2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)

The Cask Effect of Multi-source Content Delivery: Measurement and Mitigation

Xi Chen[1,2], Minghao Zhao[1], Xinlei Yang[1], Zhenhua Li[1,2*], Yao Liu[3], Zhenyu Li[4], Yunhao Liu[1,5]
[1]*School of Software, BNRCIST, and KLISS MoE, Tsinghua University, China*
[2]*Tencent Co. Ltd, China*
[3]*Department of Computer Science, Binghamton University, State University of New York, USA*
[4]*Institute of Computing Technology, Chinese Academy of Sciences (ICT-CAS), China*
[5]*Department of Computer Science and Engineering, Michigan State University, USA*
chenxi198909@gmail.com, {*mh-zhao17, yang-xl15*}*@mails.tsinghua.edu.cn*
{*lizhenhua1983, yunhao*}*@tsinghua.edu.cn, yaoliu@binghamton.edu, zyli@ict.ac.cn*

***Abstract*—With the explosive growth of Internet traffic, multi-source content delivery has been introduced for improving the performance and quality-of-experience (QoE) of Internet services. Upgrading from single-source content delivery to multi-source content delivery, however, may not always lead to a better performance. Instead, a decline in terms of delivery speed often occurs. By conducting a comprehensive study, we show that the underlying reason of this counter-intuitive phenomenon is actually due to the *cask effect* of data sources at both macro and micro level. Specifically, at the macro level, data sources with different types are highly heterogeneous in terms of delivery performance, which means data sources with certain types are particularly easy to become the "short boards". At the micro level, for the data sources chosen by a client, the high diversity of participation time (DPT) of the sources could impair the acceleration effect. Motivated by the above findings, we design MDR (Multi-source Delivery Redirector), a middleware that contains two optimizations to improve the acceleration effect. One is the feature-greedy selection algorithm which can avoid selecting data sources with inferior types, and the other is the DPT-driven shuffle strategy which can avoid using unstable data sources. Simulation-based experiments show that the MDR outperforms existing approaches in terms of overall downloading performance.**

***Keywords*-content delivery; multi-armed bandit; acceleration effect; measurement; Content Distribution Network (CDN).**

I. Introduction

Recent years have witnessed an ever-growing trend in the Internet traffic. It is predicted that the global IP traffic will reach up to 3.3 ZB per year by 2021. Among this traffic, video delivery takes up the largest portion (83%, raising from 72% in 2016) [1]. Unfortunately, the current Internet infrastructure is in dire straits to keep pace with the continuous increasing user amount and data transfer volume. Consequently, the delivering quality of experience (QoE), in terms of both download and online services, is still far from satisfactory [2]–[4].

Many observations show that there is a substantial room for improving QoE by upgrading system architectures and introducing prediction mechanisms [5], [6]. Prior systems have undergone several generations of enabling technologies, including traditional client-server (C/S) models, content delivery networks (CDNs), peer-to-peer (P2P) networks and cloud-based techniques [7]–[11]. Nowadays, most industrial content delivery systems such as Spotify [12], Thunder [13] and Speedbit [14], employ a *downloading acceleration function* which is mainly based on *multi-source content delivery* to improve their delivery [15]. Such a function enables the client to simultaneously fetch different parts of the requested content from multiple data sources using various content delivery techniques and protocols. Previous studies [16], [17] have shown that the function often possesses high efficiency when applied to the Internet video streaming service.

Unfortunately, those prior efforts fall short of providing stable and high QoE along with the growth of internet traffic, since the interactions between a client and the multiple data sources become far more complex today than before. We conduct an in-depth measurement study on a large dataset from M-Downloader, a key component embedded in Tencent products (*e.g.,* Tencent Video, QQ-Browser, and Xuanfeng download manager [18]) and evaluate its core function, *i.e.,* accelerating content delivery tasks with multiple data sources. Disappointingly, we find that the function oftentimes (with a probability of 23%) fails to meet the goal of improving delivery speed.

To understand the root cause of the failure, we investigate the detailed process of multi-source content delivery as well as comprehensively measure and analyze 1.36 billion content delivery tasks for 58 million users (including both PC and mobile clients) in M-Downloader. We figure out that the root cause of acceleration failure is the so-called *cask effect of data sources* at both macro and micro levels. Specifically, at the macro level, data sources with different types are highly heterogeneous in terms of download performance, and thus data sources with certain types are particularly easy to become the "short boards" in multi-source content delivery. At the micro level, when data sources are fixed during a period of time, the key factor that impair the acceleration is mainly the high *diversity of participation time* (referred to as *DPT*, formally defined in Equation (1)) of data sources. In

* Corresponding author

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00033

other words, *the download performance mainly relies on a small subset of data sources that participate in the download process for a relatively long period, while being distracted by other short-period data sources.*

Based on the above insights, we conclude that a major problem in the multi-source content delivery is how to select and shuffle data sources to reduce the cask effect during the delivery. Thus, we design the MDR (Multi-source Delivery Redirector), a middleware to solve this problem. MDR is implemented at the client side and contains two modules, *i.e., feature-greedy selection* and *DPT-driven shuffle*.

Feature-greedy selection. The feature-greedy selection module is designed to avoid selecting data sources with inferior types. Once the client receives the data sources allocated by the back-end cloud, MDR activates the feature-greedy selection module to select a certain number of data sources. In this module, we formalize a *contextual multi-source selection problem* and propose the feature-greedy selection algorithm. This algorithm is devised based on the classical ε-greedy algorithm in the multi-armed bandit problem.

In the classical multi-armed bandit problem, a player at a row of slot machines (*i.e.,* arms) decides which machines to play, so as to get maximum overall rewards [19]. Intuitively, each data source can be viewed as an arm individually and the acceleration rate can be used as the reward. However, one arm can be pulled *multiple times* in multi-armed bandit problem, whereas in data-source selection, each data source is only required *once* to establish the network connection. In addition, the instability (*i.e.,* fluctuation in downloading performance) of individual data sources decreases the effect of data-source selection. Thus, instead of using individual data sources directly, we propose a *feature-greedy* selection algorithm to simplify the solution of contextual multi-source selection problem.

