Towards a Fast, Scalable, and Robust End-to-End Design of Content Distribution System

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Yingying Chen

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Prof. Zhi-Li Zhang

April, 2013

Acknowledgements

I would like to thank all those who have helped me throughout my PhD life to make it an invaluable experience in my life.

First of all, I would like to thank my adviser, Prof. Zhi-Li Zhang, for providing me persistent support and encouragement, and guiding me on conducting research and achieving personal long-term career goals.

I thank my lab mates for collaborating with me on a number of exciting research projects. In particular, working with Sourabh Jain and Vijay Adhikari has greatly enriched my research experiences.

I am grateful to my talented colleagues in Microsoft, including Ratul Mahajan, Baskar Sridharan, and Nick Holt. Their valuable suggestions and feedbacks have helped me turn my research into reality.

Finally, I really appreciate Professors Abhishek Chandra, Andrew Odlyzko, Jon Weissman, and Tian He for serving as my PhD thesis (or oral) defense committee members and spending their precious time and efforts during my PhD defense.

Dedication

This thesis is dedicated to several important people in my life. I dedicate it to my parents, my advisor, my lab mates and my colleagues.

Abstract

With the rapid growth of the amount of content published and shared in the cloud, as well as the increasingly high user demands for quick access of the content over the Internet, the existing content distribution system faces a variety of challenges in bringing a fast, scalable, and robust content delivery experience to the users. Via extensive measurement studies on the existing large-scale content distribution systems, we aim to understand the limitations in the current design framework, and seek solutions towards an improved end-to-end user experience.

Our contributions are mainly three-fold. First, we study one of the largest online services provided by Yahoo!. Using the network traces collected at five major Yahoo! data centers, we make the first such effort in understanding the traffic dynamics among different back-end data centers within a content provider. Our results reveal the tiered structuring of the data centers at the back-end as used by Yahoo!. Different types of traffic including the inter-data center traffic, and the data center-to-client traffic are teased out from each other in the dataset using inference-based techniques. A deep investigation of the traffic patterns and correlations among different types of traffic has led to important insights for the distribution and replication strategies at the back-end.

Second, using two of the largest search services Bing and Google as case studies, we conduct extensive active measurement analysis in characterizing the roles of the front-end edge servers in the end-to-end latency performance of dynamic content distribution. It highlights the trade-off between the user-to-edge last mile latency and the edge-to-data center fetch time when designing the placement strategies for front-end edge servers.

Third, to complement the first two studies, one on the back-end data centers, the other on the front-end edge servers, we study how the user characteristics affect the overall performance, and how that factor affects the design decisions at the service providers. For this purpose, we collect detailed measurement data from one of the largest search service providers in US, and perform an extensive passive measurement study based on that. This study culminates with the design and deployment of an anomaly detection and diagnosis algorithm that proves to be essential in helping the content providers improve the robustness of the system.

In summary, this thesis provides an extensive end-to-end study of the existing content distribution systems, from the back-end data centers, to the front-end edge servers, and to the user-side characteristics, and how different entities interplay with each other in driving the overall user experience. Although our study mainly focuses on Yahoo! and search services, we believe its findings and methodologies have important implications on other online services as well, as they share similar content distribution framework.

Contents

List of Tables

List of Figures

Due to the massive growth in the amount of data published and shared on the Internet, content distribution systems have been widely built as a key means to access a variety of content online such as Google and Bing search, Youtube and Netflix video streaming, and Amazon shopping. However, as the user demand increases, it also poses new challenges to the existing content distribution system in terms of its latency, scalability, and robustness performance. Therefore, understanding the limitations in the existing system, and designing better strategies to enhance the performance become critical to the success and monetization ability for service providers.

The end-to-end user-perceived performance in content distribution systems hinges on a confluence of factors, from the server-side data center performance, to the Content Distribution Network (CDN) edge server performance, and their placement strategy, to the user-side network characteristics, usage patterns, and web browser (or client machine) speed. Due to the limited visibility among different entities within a content distribution system, there exist many prior works attempting to improve the user experience either on the CDN side or on the content provider side alone. Very few efforts have been found in conducting an end-to-end investigation on how each of the individual entity influence the overall user-perceived performance, and how these factors interact or counteract with each other in making design decisions to optimize the end-to-end user experience.

To overcome this limitation, we conduct an extensive study to examine the challenges in improving the end-to-end content delivery performance. Specifically, our study spans two major systems: (a) Yahoo! online services, and (b) two of the largest search services in US. Yahoo! is one of the largest content providers both within and outside US, while search has also become one of the major Web services for users to access the content of their interest. Due to the use of combined active measurements (from user's perspective) and passive measurements (from content provider's perspective), unlike other previous works, our study accounts for the impact not only from back-end servers deep in the cloud of content providers, and the CDN edge servers sitting in-between users and content providers, but also from the end users' characteristics.

