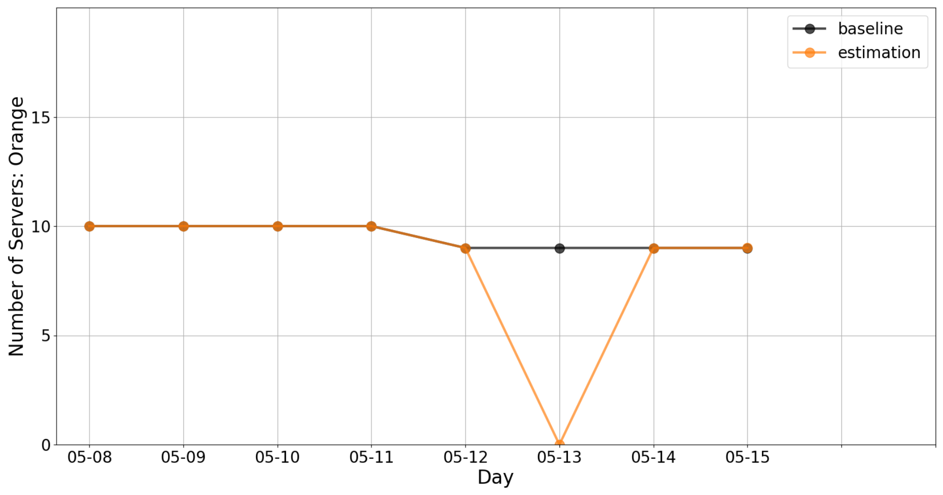


Figure 5.12: Server Number Estimation Result with the Orange Group

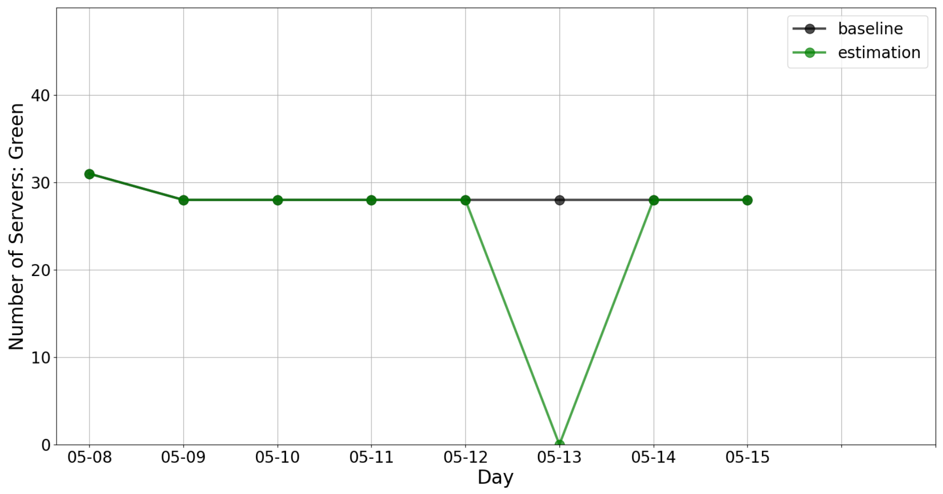


Figure 5.13: Server Number Estimation Result with the Green Group

Group	Server numbers	Average estimation error	Std
Total	538	0.0192	0.0240
Blue	497	0.0089	0.0076
Orange	10	0.1667	0.3727
Green	31	0.1667	0.3727

Table 5.4: Estimation Error Rates with K-means Clustering Phase 1

results in each group are shown in **Figure 5.11, 5.12, 5.13**. By summing up the server numbers in these three group, we can derive the total server numbers, as shown in **Figure 5.14**. The overall error rate is 1.92%, and the average error rates for these three groups are listed in **Table 5.4**.

doi:10.6342/NTU202201298

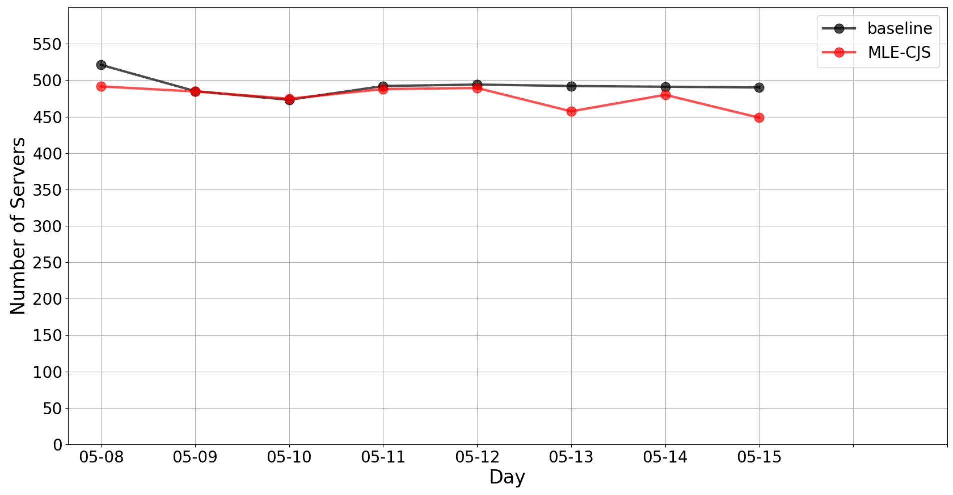


Figure 5.14: Total Server Number Estimation Result in the K-means Clustering Phase 1