According to our findings, the type of data source is an appropriate feature in the *feature-greedy* selection algorithm. Specifically, this algorithm is iteratively executed in multiple rounds. In each round, it either randomly selects a data source with probability ε (referred to as *exploration*), or combining with contextual information, chooses a data source with the highest acceleration reward (*i.e.,* the growth rate of download speed) (referred to as *exploitation*). Then, the acceleration reward of each feature is updated in each round. Through such exploration-exploitation, it is possible to achieve an approximately optimal accelerated reward.

DPT-driven shuffle. After connecting with the selected data sources, the DGT-driven shuffle module monitors the status of each connected data source. Periodically, it discards the data sources which are broken or whose diversity of participation time exceeds a certain threshold, and then activates the feature-greedy selection module to reselect data sources. By doing this, the DGT-driven shuffle module tries to avoid using the micro-level "short boards" (*i.e.,* the inactive data sources).

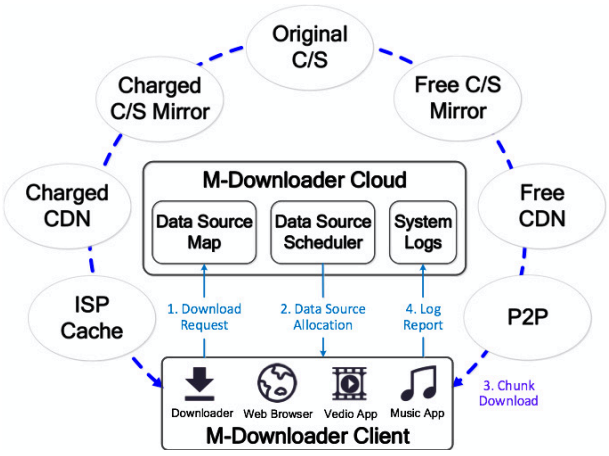

Figure 1: Architecture of the M-Downloader system.

Performance of MDR. We evaluate the performance of MDR through simulation experiments. Our simulations with large-scale traces show that MDR exhibits faster convergence than the existing random selection algorithm (*i.e.,* the core algorithm used in M-Downloader), and its average growth rate of download speed is at least 40% higher.

In summary, this paper makes three key contributions:

- A large-scale analysis of multi-source content delivery which highlights a serious anomaly in upgrading the initial single-source content delivery to multi-source, and draws data-driven insights that form the basis for our design (§III-B).
- The MDR framework for improving the multi-source content delivery performance via the online data sources selection module and DGT-driven shuffle (§IV).
- The simulation experiment that shows the advantages of feature-greedy selection algorithm compared with other related algorithms and demonstrates the effectiveness of MDR in performance improvement (§V).

II. System Overview

As demonstrated in Figure 1, the M-Downloader system is composed of two parts: the front-end client and the back-end cloud. The client is embedded in a number of popular Internet applications, such as dedicated downloader, web browser, and video streaming applications. The cloud is mainly made up of three components: the Data Source Map, the Data Source Scheduler, and the System Logs. Below we describe the working principle of the system by following the message and data flows of a typical download task.

When a user wants to acquire a file from a data source (say s_0), she first issues a *download request* to the back-end cloud through the front-end client. The download request usually

contains the URL of the original data source (*i.e.,* s_0), the user's application type, the user-configured maximum number of data sources to be used (denoted as u and set as 35 by default), and so forth. Meanwhile, the client starts to download the requested file content from s_0 to maximize its throughput, *i.e.,* begins the phase of *single-source downloading*.

On receiving the download request, the cloud first maps s_0 onto all the other data sources (say s_1, s_2, $\cdots$, s_n) that provide the same file content. This is achieved by utilizing the Data Source Map which constantly discovers and updates available data sources across the Internet. Afterwards, the Data Source Scheduler randomly picks several (say m) data sources from the n data sources (*i.e.,* s_1, s_2, $\cdots$, s_n), and then sends them to the client. Usually, m is larger than u while smaller than n (*i.e.,* $u < m < n$) as long as n is large enough. On the contrary, if $u \geq n$, all the other data sources are sent to the client (*i.e.,* $m = n$). Exceptionally, if $m = 0$, which indicates that no extra data sources are available for downloading this file, the cloud will notify the client of the impossibility of multi-source content delivery.

After getting the m data sources ($m > 0$), the client also randomly picks a few (say c) data sources to set up TCP/UDP connections with. Once a TCP/UDP connection is successfully established, the client starts downloading a chunk of the wanted file from the corresponding data source. Accordingly, the mode for downloading the file is upgraded from single-source content delivery to multi-source content delivery. Certainly, different chunks are downloaded from different data sources in parallel, so that the download of the wanted file is accelerated. Every now and then, the client ranks the connected data sources in terms of download speed, and replaces those worst data sources with new ones.

Finally, when the download task is successful, timed out, or abandoned by the user, the client sends a *log report* to the cloud which records detailed information of the data sources used during the download. By aggregating such information in the System Logs, the cloud (more specifically, the Data Source Map) is able to identify and discard those unavailable and low-quality data sources.

For each download request, M-Downloader can use up to seven types of data sources for multi-source content delivery. These types of data sources are original C/S data sources, free C/S mirrors, charged C/S mirrors, charged CDN data sources, free CDN data sources, ISP caches, P2P data sources. Normally, the CDNs are charged infrastructures that facilitate content delivery through strategically deploying edge servers at multiple locations. In this situation, the free CDNs refer to those CDNs deployed by Tencent, and they are free to M-downloader. The ISP caches, deployed by Internet Service Providers (ISPs), are also free to M-downloader users. Overall, these seven types of data sources cover almost all popular content delivery techniques and protocols at present.