By conducting studies on the two large-scale content distribution systems, we aim to seek solutions to key design decisions in building a fast, scalable, and robust content distribution system. For instance, how do content providers place and organize their

back-end data centers? What kind of traffic is carried within these data centers? How do these data centers interact with each other in providing a scalable service to the users? How can we improve the placement strategies of CDN edge servers to further enhance the user-perceived latency experience? How do the network characteristics, and user-side browser speed affect the overall latency performance? How can content providers and CDNs gear towards user-side factors, which is not directly under their control?

With these goals in mind, we will discuss three separate measurement studies we conducted on these two major content distribution systems in the following sections, understand their current practices, and limitations, and propose ways to further enhance their performance, in terms of their latency, scalability, and robustness.

0.0.1 Bibliographic Notes

Part of the content of Chapter 2 is from a conference paper, titled *A First Look at Inter-Data Center Traffic Characteristics through Yahoo! Dataset*, which appeared in the Proceedings IEEE INFOCOM, Shanghai, China, April 10-15, 2011 [1]. The active measurement study of the front-end server placement strategy in two of the largest search services is presented in a paper titled *Characterizing Roles of Front-end Servers in End-to-End Performance of Dynamic Content Distribution*, which appeared in the Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement, Berlin, Germany, Nov 2-4, 2011 [2]. This constitutes a part of Chapter 3. Chapter 4 is from a passive measurement study of the search response time we conducted on one of the largest search services in US, titled *Understanding and Diagnosing Search Response Time: A Provider-side View*. This paper appeared in ACM SIGCOMM Conference, Hong Kong, Aug 12-16, 2013 [3].

Chapter 1

Background and Challenges

Due to the increase of the scale of the content distribution system, the content providers and CDN service providers face enormous challenges in providing users with a fast, scalable, and robust content retrieval experience. In this section, we start with an overview of the existing the content distribution infrastructure. Based on this, we explain how the interplay of different entities poses great challenges in designing a large-scale content distribution system with improved end-to-end performance, and how these challenges can be addressed or mitigated.

1.1 Overview

Figure 1.1 is a general framework widely adopted by most of the existing content distribution systems. It mainly consists of three entities, the users, front-end edge servers, and back-end data centers as provided by the content providers.

Depending on the specific content providers, the type of content, and the scale of the content, back-end data centers can be organized in very different ways. Some (*e.g.*, small-scale services) are organized flat, and only located in a few locations, while others (*e.g.*, large-scale services such as Yahoo!) are in tiered structure, with several powerful backbone data centers interconnecting those less powerful satellite data centers dispersed over wider geographical regions. The link characteristics among the back-end data centers, the replication and communication strategies being used may also vary.

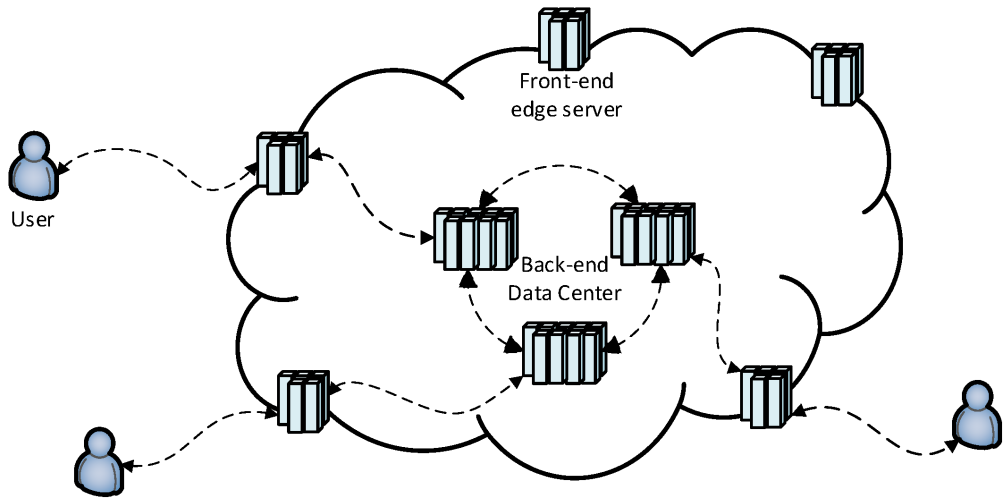

Figure 1.1: A general content distribution system framework.