According to Cheng's evaluation [19], servers in the orange group and the green group are almost in service with fixed schedules in each day. Thus, the estimation results of the two groups fit the baselines very well except in 5/13. The estimation server numbers in 5/13 were all zeros because none of servers in the two groups were in the catch during 12am to 1am (measurement errors). However, servers in these two groups still stably appeared in other time of that day. This is the main cause of the overall estimation error. Despite a relatively low estimation error for the blue server group, its server population accounts for more than 80% and content servers in this group do not show up periodically in each day and have no regular patterns. To understand more about these supportive servers in Twitch's CDN, we do further clustering and evaluation on the blue server group in K-means clustering phase two.

5.2.3.2 Clustering by Transaction Numbers in Different Days

In stage two, we further evaluate the blue server group according to the transaction numbers in different days. These servers can be divided into three groups: the cyan group,

doi:10.6342/NTU202201298

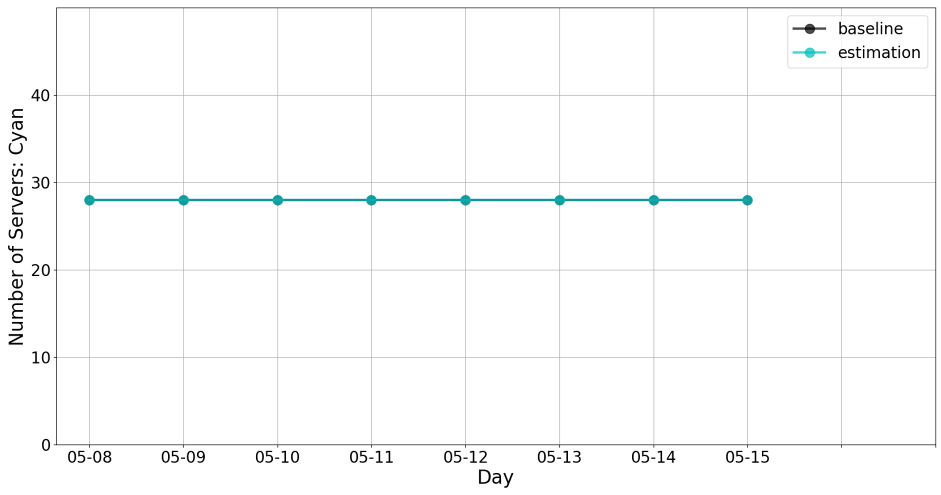


Figure 5.15: Server Number Estimation Result with the Cyan Group

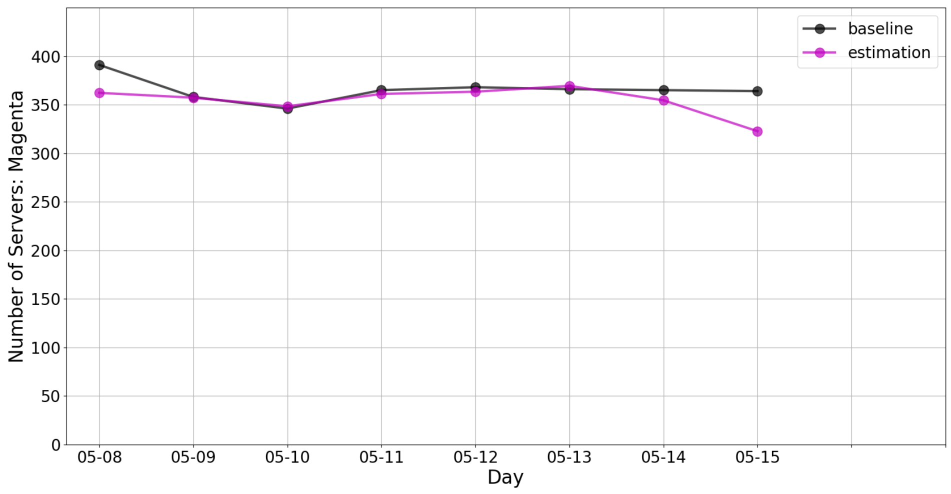


Figure 5.16: Server Number Estimation Result with the Magenta Group

Group	Server numbers	Average estimation error	Std
Total	538	0.0187	0.0230
Orange	10	0.1667	0.3727
Green	31	0.1667	0.03727
Cyan	28	0	0
Magenta	408	0.0117	0.0083
Yellow	61	0.0014	0.0030