III. Dataset and Performance Analysis

To understand the performance characteristics of multi-source content delivery, we study a large-scale dataset collected from the M-Downloader system and evaluate the performance of M-Downloader via comprehensive real-world measurements.

A. Dataset

The dataset contains complete running logs of the system during a whole week (July 13–19, 2016), involving 1,364,122,406 download tasks, 57,538,801 users and 9,827,109 unique files. Among these download tasks, the majority (59%) utilized multiple (≥ 2) data sources, and the remainder (41%) only used the original (single) data source. The number and percentage of each type of data sources used during the week is shown in Table I.

For each download task, the client sends one log report to the cloud. As illustrated in Table II, a multi-source content delivery log includes basic information about the download task, as well as information of all the data sources used during the downloading. Specifically, the *report time* is in the UNIX-timestamp format. Then, each user has a unique *user ID* which is the MD-5 hash code of the user account, and each file has a unique identifier (*file hash*) which is the MD-5 hash code of the file content. Moreover, *download time* represents the total time consumed by a download task. As for the *download result*, 0, 1, and 2 indicates a download success, timeout, and cancellation, respectively.

A *data source* is represented by an HTTP, FTP, BitTorrent, eMule, or Magnetic URL, from which we can easily recognize the type of the data source. Also, the *IP address* of the data source is recorded. Afterwards, the *acquired size* is the size of data chunks got from the data source, while the *network traffic* indicates the total traffic bytes exchanged with the data source. As the system will periodically abort data sources with inferior download speed or broken connection), the data sources have verified *participation time*. It is measured as the duration, starting from the time that the data source is connected successfully, to the time that this data source is abandoned or the download task is finished. Dividing the acquired size by this time, we get the average download speed from a data source.

B. Performance Analysis

We use the dataset mentioned before to analyze the performance of multi-source content delivery, and also evaluate the seven types of data sources introduced in §II.

Observation 1: 23% of the downloads have worse performance after being upgraded to multi-source content delivery.

Our collected dataset shows that in a whole week, 0.57 billion downloads are using a single data source with an

Table I: Number and percentage of data sources used in one week's file downloads.

Data Source	Original C/S	Free C/S mirror	Charged C/S mirror	Charged CDN	Free CDN	ISP Cache	P2P
# of Data Source	45078026	215209375	15042805	1761653	710193	2357478	37580148
% of Data Source	14.2%	67.7%	4.7%	0.6%	0.2%	0.8%	11.8%

Table II: Detailed information in a log report corresponding to a typical multi-source downloading task.

Report Time	**User ID**	**File Hash**	**File Size (KB)**	**Download Time (s)**	**Download Result**	···
1436719727	3E2···A5	EC7···2A	160000	536	0 (Success)	···
Performance Records of All Data Sources Used						
Data Source	**IP Address**	**Acquired Size (KB)**	**Network Traffic (KB)**	**Participation Time (s)**	···	
http://dl.abc.com/file.rm	116.16.17.214	640	712	10	···	
magnet:?xt=urn:btih:ffd···	183.15.13.224	10980	14765	85	···	
·····						

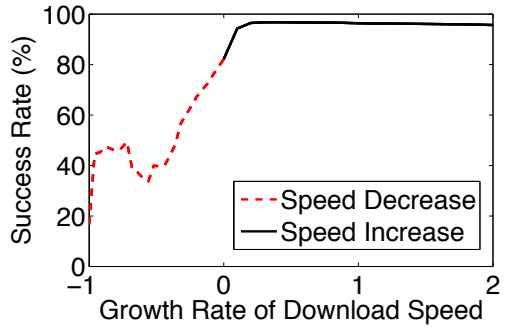

Figure 2: The success rate for different speed growth rate.

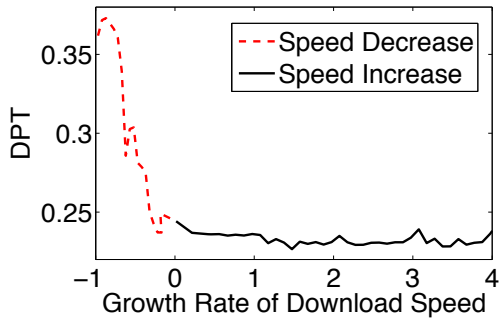

Figure 3: DPT *vs.* growth rate of download speed.

average speed of 237 KBps and a success rate of 96.7%, while 0.8 billion are using multiple (2.94 in average) data sources with an average speed of 728 KBps and a success rate of 98.4%. This general statistic comparison seems to present that multi-source content delivery definitely outperforms single-source content delivery. Nevertheless, detailed examination on the effect of upgrading (from original single-source content delivery to multi-source content delivery) reveals unexpected performance degradations.

The performance degradations first appear in the download speed. We use the *growth rate* of download speed to represent the effect of multiple sources in accelerating the downloading performance. It is defined as $\frac{S_m - S_s}{S_s}$, in which S_s and S_m denote the average speed of downloading a file in single-source and multi-source phase respectively. Once this value is negative, it means that the download speed decreases when the original single-source content delivery is upgraded to multi-source content delivery. Surprisingly, after the upgrade, the proportion of the tasks with speed decrease is 23%. Additionally, approximately 37% of downloads are trivially accelerated by almost 0 KBps.

Although multi-source content delivery sometimes generates a worse download speed, users would still believe that using multiple data sources can at least enhance the download success rate. Unfortunately, the performance degradation also appears in the download success rate. As indicated in Figure 2, there is an obviously positive correlation between the download success rate and the acceleration effect. Specifically, when a download is slightly accelerated, its success rate would exceed 80%; when a download is considerably accelerated, its success rate would be as high as 94%. On the contrary, when a download is decelerated, its success rate can hardly reach 80% (shown as the red dashed curve in Figure 2), sometimes even falling below 50%.