The back-end data centers are transparent to the users in the sense that the communication between the two entities are intercepted by the front-end edge servers. The placement of the edge servers are intended to improve the latency performance by terminating TCP connections closer to the user (leading to faster growth of the congestion window). They establish one or more long-lived, high-throughput TCP connections to back-end data center servers and multiplex users on these connections.

Due to the huge cost of building large-scale front-end edge servers that sit closer to the users, most content providers opt for delivering their content via a third-party CDN service provider, while a few large content providers such as Google start deploying their own edge servers. As one of our measurement studies will show, building dedicated edge servers exhibits significant performance enhancement. A careful design of the edge server placement strategy can not only have impact on the latencies among users and edge servers, but also the latencies between back-end data centers and edge servers.

In addition to the content providers, and the edge servers, users also play an important role in the overall end-to-end content distribution performance, which has long been neglected by most of the prior works. As we will discuss in later chapters, the user profiles such as the types of content they are interested, the types of browsers being used, and their network characteristics can vary drastically over times of the day, and days of the week, leading to a wide fluctuation on the overall user-perceived performance.

In summary, there are a number of key design decisions in building a large-scale content distribution system such as the placement strategies for front-end edge servers, and the inter-data center communication strategies among the back-end data centers, etc. All of the three entities play critical roles in achieving an optimized design decision. In particular, the inclusion of the user-side characteristics can help the content providers and CDN service providers better interpret the performance variations on the server side.

1.2 Challenges

There are quite a number of design goals we expect to achieve in building a content distribution system, among which, latency, scalability, and robustness have become the key measures that most content providers and CDN service providers have to face in order to support a large-scale user population and provide an enjoyable user experience.

As discussed above, one of the major challenges in achieving these goals originates from the interplay of different entities, *e.g.*, how the placement of edge servers affects the network performance between users and edge servers, as well as that between edge servers and back-end data centers, and how the variation on the user-side characteristics may affect the design decisions on the provider side. In this section, we explain how each individual entity as described above plays a role in achieving these designing goals, and how the change of the properties on one entity might affect the other two.

1.2.1 Latency

The end-to-end content delivery latency has become one of the several key measures of user experience. Even small increases in latency can start affecting the market share of the content providers.

To improve the overall latency, content providers have worked on applying a variety of techniques on their back-end infrastructures. At the application layer, there are various research activities attempting to design more efficient algorithms in locating and fetching the content. Different caching mechanisms (for popular content) are also applied to improve the internal processing time at the individual back-end server; At the network layer, improving the efficiency of inter-data center communication (*e.g.*, by building dedicated connection links, and setting optimized routing policies) can help

minimize the total processing time inside the content providers. In addition, strengthening the hardware facilities such as building more back-end data centers, in particular for places with more user demands can increase the overall capacity of the data centers, and therefore serve the requests much faster, and become more robust to the user request bursts.

As the intermediary of content delivery, CDN service providers also devote a lot of efforts in improving the user latency experience. For instance, they build large-scale edge servers that sit closer to the users. In the meantime, they also design optimized routing strategies that can choose the closest and fastest edge servers to serve the incoming user requests.

Despite the efforts made on the content provider and CDN side, the overall end-to-end performance still suffers from time to time, and varies widely. One of the challenges to optimize the overall user experience from either the content provide side or the CDN side is the unawareness of the interplay of different entities within the content distribution system.

First, the placement strategy used on the edge servers can affect both the users and content providers. While most CDNs put great emphasis on reducing the last mile latency, this effort may on the other hand increase the latency between the edge servers and the back-end data centers.

Second, the design decisions or performance at the content providers may affect the CDN performance. For instance, while most CDNs utilize caching techniques to improve the latency, some content providers (depending on the type of content) are not willing to provide the content from CDN caches, as certain content may contain dynamically generated, or user-specific content. In the meantime, as the front-end edge servers get closer to the users, there exists a lower limit on the last mile latency that CDN can achieve. If CDN attempts to further reduce the last mile latency, the processing time at the content provider becomes dominant, and therefore leads to very little improvement on the overall end-to-end user-perceived latency.

Third, the change on the user characteristics such as the fluctuation of the user interest, as well as the browser and network speed at different times of the day, and days of the week can also play critical roles in driving the server-side decisions, *e.g.*, sending content of lower resolution, or compressing the content for users with worse

network conditions.

1.2.2 Scalability

As the amount of content as well as the user demand increase, scalability has become one of the key goals that most large-scale content providers are trying to achieve. The main challenges in achieving scalability are mainly two aspects.

First, due to the size of the content, a careful design of the storage and replication of the content becomes critical. Maintaining a single copy of all the content in one data center not only has the risk of losing the data but also makes the content fetching sluggish, as all the requests have to be processed within this data center that can potentially extremely far away from the users. That being said, we cannot replicate the content in all data centers as well, as it incurs huge maintenance cost. Choosing what to replicate and how often to perform the replication can be another challenge. While less frequent replication can lead to data inconsistency, excessively frequent replication of all the content can consume too much of the back-end server capacity. As a result, more frequent replications of selective content can be more cost-effective.