Table 5.5: Estimation Error Rates with K-means Clustering Phase 2

the magenta group, and the yellow group. The server estimation results in these three groups are shown in **Figure 5.15, 5.16, 5.17**. With the orange and green groups in stage one and these three new groups in stage two, we estimate the total server number in each

doi:10.6342/NTU202201298

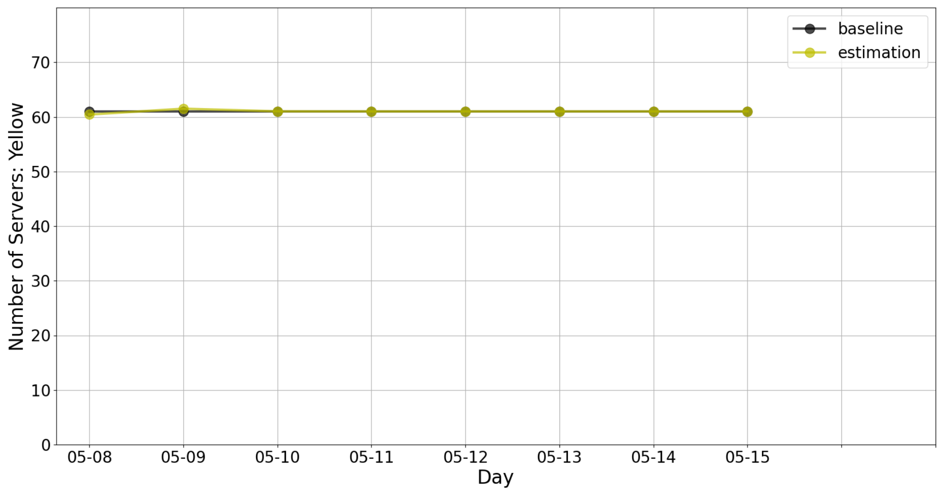


Figure 5.17: Server Number Estimation Result with the Yellow Group

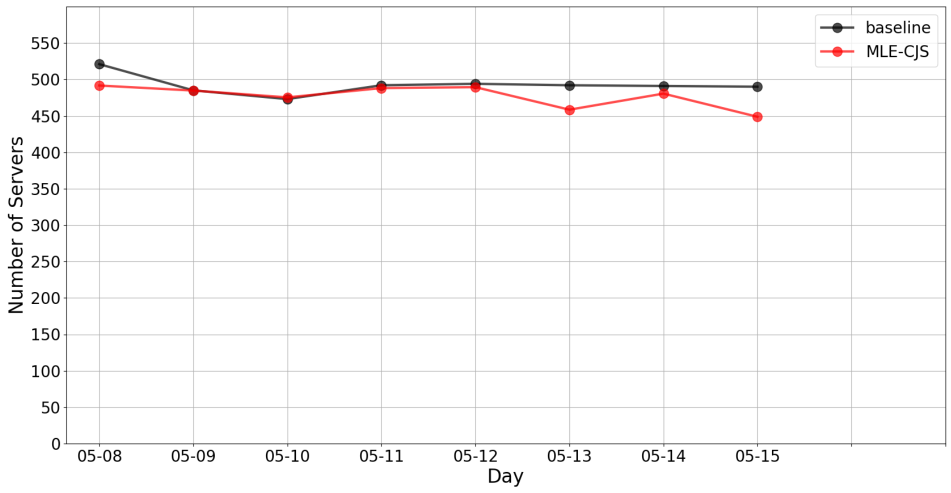


Figure 5.18: Total Server Number Estimation Result in the K-means Clustering Phase 2

(shown in **Figure 5.18**) and the overall estimation error is 1.87%. The average estimation errors are listed in **Table 5.5**.

The estimation results of the orange and green group have been discussed in 5.2.3.1. For the three new clusters in phase 2, both the cyan and yellow groups fit the baseline very well, verifying that servers in each group share very similar behaviors. As for servers in the magenta group, their transaction numbers are very low and do not have regular patterns. These servers are served as supportive ones and only appear if needed. Servers in this

doi:10.6342/NTU202201298

group have large variance in service patterns and account roughly 75% of the total server numbers, and thus leading to a relatively high estimation error.

5.3 Discussion

In this section, three issues are discussed. One is related to higher estimation error rates in some groups, another is is related to the measurement error in 5/13, and the other is the computational overhead of the MLE-based CJS model.