Observation 2: The diversity of participation time of all data sources used in one download task remarkably affects the acceleration performance.

The above several paragraphs reveal a counter-intuitive phenomenon in our study, *i.e.,* multi-source content delivery is sometimes worse than the original single-source content delivery in terms of both download speed and success rate. Seeking for a reasonable explanation to this phenomenon, we examine the relationships between the acceleration effect and some metrics. Actually, we find the Diversity of Participation Time (DPT) of data sources largely affects the overall download performance.

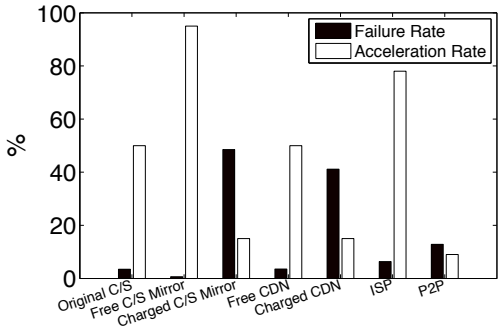

Figure 4: Accelerate rate and failure rate of different types.

The DPT is defined to measure heterogeneities in terms of contributions among the data sources used. It is calculated as the standard deviation divided by the range, *i.e.,*

$$DPT = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(T_i - \overline{T})^2}}{T_{max} - T_{min}}, \quad (1)$$

where T_i denotes the participation time of the i-th data source, $\overline{T}$ denotes the average participation time of the data sources used, and T_{max} and T_{min} denote the the maximum and minimum participation time of these sources. As described in §II, when multiple data sources are used during a download, all these data sources do not upload data all the time. Instead, each data source uploads data for a specific participation time.

Figure 3 quantifies the obviously negative correlation between the diversity of participation time and the growth rate of download speed. Specifically, when the diversity of participation time is small (< 0.2), a download task can usually benefit from using multiple data sources. In fact, this means that all data sources used are making similar contributions. On the other hand, once the diversity of participation time is large (> 0.25), the user can hardly benefit from multi-source downloading. Essentially, the download process mainly relies on a small subset of data sources that participate for a long period, while being distracted by other short-period data sources. Thus, to achieve effective acceleration by multi-source content delivery, the data sources need to be carefully probed and selected. Specifically, once multi-source content delivery becomes slower than single-source content delivery and the diversity of participation time is larger than the threshold (0.25), multi-source content delivery should be degraded.

Observation 3: There is a significant diversity among data sources with different types, which reflects the differences of acceleration reward and download result.

Table III: IG of Attributes in Download and Acceleratioin.

	File Size	Type of Data Source	Original Speed
Download	0.68	0.83	0.61
Acceleration	0.35	0.60	0.33

The DPT of data sources heavily affects the growth rate of download speed, thus it can be used to adaptively adjust the accelerating strategy in real-time. However, the DPT can be calculated only after the download task switches to multi-source downloading mode. Accordingly, we hope to find some inherent attributes to versatilely guide the acceleration.

In order to quantitatively evaluate the effects of download (in terms of success rate) and acceleration (*i.e.,* increase in download speed) of different inherent attributes, we use information gain to measure the importance of each attribute. Information gain (IG) is defined as $IG(X) = H(C) - H(C|X)$, where $H(C)$ and $H(C|X)$ are the entropy of C and the conditional entropy of C. It is normally used to quantify the usability and effectiveness of a feature for prediction. Here, the selected attributes include file size, the type of data source and the original download speed.

Table III displays the result of information gain of each attributes. Specifically, the type of data source is the most important indicators for both download and acceleration, while file size and original download speed are not the essential determinants. Furthermore, we compare seven types of data source on the download failure rate and the growth rate of acceleration (*i.e.,* acceleration rate). Based on the comparison result shown in Figure 4, we summarize that among seven types of data source, free C/S mirror has the best delivery quality, followed by ISP, free CDN and original C/S, with charged CDN, charged C/S mirror and P2P worst, considering both download failure rate and acceleration rate. This indicates that the *type of data source* is a fundamental attribute that influences the overall acceleration effect, and among these types, free C/S mirror and ISP should gain priority in the data source selection.

Key observations: In summary, our analysis of acceleration effect suggests that:

- Multi-source content delivery may not always result in downloading QoE (in terms of download speed and success rate) improvement. Sometimes it may deteriorate the download performance.
- Excessive or inappropriate use of data sources may hurt the download performance and DPT is a core factor influences the acceleration effect.
- Data sources with different types are highly heterogeneous in terms of delivery performance, which results in different average growth rates of download speed, success rates of acceleration and download.

IV. MDR Overview and Design

Driven by the above findings, we propose the Multi-source Delivery Redirector (MDR), a middleware that helps the clients to improve the overall downloading performance. In this section, we first present an overview of MDR, including its workflow, usage scenarios and components, and then describe the feature-greedy selection algorithm and DPT-driven shuffle strategy used in it.

A. Overview

Figure 5 shows the basic framework of MDR. After receiving the allocated data sources from the server, the client picks some data sources to set up TCP/UDP connections with. Instead of picking data sources randomly, MDR utilizes feature-greedy selection algorithm to reasonably select data sources with high quality. Meanwhile, the DPT-driven shuffle component monitors the connected data sources and then throws out inferior data sources. Once some connected data sources are abandoned, the feature-greedy selection algorithm is invoked again and selects some new data sources to built TCP/UDP connections with.

As we can see from the Figure 5, MDR is designed to plug in the M-Downloader client. Besides, MDR can also be regarded as a plug-in component integrated into many content delivery tools, such as online video platforms (*e.g.,* iQIYI, youtube, Tencent Video), music app (*e.g.,* QQ Music), and other file downloader (*e.g.,* Thunder). In order to improve transfer performance, these tools usually utilize a function to accelerate the content delivery by connecting a variety of data sources, For example, Tencent Video utilizes P2P and CDN data sources simultaneously to improve the fluency of media display. This function is very similar to the acceleration principle used in the M-Downloader client. Thus, it is feasible to integrate MDR into general content delivery tools with acceleration capability.