Second, the content should be organized in such a way that it can be easily located. For this purpose, the content can be organized in semantically tier-ed structure, or clustered based on the type of the content (*e.g.*, news related, video related, etc.). In this way, if a user request requires content from all different types with a similar semantic meaning, it will be easier for the back-end servers to process them in tiers, and in parallel. This is fairly common in the search context, in which a user search request can involve results from different areas, including maps, restaurant, video, finance, and so forth.

Beyond that, user-side characteristics may as well help improve the scalability performance. For instance, based on the historical user access patterns, part of the content can be prefetched. For the top popular content, they can be fetched directly from cache.

1.2.3 Robustness

As the scale of the content distribution system gets larger, chances are that more and more elements within the system are subject to failures. To design a system of high

robustness, content providers should quickly detect, diagnose, as well as providing remedies to the failures.

Detection of failures or anomalies is more than just alerting on certain measures when it is beyond a fixed threshold. Performance measures are normally a mixing behavior of multiple factors. For instance, the latency measure can be affected by the long-term evolution trend, the seasonal (times of the day, and days of the week) behaviors (*e.g.*, from the users), and some real anomalies due to failures (*e.g.*, at the data center). Understanding each of the component, and the factors associated with each of them, as well as teasing out the anomalies from other factors can be challenging.

Diagnosing the root cause of the anomaly is even more challenging than the detection, as it involves the investigation of all the components within the system, from the user side (including the ISP routing), to the CDN side, and to the back-end data center side. For this purpose, we need to carefully design and collect measures from different part of the system, and investigate them at both coarse and finer granularity. Oftentimes, it requires the operational information from the operators as well for a better understanding of the failures.

Again, user characteristics also plays an important role in diagnosis. Not all anomalies are caused by server-side failures. Some anomalies can be resulted from user-side attacks on the back-end infrastructure. Others are attributed to the change of user behaviors due to certain holidays, or big events. For instance, users tend to send richer search queries during holidays, which can lead to an anomaly in the overall latency as measured by the content providers. This type of anomalies is false positive, which does not require any action items on the content provider side. Hence, understanding these effects can help operators less affected by the noise coming from the users.

Providing immediate remedies as the failures occur is equally critical. This may include a failover of the affected traffic to another back-end data center if the problem is diagnosed to be failures in the data center, alerting CDN service providers if they are the ones to blame. In the longer terms, if the majority of the failures are attributed to the failures at the back-end data center, content providers can build data centers with more capacities and better availability. If network issues become the main concern, depending on the nature of the network failure, a better ISP peering might help. On the other hand, if most failures are caused by user-side attacks, designing stronger attack

filtering algorithms will be essential.

We will discuss in later chapters how we address the challenges discussed here in detecting and diagnosing the failures. Our study shows using search a case study that there are equal chances for the anomalies to happen within the back-end data centers, inside the networks, and at the user side. Therefore, they are three equally important fronts that we should work towards a robust content distribution system.

Chapter 2

Yahoo! Inter-Data Center Traffic Characteristics

2.1 Introduction

Recent years have seen unprecedented growth in the data center driven technologies and services. Various organizations are now sourcing their computing to "cloud-based" infrastructures. Therefore, large scale data centers and associated cloud services are developed and deployed by various organizations and service providers to store massive amounts of data, and enable "anywhere, anytime" data access as well as computations on the data. Further, for scalability, robustness and performance (e.g., latency), multiple data centers are often deployed to cover large geographical regions. For instance, Microsoft, Google, and Yahoo! own large scale data centers that are located in different geographic locations around the world.

While there are a few recent studies [4, 5] regarding the traffic characteristics within a single data center, little is known about the inter-data center (D2D) traffic dynamics among multiple data centers. Just as the studies of traffic characteristics within a data center, such as workload distribution and where congestion occurs, helps the design and management of data centers, we believe that better understanding of the traffic characteristics between multiple data centers (within a single service provider, e.g., a content provider) and their interactions with client-triggered traffic is critical to effective operations and management of multiple data centers. For instance, such understanding can help in deciding what and how services should be deployed across multiple data centers, what caching and load-balancing strategies [6, 7] should be employed, and how to manage the traffic in the wide-area network backbone connecting the data centers to optimize performance and minimize operational costs [6, 7].