5.3.1 High Estimation Error Rates in Some Groups

In **Section 5.2.3.1 and 5.2.3.2**, the blue server group and the magenta server group have the estimation error 0.89% and 1.17% respectively. Servers in these two server groups do not share similar patterns. Note that the estimation error without heterogeneity is 0.82%. We can treat the estimation result without heterogeneity as the server estimation of the whole server group. It is worth noticing that the blue group is split from the whole server group, and error rate of the blue group is slightly higher than that of the whole group. In addition, the magenta group is also split from the blue group, and the error rate of the magenta group is higher than that of the blue group as well. One reasonable explanation is that when some servers sharing some similarities are removed, the variance of the remaining server patterns becomes higher, and thus leading to a higher estimation error rate.

doi:10.6342/NTU202201298

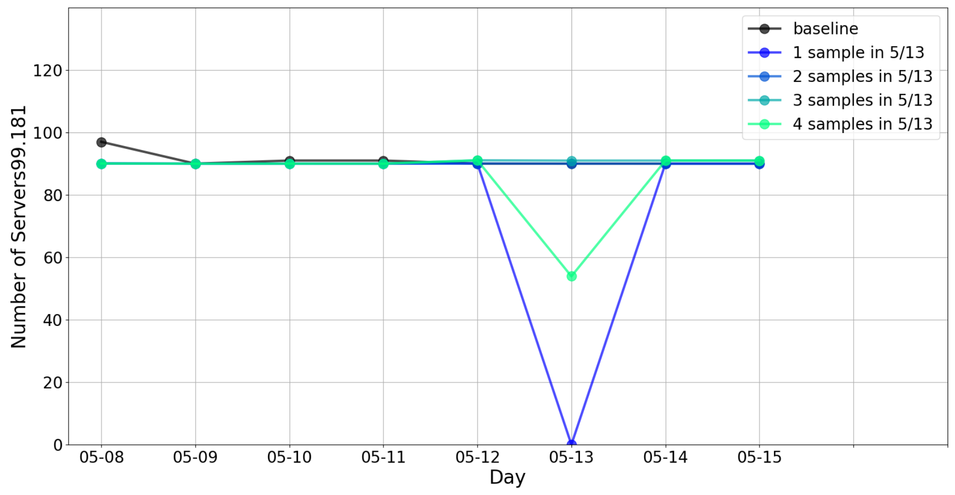


Figure 5.19: Server Number Estimation Result with 16-bit IP Prefix ”99.181” (Taking 2-4 Samples in 5/13)

5.3.2 No Captures in Some Samples

In **Figure 5.7, 5.12, 5.13**, the server number estimations fit the baselines well except 5/13. Due to measurement errors, the numbers of the servers caught from 12am to 1am on that day dramatically drop. In some groups, even no servers are caught during the 5/13 sampling, causing the server estimation to be zero. In fact, these servers still appear at other time of that day, thus contributing to a high estimation error rate. On the other hand, for the estimation without clustering, all servers are assumed to have the same capturing probability. As a result, even if some servers are lost in 5/13 and show up in later samples, ‘marked‘ still can estimate the capturing probability based on the number of the remaining servers captured in 5/13 sample. If the abnormal effect can be detected and compensated, the clustering-based estimation result may perform better than the one without heterogeneity.

In this study, we try to compensate the situation of no captures by sampling more times in 5/13. For sampling twice in 5/13, we sample at 0-1 am and 0-1 pm; for sampling

doi:10.6342/NTU202201298

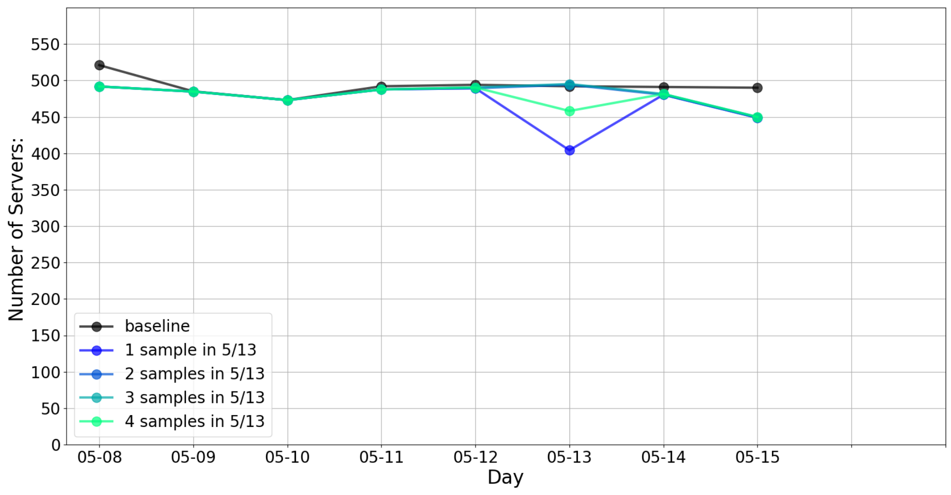


Figure 5.20: Server Number Estimation Result with Three 16-bit IP Prefix Groups (Taking 2-4 Samples in 5/13)

three times, we sample at 0-1am, 8-9am, and 4-5pm; for sampling four times, we sample at 0-1 am, 6-7am, 0-1 pm, and 6-7pm. Next, since the sampling space in 5/13 is different from other days, we need to set the corresponding sampling intervals in ‘marked‘. Then, we will use the mean value of the estimation results from these several samples (with captures > 0) as the estimation result in that day.