In conclusion, the core objective of MDR is to improve the performance of acceleration function adopted in content delivery tools. The solutions are mainly built on the data-driven observations in §III-B. First, based on **Observation 3** in §III-B, MDR leverages the features of data sources to predict their acceleration effect and then selects the data sources with the best acceleration effect. Second, to correct false selection and prevent connection failure, MDR evaluates the combination effects of these selected data sources and throws out the data sources of low quality based on **Observation 2** in §III-B.

Given the basic overview, there are two practical questions:

1) *How to select data sources reasonably?*
2) *How to update the set of connected data sources?*

Next, we address these questions.

B. Online Data Source Selection

Problem Formalization. In data source selection problem, the client receives a number of data sources (denoted as

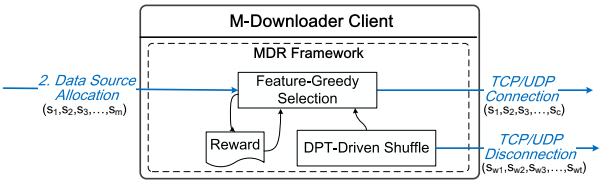

Figure 5: Architecture of the M-Downloader system.

$s_1, s_2, \ldots, s_m$) allocated by the server, and needs to pick c data sources to establish TCP/UDP connections (normally $c < n$). Before that, the client doesn't know the connecting quality of each data source, but knows some features ($\mathcal{F}$) of data source, *e.g.,* the type, IP address, previous quality and *etc.*The data source selection algorithm must find c data sources to optimize the acceleration effect. Here, we use the growth rate of download speed to quantify this effect. Thus, the quantitative value is also called acceleration rate (AR).

This problem can be naturally modeled as a *multi-armed bandit problem* with context information in *reinforcement learning*. The classical multi-armed bandit problem is a contextual-free bandit problem in which a player at a row of slot machines (*i.e.,* arms) has to decide which machines to play. When played, each machine provides a random reward from a probability distribution specific to that machine. The objective of the player is to maximize the sum of rewards earned through a sequence of lever pulls [19]. This poses an exploration-exploitation trade-off: the player simultaneously attempts to acquire new knowledge (called "exploration") and optimize his or her decisions based on existing knowledge (called "exploitation").

In our formalization, we may view data sources allocated by server as arms and define the acceleration rate as reward. Based on the previous knowledge, we may select a data source with the highest reward to build a TCP/UDP connection. However, by contrasting our problem and multi-armed bandit problem, we find that one arm could be pulled more than one time during the play, while one data source only need to be connected once at most in one download task. Besides, the server may allocate dozens of data sources to the client for one download request. Therefore, the exploration for each data source in one client may not make much sense and even be time-consuming.

In order to deal with the aforementioned dilemma, we propose to simplify the online data source selection problem to the feature-based online data source selection problem. Here, the feature mainly denotes the seven types of data sources. This simplification is inspired by **Observation 3** in §III-B, *i.e.,* data sources with different types are highly heterogeneous. In this way, the exploration-exploitation is mostly depended on the type of data sources. Following the previous works [20], [21], we formally define the feature-based data source selection problem as follows.

Definition 1: (Feature-Based Online Data Source

Selection Problem) *In a feature based online data source selection problem, there is a distribution P over* $(\mathcal{F}, s_1, \ldots, s_m)$*, where* $\mathcal{F}$ *is the set of features,* $a \in (1, \ldots, m)$ *is one of the m data sources to be connected, and* $AR_a \in [-1, 1]$ *is the reward for data source a. The problem is a repeated game: on each round t, a data source* a *is chosen by the client based on its feature* $\mathcal{F}$*, and its reward* $AR_{a_t,t}$ *is revealed.*

In the process of selection, we define the total T-round reward as $\sum_{t=1}^{T} AR_{a_t,t}$. Similarly, the optimal expected T-trial reward is defined as $E(\sum_{t=1}^{T} AR_{a_t^*,t})$ where a_t^* is the data source with maximum expected reward at the round t. For multi-armed bandit model, the optimal arm-selection strategy should minimize the *regret*. Similarly, the selection algorithm should employ a selection strategy which minimizes the *regret*. Here, the T-round *regret* $R(T)$ is defined formally by

$$R(T) = E(\sum_{t=1}^{T} AR_{a_t^*,t}) - E(\sum_{t=1}^{T} AR_{a_t,t}) \qquad (2)$$

Alternatively, our goal is to design an algorithm such that the $E(\sum_{t=1}^{T} AR_{a_t,t})$ is maximized.

Algorithm Design. - uring each round, the standard form of multi-armed bandit problem does not consider any additional information besides the observed reward of previous selected arms. Whereas our formalization of feature-based online data source selection problem is inspired by contextual multi-armed bandit model [20], which includes external information to direct the selection strategy. Therefore, we refer to the solution of contextual multi-armed bandit problem, and propose a *feature-greedy selection algorithm* (**Algorithm 1**) to solve the online data source selection problem in client.

This algorithm is based on the simplest and the most widely-used algorithm (*i.e.,* ε-greedy algorithm) in multi-armed bandit problem [20], [22], and is improved by taking advantage of the observation in §III-B. Formally, the feature-greedy selection algorithm earns the rewards from discrete rounds $t = 1, 2, \cdots, n$. On the whole, the improvements in our algorithm include two aspects. On one hand, Algorithm 1 integrates ε-decreasing strategy to improve the performance. In ε-greedy algorithm, ε is usually a constant value to balance the exploration and exploitation. However, at first the client doesn't know the performance of data sources with different features. Thus it needs explore frequently to learn this knowledge, so the value of ε should be set large enough. Once obtaining this knowledge, the client expects to use the data sources with best performance so as to maximize the benefit. At this time, a smaller ε is needed. Note that, the experimental result in section V-B gives the optimum value's choice of ε_0, *i.e.,* the decreasing limit of ε is 0.05. On the other hand, we choose the type of data source as the feature $\mathcal{F}$ according to the actual measurement result, since the different types of data sources can result in different performances and rewards. Moreover, one thing to be aware of is that the feature selection is scalable in this algorithm. That is to say, if we discover that some other features (*e.g.,* the server's geographic location and history performance) have significant impacts on the performance of content delivery, these features can also be adopted into our algorithm to improve the effectiveness.