In this chapter we present a first study of inter-data center (D2D) traffic characteristics using the anonymized NetFlow datasets collected at the border routers of five major Yahoo! data centers. Our contributions are multi-fold. First, we develop novel heuristics to infer the Yahoo! IP addresses that are involved in data center-client (D2C) traffic and localize their locations from the anonymized NetFlow datasets. Based on several key observations regarding traffic directions and router interfaces, we develop an effective methodology to extract and separate inter-data (D2D) traffic from data center-client (D2C) traffic, and analyze the characteristics of both D2D and D2C traffic and their correlations. Our analysis reveals that Yahoo! organizes data centers in a

hierarchical way. In "satellite" data centers, D2D traffic is strongly correlated with the client traffic. In "backbone" data centers, we classify D2D traffic into two categories: i) *client-triggered* D2D traffic, i.e., D2D traffic triggered by the front-end "customer-facing" services such as web search, email, online chat, gaming, video, and so forth; ii) *background* D2D traffic, i.e., D2D traffic due to internal tasks such as routine background computation (e.g., search indexing), periodic data back-up, and so forth. Using novel port based correlation analysis, we are able to further separate these types of D2D traffic, and study their respective characteristics. We find that background D2D traffic has smaller variance, with no significant trends over the day; on the other hand, client-triggered D2D traffic exhibits varying trends over the day. Furthermore, we show that several D2C services are strongly correlated with each other. These correlations among different services have important implications for distributing different services at multiple data centers. For instance, services with highly correlated traffic can be served from the same data center to minimize the inter-data center traffic.

To our best knowledge, our work is the first study of inter-data center traffic characteristics of a large global content provider. It sheds light on the interplay of multiple data centers and their traffic dynamics within a large content provider. Though the D2D and D2C traffic characteristics studied in the chapter may be specific to Yahoo! and the services it provides, our methodology is nonetheless general, and can be applied to understand the D2D and D2C traffic characteristics of any other large content provider or cloud-service provider. All in all, we believe that our work provides useful insight to data center designers and operators as well as researchers.

The remainder of the chapter is organized as follows. In Sec. 2.3 we provide the overview of the datasets and Yahoo! data centers. Sec. 2.4 presents the methodology for separating Yahoo and non-Yahoo IP addresses, and analysis of inter-data center traffic are presented in Sec. 2.5. Finally, we provide a discussion of the implications for our findings in Sec. 2.6 and conclude the chapter in Sec. 2.7.

2.2 Related Work

As mentioned earlier, there have been a few recent studies [4, 5] regarding the traffic characteristics within a single data center. In [4], authors provide both macroscopic and a microscopic view of the traffic characteristics and congestion conditions within data center networks. In [5], authors analyze the end-to-end traffic patterns in data center networks, and examine temporal and spatial variations in link loads and losses. On the other hand, little is known about inter-data center traffic characteristics. Similarly in [7], the authors study the YouTube data center traffic dynamics using the Netflow data collected at a tier-1 ISP, with the emphasis on inference of load-balancing strategy used by YouTube and its interaction and impact on the ISP network. Due to the nature of data used, the traffic seen is primarily D2C traffic, and limited to the perspective to a single ISP. To our best knowledge, our work is the first attempt at analyzing and characterizing inter-data center traffic characteristics; we also develop novel methods for separating D2D traffic from D2C traffic, and for further separating background D2D traffic and client-triggered D2D traffic.

2.3 Overview of Yahoo! datasets

In this section we provide the overview of the Yahoo! data centers and their connectivity. We also describe the network flow datasets [8] used in this study. Further, to facilitate the discussion in this chapter we classify the flows into several meaningful categories which is described later in the section.

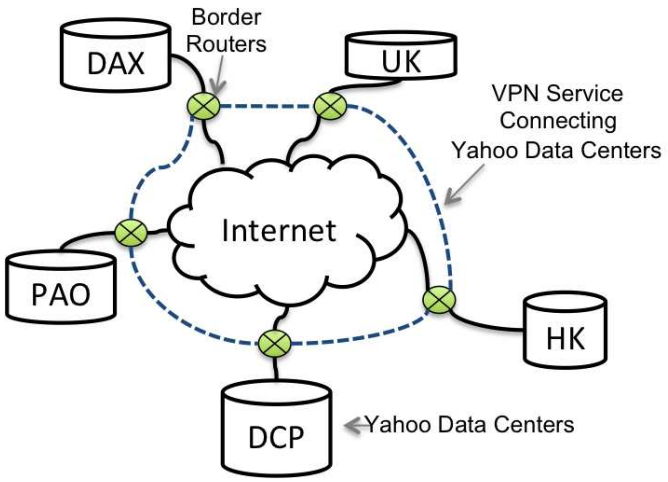

Figure 2.1: Overview of five major Yahoo! data centers and their network connectivity.