For the group with 16-bit IP prefix ”99.181”, the estimation results with more samples in 5/13 are shown in **Figure 5.19**, and the overall estimation results for the three 16-bit IP prefix groups are shown in **Figure 5.20**. In the case of 2 and 3 samples in 5/13, both of them fit the bas eline very well. As for the case of 4 samples, the numbers of servers being captured differ greatly in the three samples (6-7am, 0-1pm, and 6-7pm), causing the average estimation result far from the baseline in 5/13. The detailed estimation errors for group ”99.181” and the overall estimations are listed in **Table 5.6**. Note that both the overall estimation errors with 2 and 3 samples in 5/13 (0.75% and 0.72%) are lower than the error without clustering 0.82%. This indicates that clustering-based estimation can

doi:10.6342/NTU202201298

even perform better than estimation without clustering.

Sampling Times	Group ”99.181”	Overall
1	0.1704	0.0366
2	0.0038	0.0075
3	0.0093	0.0072
4	0.744	0.0176

Table 5.6: Average Estimation Errors with Three 16-bit IP Prefix Groups (Taking 2-4 Samples for Group ”99.181” in 5/13)

Similar operations can be applied to the estimations of the orange and green groups in the estimations with K-means clustering as well. Once we discover any abnormal effect, we can sample more times in that day for estimation. Although the sampling overhead slightly increases, the overall estimation accuracy significantly improved. Note that the numbers of extra samples required for the estimations vary from different platforms and data sets.

5.3.3 Computational Overhead of MLE-based CJS Model

Maximum Likelihood Estimation is a method to search model parameters fitting input data. This operation usually updates parameters with hundreds or thousands of iterations. Hence, compared with traditional CJS, MLE-based CJS requires more computation resources. In this study, a server with Intel Core i9-9900k CPU is used to measure the computation time.

Clustering Strategy	Estimation Error	Num of Params	Computation Time (Sec)
No Heterogeneity	0.0082	18	0.4325
Freq of Occurrence (3Groups)	0.9722	54	1.7431
16-bit IP Prefix (3 Groups)	0.0365	54	1.8937
24-bit IP Prefix (15 Groups)	0.0774	216	29.6354
Kmeans Phase 1 (3 Groups)	0.0192	54	1.6284
Kmeans Phase 2 (5 Groups)	0.0187	90	4.5393

Table 5.7: Computation Time and Estimation Errors for Each Clustering Strategy

doi:10.6342/NTU202201298

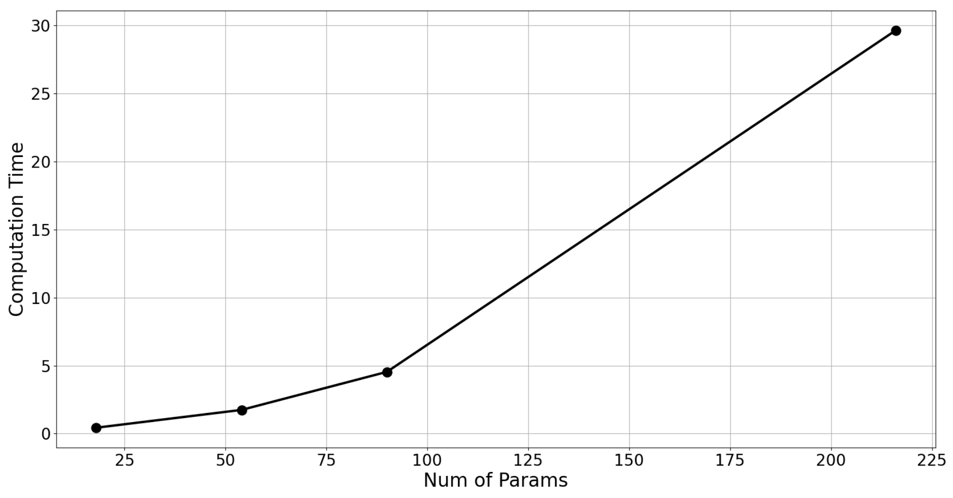


Figure 5.21: Comparison of Computation Time and the Number of Model Parameters

Table 5.7 shows the computation time and numbers of parameters for each clustering strategy. Numbers of Parameters are proportional to numbers of server groups and numbers of samples. The only exception is 24-bit IP prefix. As mentioned in **Section 5.2.2.2**, none of the servers in the 3 groups show up in any samples. Thus, only servers in the other 12 groups will be considered by the MLE-based CJS model.