Algorithm 1 Feature-Greedy Selection in M- R.

Input:
The number of rounds T, a set of data sources with features $< F, S >= \{(f_{s_1}, s_1), (f_{s_2}, s_2), \ldots, (f_{s_m}, s_m)\}$, a limit probability of exploration ε_0, and the exploration samples W;

Output:
renewed exploration samples W;

for $t = 1, 2, \ldots, T$ **do**
  $\varepsilon = \frac{1}{\sqrt{|W|}}$;
  if $\varepsilon < \varepsilon_0$ **then**
    $\varepsilon = \varepsilon_0$;
  end if
  if $rand() < \varepsilon$ **then**
    //*do one-step exploration*//
    $a \Leftarrow$ a data source randomly selected from S;
  else
    //*do one-step exploitation*//
    $a \Leftarrow$ a data source with the feature $\arg\max AR_{f_i}$;
  end if
  Observe a real-valued reward $AR_{a_t,t}$;
  $W = W \bigcup \{(f_t, a_t, AR_{a_t,t})\}$;
  Update reward $AR_{f_t} = \frac{\sum_{i=1}^{|W|} AR_i I(f_i = f_t)}{count(I(f_i = f_t))}$;
end for
return W;

C. DPT-Driven Shuffle

In current content delivery system, the shuffle of data sources mainly depends on a relatively empirical rule, *i.e.,* if the connected data sources transmit slowly or are disconnected, they will be replaced with new ones directly. This straightforward design only considers the transfer speed of data sources. However, as more than one data source undertakes the acceleration task together in multi-source content delivery system, the acceleration effect is more complex and thus it needs more systematic measurements to conduct the shuffle of data sources.

In §III-B, we have explained the reason of worse performance after accelerating. The measurement result reveals that the diversity of participation time of data sources (- PT) is one of the crucial factors in assessing the effect of acceleration. More specifically, the multi-source content

delivery mainly relies on a small subset of data sources that participate for a long period steadily and continuously, while being distracted by other short-period data sources. Motivated by this observation, we propose DPT-Driven shuffle strategy to improve the original design.

Our shuffle strategy is launched after data sources are connected successfully. It records the delivery performance of each data source at regular intervals, including the result of connection, the participant time and the average speed. The shuffle rules include two: 1. Once broken links are discovered, these disconnected data sources should be abandoned; 2. Once the total speed of multi-source content delivery becomes slower than that of single-source content delivery, and DPT is larger than threshold 0.25, the data sources with smallest participant time should be thrown away. This step should be executed cyclically until DPT is smaller than 0.25. The feature-greedy selection algorithm will be performed to select other available data sources if either of the rules is satisfied. Once multiple data sources are used simultaneously to download one file, this step should be executed cyclically until DPT is smaller than 0.25. The feature-greedy selection algorithm will be performed to select other available data sources if one or more data sources are abandoned after executing the shuffle rules.

V. EVA0UATIO&

To evaluate the general effectiveness of our optimizations, we conduct comprehensive experiments with the MDR. In this section, we first present the details of the evaluation methodologies and then demonstrate the evaluation result of MDR with large-scale simulations.

A. Evaluation Setup

Evaluation Framework: We evaluate the performance of our system via simulations with a large data set (detailed in §V-B). Specifically, in the simulation framework, we simulate the process of data source selection and replacement. The dataset is from M-Downloader logs. The purposes of simulation are to determine the optimal coefficient of feature-greedy algorithm and compare the performance of different multi-armed bandit algorithms. After achieving this, we get our feature-greedy algorithm with best-suited parameters. We finally compare the performance of our feature-greedy algorithm with other methods.

Algorithms: We compare the following algorithms with feature-greedy algorithm in simulations to evaluate their relative performance. All of these algorithms are proposed to solve the multi-armed bandit problem so as to be useful for the data source selection problem.

1) ε-greedy: This algorithm selects the data source of highest expected AR with $1-\varepsilon$ proportion and otherwise selects a random arm with ε proportion [22]. ε is usually a constant value set by the user to balance the exploration and exploitation.
2) Softmax: This algorithm chooses a data source according to a Gibbs distribution. The probability of being chosen is proportional to its previous AR, *i.e.,* $p_k = e^{\hat{u}_k/\tau} / \sum_{i=1}^{n} e^{\hat{u}_k/\tau}$, where $\hat{u}_i$ is the estimated mean of AR brought by the data source i and $\tau \in \mathbb{R}^+$is a parameter called the temperature [23]. τ is left to the user. A higher value of τ means more exploration of data sources.
3) UCB2: In each round, data source i is selected to maximize $\overline{x}_i + \sqrt{\frac{(1+\alpha)(ln(en/(1+\alpha)^{r_i}))}{2(1+\alpha)^{r_i}}}$ and then performed exactly $\lceil (1+\alpha)^{r_i+1} - (1+\alpha)^{r_i} \rceil$ times before ending the round and selecting a new data source [22]. The parameter α controls the intensity of the confidence bound.