In this study we consider five major Yahoo! data centers which are located at Dallas (DAX), Washington DC (DCP), Palo Alto (PAO), Hong Kong (HK), and United Kingdom (UK). DAX, DCP and PAO are located in US, and provide most of the core services such as web, email, messenger and games, etc. They are also the largest Yahoo! data centers in terms of the amount of traffic exchanged. At each of the data centers, Yahoo's border routers connect to several other ISPs to reach its clients and other data centers. These data centers are also directly connected to each other through a private network service(e.g. VPN, leased lines etc), and hence may carry traffic for each other through this private network. Fig. 2.1 provides an overview of the Yahoo! data centers and their connectivity.

Our study is based on NetFlow datasets collected at one of the border routers at each of the locations mentioned. Unlike the datasets used in the previous studies related to data center traffic analysis (such as [4, 5]) the NetFlow datasets used in

our study provide us with not only the profiling of Yahoo! to "client"[1] traffic, but also the traffic exchanged between different Yahoo! data centers, which we believe is the first such work that sheds light on the inter-data center traffic characteristics for a large content provider. The network flow data collected at each border router, includes both the inbound and outbound traffic. Each record in the NetFlow data contains a "sampled flow" information, which includes following fields: a) *timestamp*, b) source and destination IP addresses and transport layer port numbers, c) source and destination interface on the router, d) IP protocol, e) number of bytes and packets exchanged.

An important challenge with the datasets is that the IP addresses in the network flow traces are permuted to hide the identities of the Yahoo! users. However, prefix-preserving schemes [9, 10] are used in permutation, i.e. if an IP address $a.b.c.d$ is permuted to $w.x.y.z$ then another IP address $a.b.c.\bar{d}$ is mapped to $w.x.y.\bar{z}$. Due to this reason, through out this chapter we represent summarized IP address based statistics using /24 IP prefixes. Also, we use the term "client" to represents the non-Yahoo hosts connected to Yahoo! servers. These hosts may be the actual Yahoo! users connecting to Yahoo! servers to access various services, or other servers connecting to Yahoo! servers, such as other mail servers may connect to Yahoo! mail servers to exchange emails.

Classification of Flows: In order to facilitate the discussion in this chapter, we classify the flows collected into following two categories:

i. *D2C traffic*: The traffic exchanged between Yahoo! servers and clients.

ii. *D2D traffic*: The traffic exchanged between different Yahoo! servers at different locations.

A border router at a given location may also carry D2C and D2D traffic for other locations. We refer to these types of traffic as *transit D2C traffic* and *transit D2D traffic*, respectively. Accordingly, we also define two types of Yahoo! prefixes. One is the Yahoo! prefixes that are involved in the D2C traffic, referred to as *D2C prefix*. The other is the ones that are involved in D2D traffic, denoted as *D2D prefix*. Note that a Yahoo! prefix can potentially be involved in both D2C and D2D traffic. In fact, we will see in the later sections that there is significant amount of overlap in the prefixes belonging to each category.

[1] We refer to non-Yahoo hosts connecting to Yahoo! servers as clients unless specified.

2.4 Identifying Yahoo! Prefixes

Understanding D2C and D2D traffic characteristics is not possible without identifying the IP addresses used by Yahoo! hosts, and therefore, presents a key challenge to our analysis. In this section we describe our heuristics to infer the IP addresses used by Yahoo! hosts using basic features of the traffic seen at border routers of each data center.

2.4.1 Challenges

Inferring original information from anonymized data has already been studied in several other previous studies e.g. [11, 12]. However, these solutions are specific to the datasets, and do not apply for sampled NetFlow datasets. For instance, the inference techniques discussed in [11] require ARP traffic information, hardware addresses in the link layer, as well as other specific header and transport protocol requirements. In addition, they also make use of a lot of other auxiliary public information. Furthermore, authors explicitly note that NetFlow data is invulnerable to their inference techniques because of the lack of required header and transport protocol information. In contrast to the previous work, we need to look at all the services provided by one content provider, with very limited information presented in NetFlow data.

In addition to the limited information provided by the data, there are also several challenges specific to our problem that we need to address. These challenges include the following. 1) Our goal is to study the characteristics of both D2C and D2D traffic. However, the IP addresses involved in each type of communication may have quite different network characteristics, which led to a two-step process in identifying the Yahoo! prefixes. Where, in first step we separate Yahoo! IP addresses from non-Yahoo IP addresses in the D2C traffic, and in the second step we further extract the D2D IP addresses. 2) As we have observed, the border router at one location carries not only its own traffic (i.e. the traffic belonging to one of the hosts at that data center), but also transit traffic for other Yahoo! locations, which does not involve the hosts from the same location. Due to such "transit traffic" carried by Yahoo! border routers for the other Yahoo! locations, Yahoo! prefixes that belong to one location can also appear in the data collected at other Yahoo! locations. Therefore, heuristics to localize the

inferred Yahoo! prefixes is needed. 3) Some of the IP addresses used in the D2D traffic may not be announced to other ISPs during the BGP announcements, and therefore it is hard to use the publicly available auxiliary resources, e.g. RouteViews [13], to help inference the data or to validate our inferred results. To address these limitations, we provide novel approaches to inference the NetFlow data. In particular, it is a two-step approach, which consists of identifying the D2C and D2D prefixes, respectively.