As **Figure 5.21** shows, computation time increases faster than the numbers of model parameters. This is the main reason we use clustering strategies to reduce the model complexity. If we assign each server with different capturing probability and survival rate, the run time will be much longer and the MLE computation may even diverge. On the other hand, estimation errors may not reduce with the growth of group numbers. Take the estimation with 24-bit IP prefix as an example. The server numbers in some groups are too small if there are too many server groups. While estimating survival and capturing rate in each group, the MLE-based CJS cannot fit the small amount of input data well, leading to high estimation errors.

Selecting a clustering strategy is extremely important for server population estimation

doi:10.6342/NTU202201298

with heterogeneity. The key point influencing the overall estimation accuracy is whether a server clustering result really matches its CDN server assignment. By observing the server patterns, we discover that Twitch's servers seem to be classified as a core server group or a supportive server group. Core servers almost stay online and share very similar behaviors. Thus, the error rate of the core server group is usually lower than the estimation error rate without heterogeneity. However, core servers only account for a small proportion in Twitch's video servers. In fact, most servers in Twitch's CDN are supportive ones. These servers occasionally appear to provide services and may not have regular patterns. Thus, the estimation error basically comes from the population estimation of the supportive server group. To reduce the estimation error for supportive servers, it is a reasonable way to cluster the servers with similar characteristics such as IP prefix or transaction numbers in several time periods. Estimations on the server group sharing similar patterns also have lower estimation errors on average.

According to our observation, Twitch's servers have roughly three to five different patterns. Server IP prefixes and transaction numbers are useful features to divide server into these groups. Hybrid model for clustering such as the random forest [4] may work as well. Furthermore, three to five are also suitable cluster numbers for 'marked'. The computational overhead is acceptable and the estimations for capturing probability and survival rate can converge well in most cases. If one wants to apply the clustering-based server population estimation in other CDN, the most suitable cluster numbers may slightly change. Yet, we consider that the clustering principle still maintains.

doi:10.6342/NTU202201298

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This research aims to improve the server estimation accuracy with Maximum-Likelihood-Estimation-based CJS model. This model derives the values of capturing probability and survival rate at each sample by solving the CJS probability model with MLE. In addition, the MLE-based CJS model with heterogeneity relaxes the assumption that all individuals have the same capturing probability and survival rate. However, if the parameters of each individual are all independent, the resulting MLE problem is of extremely high dimension and may not converge. Hence, we try several clustering strategies to divide servers into several groups. In each group, servers with some similar behaviors will share parameters in the CJS probability model. As a result, we can reduce the parameter space and estimate the survival and capturing rate per cohort.

Marked, an implementation of the MLE-based CJS model with heterogeneity, is used for server number estimation in this study. Using a data set collected in May 2021, we

doi:10.6342/NTU202201298

compare two CJS models in offline/online estimation. Furthermore, we try a variety of clustering strategies such as IP prefix, server working frequencies, and K-means to evaluate if MLE-based CJS with heterogeneity will improve the estimation results.

In **Chapter 4**, we first compare the traditional CJS and MLE-based CJS in offline/ online estimation without heterogeneity. In the offline estimation, both CJS models estimate server numbers based on capturing history and future records. Even though the traditional CJS and MLE-based CJS estimate server population under different probability assumptions, there is no significant difference between the offline estimation results of the two CJS models. Furthermore, we try real time estimation and estimation with one day delay. Both CJS models cannot converge well under real time estimation because they both cannot estimate population without future records. As for estimation with one day delay, MLE-based CJS has a lower estimation error because it can estimate the capturing probability and survival rate at the sample point with one sample delay. This indicates that MLE-CJS has a shortor convergence period.

In **Chapter 5**, we evaluate if MLE-based CJS with heterogeneity can improve the estimation accuracy with three clustering strategies. For server clustering with frequency of occurrence, the size of group ”mid frequency” and ”low frequency” are too small for MLE-based CJS to estimate group population. The values of capturing probability at some samples diverge, leading to an extreme high error rate. Another straightforward clustering method is to cluster servers with IP prefixes. In the case of 16-bit IP prefix, there are three server groups. The estimation error rate is relatively high because the server number in the sample in 5/13 is zero. This cause the population estimation in that sample to be zero. In addition, not all servers in one group have similar patterns, which shows that the clustering

doi:10.6342/NTU202201298

is too coarse. In the case of 24-bit IP prefix, there are fifteen server groups. Two reasons contribute to high estimation errors. One is the server numbers are zeros in the sample in 5/13. The other is that none of servers in some small server groups have ever appeared in any of the samples, and they are neglected in the server population estimation. The last clustering strategy we try is K-means clustering. We find that the estimation results in each group fit the baseline well. However, for some groups, the server numbers are still zeros in sample in 5/13, leading to a relatively high error rate. If the problem of zero server being caught at some sample points can be compensated such as doing extra sample in that day, clustering-based estimation may perform better than estimation without heterogeneity.