B. System Performance

First, we compare the performance of feature-greedy selection algorithm with that of three multi-armed bandit algorithms (*i.e.,* ε-greedy, softmax and UCB2) when they are adopted to solve data source selection problem. We utilize the dataset from M-Downloader to simulate the process of multi-source selection. We run each algorithm for 10000 simulated tasks and report the average ASR for 1000 rounds of each task. In each round, the algorithm should pick 5 data sources to accelerate the content delivery. For the convenience of comparison, we normalize AR of each data source into the range $[-1, 1]$ using the formula $X' = a + \frac{(X - X_{min})(b-a)}{X_{max} - X_{min}}$. Here, a and b denote the interval scope ($a = -1$ and $b = 1$), X_{min} and X_{max} denote the minimum and maximum value of array.

Figure 6 and 7 show the parameter values of each algorithm against the average AR and ASR. The ε-greedy and softmax algorithm achieve the highest AR and ASR at small values of ε and τ, while the UCB2 algorithm is the opposite. By comparison, the optimal parameter values of $\varepsilon \in [0, 0.1]$, $\tau \in [0, 0.1]$ and $r \in [0.75, 0.85]$. Accordingly, we set $\varepsilon = 0.05$, $\tau = 0.05$ and $r = 0.8$ respectively.

&ext, we evaluate the performance of feature-greedy selection algorithm by comparing it with other three algorithms. These algorithms are tested offline with data sampled from M-Downloader. We run each algorithm 1000 rounds on 10000 tasks randomly sampled from the dataset, and then report the ASR values and the average speed after accelerating. Here, we suppose each task also should use 5 data sources to accelerate. We compare the feature-greedy selection algorithm with the algorithms selected in previous experiments. The parameter value ε_0 in the feature-greedy selection algorithm is set as 0.05 according to the previous experimental results. Figure 8 and 9 display the comparison results. The ASR and average speed increase with the number of completed rounds. The feature-greedy selection algorithm performs consistently better than other algorithms

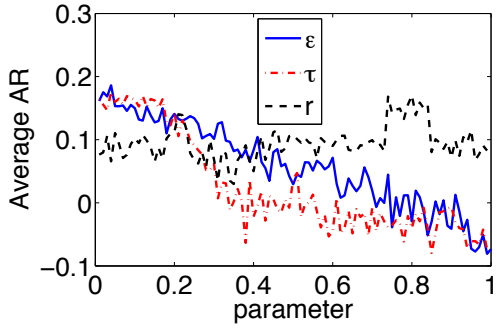

Figure 6: Average AR in different parameters.

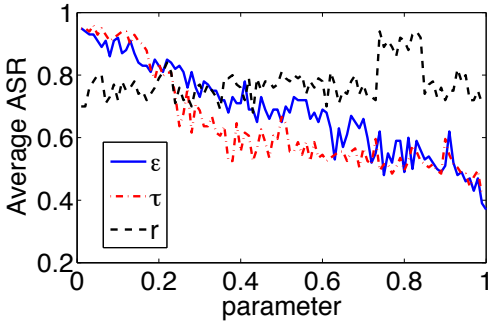

Figure 7: Average ASR in different parameters.

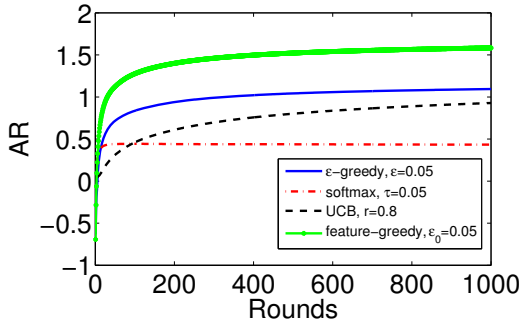

Figure 8: Comparison of AR on 1000 rounds.

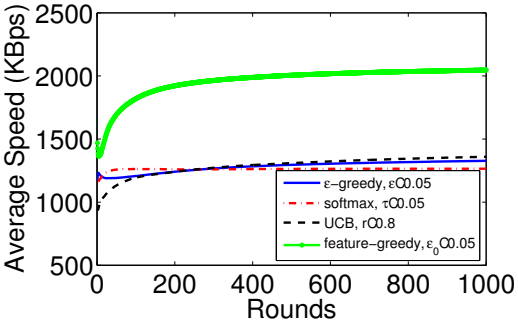

Figure 9: Comparison of average speed on 1000 rounds.

in both ASR and average speed. This highlights the fact that MDR is effective in improving the acceleration effect through learning and adjusting data source selection strategy.

VI. Related Work

System architecture in multi-source content delivery: As the state-of-the-art approach to accelerating file delivery, multi-source content delivery has attracted wide attention. Cloud-based CDN has begun to enhance their content delivery performance by purchasing resources from clouds [24]. In this way, bandwidth/storage resources pervasively existing in the Internet could be fully and collaboratively utilized. Hybrid CDN-P2P or CDN-ISP [16], [17], [25] benefits from the quality control and reliability of a CDN and the inherent scalability of a P2P or ISP system. Open-P2SP, a generalized and extended mode of P2SP, integrates various third-party servers, content and data transfer protocols across the Internet [26]. Compared with the architecture mentioned above, M-Downloader practically includes all types of data sources and thus could be used to provide more effective acceleration function and make fully comparative study.

Large-scale data analytics for system improvement: Data-driven techniques for performance improvements have been applied in many works. The design of RCA is inspired by a case study which demonstrates the existence of malicious clients to influence accounting accuracy [17]. CS2P achieves improvement on overall QoE by using a data-driven model to learn clusters of similar sessions, an initial throughout predictor and a Hidden-Markov-Model [6]. CFA achieves scalable and accurate prediction based on a global view of quality measurements [27]. Unlike prior works, we investigate the core function of multi-source content delivery, *i.e.,* acceleration, which is widely applied in many large-scale systems. Driven by comprehensively observations, we improve the existing data source selection algorithm and shuffle strategy in client to enrich QoE.