2.4.2 Identifying Yahoo! D2C Prefixes

We separate Yahoo! prefixes from the client prefixes in D2C traffic based on the degree and ports observed in the flows. A prefix is considered Yahoo! D2C prefix if it talks to large number of other prefixes, and if a large fraction of their traffic uses the TCP ports used by several popular services provided by Yahoo! (such as email, web, messenger etc.). There are two thresholds implied in this heuristic, which are defined as follows. We choose top α prefixes out of all the prefixes based on how many other prefixes these prefixes talk to. Next we choose the prefixes for which at least β fraction of traffic is received at (or sent from) the popular Yahoo! ports. Furthermore, it is important to note that we need to choose the parameters in a relatively conservative manner such that prefixes we get are mostly Yahoo! prefixes, so as to minimize the number of non-Yahoo IP addresses classified as Yahoo! (false negative).

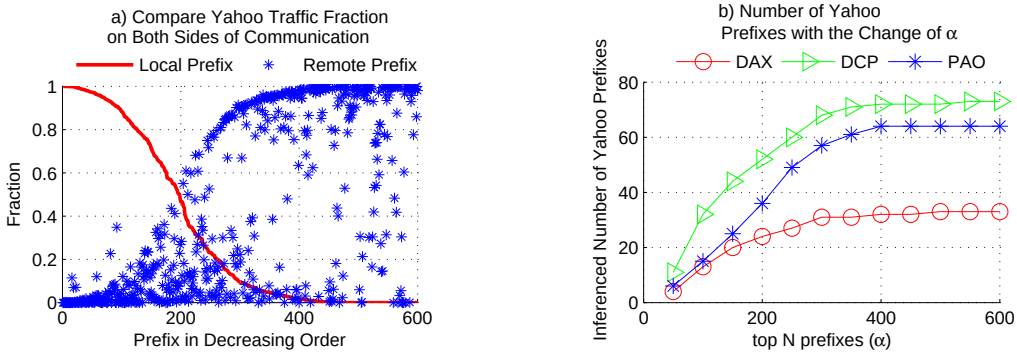

Figure 2.2: Choosing parameters α and β.

To choose the proper value of β, we first fix $\alpha = 600$, considering top 600 prefixes[2] .

[2] Using routeviews [13] we found that the number of /24 prefixes announced by Yahoo! ASes at different location is in the range 50-500

In Figure 2.2(a), red continuous line shows the fraction of traffic for the top 600 prefixes which use Yahoo! service ports, and blue dots represent the fraction of traffic containing Yahoo! ports for the prefixes that each top 600 prefix talks to. Therefore, in this figure, we compare the fraction of traffic that uses Yahoo! service ports on the same side as top 600 prefixes, with the fraction of traffic on the other side of these prefixes. From this figure, we learn that prefixes in the left region, $\beta > 0.5$ are more likely to be Yahoo! prefixes, talking to other prefixes that mostly communicate using popular client ports. In contrast, prefixes in the right region are more likely to be client prefixes. Therefore, we choose $\beta = 0.5$ for DAX.

In order to see how sensitive our D2C prefix inference result is to the change of α value, we experimented with different values of α between 50 to 600, while keeping the value for $\beta = 0.5$. In Fig. 2.2(b) we show the inference results for three data centers located in US. In this figure, x-axis shows the different values for parameter α and y-axis shows the number of candidate prefixes. We see that candidate prefix set grows initially with the increase in α, however, it becomes stable after α goes beyond 400, and does not increase much by beyond this value. Hence it shows that our D2C inference algorithm is not very sensitive to parameter α, whereby makes easier to find an appropriate value for α.

2.4.3 Localizing Inferred D2C Prefixes

The above process only identifies IP addresses (prefixes) that belong to Yahoo!, but could not assign appropriate location to each prefix, due to the challenges mentioned earlier in the section. To assign a correct location to each prefix, we utilize the traffic direction observed due to the use of early-exit routing, which is prevalent in the Internet [14, 15]. Because of the early-exit routing, the D2C traffic sent from a client and destined to a host at any given data center may enter in the Yahoo!'s network from any border router at another location, and carried through Yahoo!'s own private network. In contrast, the D2C traffic sent from a Yahoo! host to the client always exits Yahoo! network from the same location, and therefore is not carried through the Yahoo! network connecting different locations. We use this observation to locate a Yahoo! IP prefix to its correct data center location. Finally, we assign a location to a Yahoo! IP prefix only if it appears in both incoming and outgoing D2C traffic seen at that location.