6.2 Future Work

Our work currently focuses on server population estimation with the MLE-based CJS model. We apply our methodology to the Twitch's US data in November 2019 and May 2021 and evaluate the estimation results. Looking forward to the future, we would like to introduce deep learning to estimate the CDN size. Current CJS model estimates population based on capturing history. In other words, it is a prediction process with long sequences of input data. LSTM(Long Short-Term Memory) [18] is one of the RNN(Recurrent Neural Network) implementations. Besides the original hidden states in neural networks, LSTM adds extra cell states in order to maintain long-term memory. A forget gate, an input gate, and an output gate are the key elements in state transitions. A forget gate determines if old information in a cell state should be forgotten. An input gate, on the other hand, controls the amount of new information added to a cell state. At last, an output gate decides the output of an hidden gate. In general, these three parameters in each pair of hidden

doi:10.6342/NTU202201298

and cell state are determined based on the input data sequence. As a result, multinomial probability assumptions on individuals are further relaxed for population estimation. It is then expected to apply LSTM or other recurrent neural networks to estimating server population at each sample. Nevertheless, during the training process, neural networks iteratively update model parameters with a large amount of data. Hence, consecutive data sets with longer periods are required to train the machine learning models.

Furthermore, other machine learning techniques can be applied to find the best clustering strategy for each data set as well. At the current stage, we try several naive and K-means clustering strategies, yet none of them significantly improve the estimation accuracy. K-means, an unsupervised learning method, clusters individuals according to the Euclidean distances of feature vectors. Since not all the features are equally correlated to server clustering, manual feature screening are still required before running K-means. However, potential useful features may be neglected in the feature selection process. Random forest algorithm [4] may be a possible solution to addressing this issue. Without assigning the outcome variable, the random forest will be run in unsupervised mode. During the clustering process, several hierarchical trees are built and the distances between individuals are roughly estimated based on the proportion of times these individuals end up in the same leaf nodes. To search the best clustering result, the tree structures are adjusted within several iterations. In different branches of hierarchical trees, different features are automatically selected for clustering. Hence, no potential features will be sacrificed before the random forest clustering. If we apply the clustering-based server population estimation to a new data set, the random forest may assist us to find the best clustering result by fitting the input data.

doi:10.6342/NTU202201298

Another direction this study is to discuss the relationship between estimation accuracy and the overall system performance. As mentioned in **Chapter 1**, server population estimation with lightweight probing is the first phase to detect anomalies in Internet-scale services. In the future work, it is expected to build up a complete system for CDN measurement in different geographic regions and evaluate how the estimation errors influence the following applications. In addition, it is worth noticing that the server assignment policies may vary in different regions. To accurately estimate the server population, we may need to adjust the sampling strategies (including the lengths of the sampling space and the sampling duration) in different regions. The system integration is also an important issue to be focused in the future.

doi:10.6342/NTU202201298

References

[1] H. Akaike. Information Theory and an Extension of the Maximum Likelihood Principle, pages 199–213. Springer New York, New York, NY, 1998.

[2] D. Borchers and M. Efford. Spatially explicit maximum likelihood methods for capture-recapture studies. Biometrics, 64:377–85, 07 2008.

[3] T. Böttger, F. Cuadrado, G. Tyson, I. Castro, and S. Uhlig. Open connect everywhere: A glimpse at the internet ecosystem through the lens of the netflix cdn. SIGCOMM Comput. Commun. Rev., 48(1):28–34, Apr. 2018.

[4] L. Breiman. Random forests. In Machine Learning, pages 5–32, 2001.

[5] C. Brownie and D. S. Robson. Models allowing for age-dependent survival rates for band-return data. Biometrics, 32(2):305–323, 1976.

[6] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pages 108–122, 2013.

doi:10.6342/NTU202201298

[7] K. P. Burnham. Design and analysis methods for fish survival experiments based on release-recapture / Kenneth P. Burnham ... [et al.]. American Fisheries Society Bethesda, Md, 1987.

[8] Chase. State of the stream april/2020: Valorant and valorant streamers top the charts, May 2020.

[9] R. M. CORMACK. Estimates of survival from the sighting of marked animals. Biometrika, 51(3-4):429–438, 12 1964.

[10] A. W. F. Edwards. Likelihood. Cambridge University Press, 1972.

[11] M. G. Efford and M. R. Schofield. A spatial open-population capture-recapture model. Biometrics, 76(2):392–402, 2020.