Multi-armed bandit problem and algorithms: Multi-armed bandit problem is a lighter-weight version of reinforcement learning. One of the challenges in this problem is to balance the exploration and exploitation so as to achieve highest reward from multiple choices [28]. Traditional multi-armed bandit algorithms assume each decision has a fixed distribution of rewards, but the acceleration reward of a data source also depends on features of this data source and client-side [22], [23]. In contrast, we leverage contextual multi-armed algorithms which assume the best decision depends on contextual information [20], [21]. This information is derived from the large-scale data analytics. Consequently, the overall reward gets a noticeable upgrading.

VII. Conclusion

In this paper, we conducted an in-depth measurement study on large-scale content delivery data. The measurement result reveals that multi-source content delivery cannot always improve the download performance, and it is subjected to the so-called *cask effect of data sources*. Accordingly, we develop the MDR framework to alleviate the *cask effect* and improve the acceleration effect. MDR uses the feature-greedy selection algorithm and DPT-driven shuffle strategy to pick appropriate data sources to connect with. As a general middleware framework, the MDR can be easily plugged into many content delivery tools to improve their acceleration performance.

Acknowledgements

This work is supported in part by the National Key R&D Program of China under grant 2018YFB1004700, and the National Natural Science Foundation of China (NSFC) under grants 61822205, 61432002 and 61632020.

References

[1] "Cisco Visual Networking Index: Forecast and Methodology, 2016-2021," https://www.cisco.com.

[2] S. Sundaresan, N. Feamster, R. Teixeira, and N. Magharei, "Measuring and Mitigating Web Performance Bottlenecks in Broadband Access Networks," in *Proceedings of ACM IMC*, 2013.

[3] S. Sundaresan, W. De Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè, "Broadband Internet Performance: A View From the Gateway," in *Proceedings of ACM SIGCOMM*, 2011.

[4] A. Balachandran, V. Sekar, A. Akella, and S. Seshan, "Analyzing the Potential Benefits of CDN Augmentation Strategies for Internet Video Workloads," in *Proceedings of AMC IMC*, 2013.

[5] V. K. Adhikari, Y. Guo, F. Hao *et al.*, "Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery," in *Proceedings of IEEE INFOCOM*, 2012.

[6] Y. Sun, X. Yin, J. Jiang *et al.*, "CS2P: Improving Video Bitrate Selection and Adaptation with Data-driven Throughput Prediction," in *Proceedings of ACM SIGCOMM*, 2016.

[7] Y. Huang, T. Z. Fu, D.-M. Chiu, J. Lui, and C. Huang, "Challenges, Design and Analysis of a Large-scale P2P-VoD System," in *Proceedings of ACM SIGCOMM*, 2008.

[8] C. Wu, B. Li, and S. Zhao, "On Dynamic Server Provisioning in Multichannel P2P Live Streaming," *IEEE/ACM Transactions on Networking*, Oct. 2011.

[9] Y. He and Y. Liu, "VOVO: VCR-Oriented Video-on-Demand in Large-Scale Peer-to-Peer Networks," *IEEE Transactions on Parallel and Distributed Systems*, Apr. 2009.

[10] G. Chen and Z. Li, *Peer-to-Peer Network: Structure, Application and Design*. Tsinghua University Press, 2007.

[11] C. Jiang, Y. Chen, Y. Ren, and K. R. Liu, "Maximizing Network Capacity with Optimal Source Selection: A Network Science Perspective," *IEEE Signal Processing Letters*, Jul. 2015.

[12] G. Kreitz and F. Niemela, "Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming," in *Proceedings of IEEE P2P*, 2010.

[13] "Xunlei offline downloading system." http://lixian.xunlei.com.

[14] "Speedbit - a Free Download Manager & Video Downloader," http://www.speedbit.com/.

[15] Z. Li, Y. Huang, G. Liu *et al.*, "Challenges, Designs, and Performances of Large-Scale Open-P2SP Content Distribution," *IEEE Transactions on Parallel and Distributed Systems*, Nov. 2013.

[16] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming: Experiences with LiveSky," in *Proceedings of ACM MM*, 2009.

[17] P. Aditya, M. Zhao, Y. Lin *et al.*, "Reliable Client Accounting for P2P-infrastructure Hybrids," in *Proceedings of USENIX NSDI*, 2012.

[18] "Tencent Products and Services." https://www.tencent.com/en-us/system.html.

[19] W. G. Macready and D. H. Wolpert, "Bandit Problems and the Exploration/exploitation Tradeoff," *IEEE Transactions on Evolutionary Computation*, Apr. 1998.

[20] J. Langford and T. Zhang, "The Epoch-greedy Algorithm for Multi-armed Bandits with Side Information," in *Proceedings of ICML*, 2008.

[21] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A Contextual-bandit Approach to Personalized News Article Recommendation," in *Proceedings of ACM WWW*, 2010.

[22] G. Burtini, J. Loeppky, and R. Lawrence, "A Survey of Online Experiment Design with the Stochastic Multi-Armed Bandit," *Computer Science*, 2015.

[23] J. Vermorel and M. Mohri, "Multi-armed Bandit Algorithms and Empirical Evaluation," in *Proceedings of ECML-PKDD*, 2005.

[24] Z. Li, Y. Dai, G. Chen, and Y. Liu, *Content Distribution for Mobile Internet: A Cloud-based Approach*. Springer, 2016.

[25] B. Frank, I. Poese, Y. Lin *et al.*, "Pushing CDN-ISP Collaboration to the Limit," *ACM SIGCOMM Computer Communication Review*, Jul. 2013.

[26] P. Dhungel, K. W. Ross, M. Steiner, Y. Tian, and X. Hei, "Xunlei: Peer-assisted Download Acceleration on a Massive Scale," in *Proceedings of PAM*, 2012.

[27] J. Jiang, V. Sekar, H. Milner *et al.*, "CFA: A Practical Prediction System for Video QoE Optimization," in *Proceedings of USENIX NSDI*, 2016.

[28] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," *Machine Learning*, 2005.