[12] A. et al. A tale of three cdns: An active measurement study of hulu and its cdns. In 2012 Proceedings IEEE INFOCOM Workshops, pages 7–12, 2012.

[13] D. et al. Internet scale user-generated live video streaming: The twitch case. In Proceedings of PAM, pages 60–71, New York, NY, 02 2017. Springer.

[14] W. et al. Anatomy of a personalized livestreaming system. In Proceedings of the 2016 Internet Measurement Conference, IMC '16, page 485–498, New York, NY, USA, 2016. Association for Computing Machinery.

[15] C. (evoli) Conners and R. S. Breslau. Wall street journal chart lists twitch.tv fourth in u.s. peak traffic, Feb 2014.

[16] D. Giordano, S. Traverso, L. Grimaudo, M. Mellia, E. Baralis, A. Tongaonkar, and

doi:10.6342/NTU202201298

S. Saha. Youlighter: A cognitive approach to unveil youtube cdn and changes. IEEE Transactions on Cognitive Communications and Networking, 1(2):161–174, 2015.

[17] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1):100–108, 1979.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9:1735–80, 12 1997.

[19] C. Hsu. Data analysis for cdn server population estimation based on crm model with heterogeneity, 2022.

[20] Y. Jin, S. Renganathan, G. Ananthanarayanan, J. Jiang, V. N. Padmanabhan, M. Schroder, M. Calder, and A. Krishnamurthy. Zooming in on wide-area latencies to a global cloud provider. In Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM '19, page 104 – 116, New York, NY, USA, 2019. Association for Computing Machinery.

[21] G. M. Jolly. Explicit estimates from capture-recapture data with both death and immigration-stochastic model. Biometrika, 52(1/2):225–247, 1965.

[22] C. Krebs. Ecological Methodology, 3rd ed. 2014. https://www.zoology.ubc.ca/~krebs/books.html.

[23] J. Laake, D. Johnson, and P. Conn. marked: An r package for maximum likelihood and markov chain monte carlo analysis of capture-recapture data. Methods in Ecology and Evolution, 4:885–890, 09 2013.

doi:10.6342/NTU202201298

[24] F. Lincoln. Calculating Waterfowl Abundance on the Basis of Banding Returns. Circular (United States. Department of Agriculture). U.S. Department of Agriculture, 1930.

[25] A. Mickunas. How does league’ s worlds viewership compare to the super bowl?, Feb 2018.

[26] A. M. Mood, F. A. Graybill, and D. C. Boes. Introduction to the theory of statistics. McGraw-Hill New York, 3rd ed. edition, 1974.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[28] C. G. J. PETERSEN. The yearly immigration of young plaice into the limfjord from the german sea, ect. Report of the Danish Biological Station for 1895, 6:1–48, 1896.

[29] S. Pledger, K. H. Pollock, and J. L. Norris. Open capture-recapture models with heterogeneity: I. cormack-jolly-seber model. Biometrics, 59(4):786–794, 2003.

[30] S. Pledger, K. H. Pollock, and J. L. Norris. Open capture−recapture models with heterogeneity: Ii. jolly−seber model. Biometrics, 66(3):883–890, 2010.

[31] K. H. POLLOCK. A K-sample tag-recapture model allowing for unequal survival and catchability. Biometrika, 62(3):577–583, 12 1975.

[32] B. L. S. Prakasa Rao. Maximum likelihood estimation for markov processes. Annals of the Institute of Statistical Mathematics, 24(1):333–345, Dec 1972.

doi:10.6342/NTU202201298

[33] L. Quan, J. Heidemann, and Y. Pradkin. Trinocular: Understanding internet reliability through adaptive probing. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM '13, page 255–266, New York, NY, USA, 2013. Association for Computing Machinery.

[34] D. S. R. R. M. Cormack, G. P. Patil. Sampling Biological Populations. Fairland, Maryland: International Cooperative Publishing House., 1979.

[35] C. J. Schwarz and A. N. Arnason. A general methodology for the analysis of capture-recapture experiments in open populations. Biometrics, 52(3):860–873, 1996.

[36] G. A. F. Seber. A note on the multiple-recapture census. Biometrika, 52(1-2):249–260, 06 1965.

[37] TwitchTracker. Twitch statistics and charts, Aug 2021.

[38] C. Wang. Discovering twitch's video delivery infrastructure utilizing cloud services and vpns, 2021.

[39] W.-S. Wung, G.-T. Ting, R.-T. Hsu, C. Hsu, Y.-C. Tsai, C. Wang, Y.-T. Liu, H. Chen, and P. Huang. Twitch' s cdn as an open population ecosystem. In Asian Internet Engineering Conference, AINTEC '21, page 56 – 63, New York, NY, USA, 2021. Association for Computing Machinery.

doi:10.6342/NTU202201298