2.4.4 Identifying Yahoo! D2D Prefix

The heuristics discussed so far are only applicable in identifying the D2C prefixes, however, these heuristics can not extract all the D2D prefixes. It is because prefixes in D2D traffic only talk to a limited number of other Yahoo! prefixes, and the ports used by them may not be listed in the well-known Yahoo! service ports. In addition, unlike asymmetric routing observed in D2C traffic, D2D traffic is mostly symmetric, and carried in Yahoo!'s private network. To infer the D2D prefixes, our heuristics are based on the key observation that there are two types of physical interfaces that play specific roles on each border router.

a. *Foreign interfaces*: All the traffic (including D2D and transit D2C traffic) sends to (or receives from) other data-centers are exchanged through these interfaces on the local border router.

b. *Local interfaces*: These interfaces are only connected to the local hosts at each location.

Since different data centers exchange traffic only through foreign interfaces, a Yahoo! D2D prefix must appear in the traffic that is exchanged through these interfaces. Moreover, to further exclude the possible transit D2C traffic that is also exchanged through the same set of interfaces, a prefix is considered Yahoo! D2D prefix only if its traffic is also symmetric, i.e. both the incoming and outgoing traffic are exchanged through these interfaces. Finally, the local interfaces further help us in completing the list of Yahoo! prefixes at each location.

2.4.5 Inference Results & Validation

Inference Result:

Table 2.1: Inference Result.

	D2D prefix	D2C prefix	overlap prefix	D2D IP	D2C IP	overlap IP
DAX	104	108	104	8927	8056	5974
DCP	451	556	446	25299	22020	14257
PAO	280	289	277	15415	12972	7974
UKL	34	35	34	2800	3361	2278
HKX	51	57	51	2226	4795	1754

Using the heuristics proposed in this section the inferred prefixes (and IP addresses) are summarized in Table 2.1. It shows the number of prefixes/IPs participating in the D2C traffic and D2D traffic, and the number of overlapping prefixes/IPs in both categories. As we can see, most of the D2D and D2C prefixes overlap. Moreover, the three US locations have more D2D IP addresses than D2C IP addresses, while UK and HK have more D2C IP addresses, implying that more IP addresses are involved in background D2D traffic in the three main data centers in US.

Validation:

Table 2.2: Comparing validated results and inference results.

	DAX	DCP	PAO	HK	UK
inferred	108	561	292	57	35
IP addresses from local interfaces	106	472	271	20	34

We validate our results by using testing against some basic constraints. As discussed before, each location have the local interfaces that only connect to the local Yahoo! data centers. Therefore, we first get all the possible local interfaces using our inference results, and see if the union of all the prefixes appearing on these interfaces are close to the number of prefixes we have inferred for each location. If our inferences are not correct, then there is a good chance that we will get a much smaller set of prefixes than extracted by our inference mechanism. Using this validation mechanism we summarize the resulting number of inferred prefixes we get for each location and the union of all the IP addresses appearing at the local interfaces in Table 2.2.

In addition, we also talked to operators at Yahoo! to verify the correctness and completeness of our inference results. Among all the Yahoo! prefixes, our heuristic based inference methodology extracted around 95% (DAX), 95% (DCP), 75% (PAO), 100% (UK), and 75% (HK) of the total prefixes for each location. Further, only less than 5% non-Yahoo! prefixes were classified as Yahoo! (i.e. false negative) and around 5% Yahoo! prefixes were assigned incorrect location. Most of our inference results seem correct, except that we get more than the number of prefixes HK owns. It is not because of the failure of our algorithm, but that HK also carries some traffic from other Asian countries. So these prefixes are coming from other (small) Asian Yahoo! location. However, it will not have negative impact on our D2C and D2D traffic analysis, because

these prefixes have been validated to be Yahoo!'s prefixes (i.e. false positive).

2.5 Traffic Characteristics

In this section we present various characteristics of the traffic seen at the border routers using the inferred D2C and D2D IP prefixes of Yahoo! hosts. In the following we begin with the aggregate statistics for Yahoo! traffic. Next, we present detailed characteristics of D2C and D2D traffic, respectively. In addition, we also present the results on the interaction of D2C and D2D traffic using the port based traffic correlation.

2.5.1 Aggregate Traffic Statistics

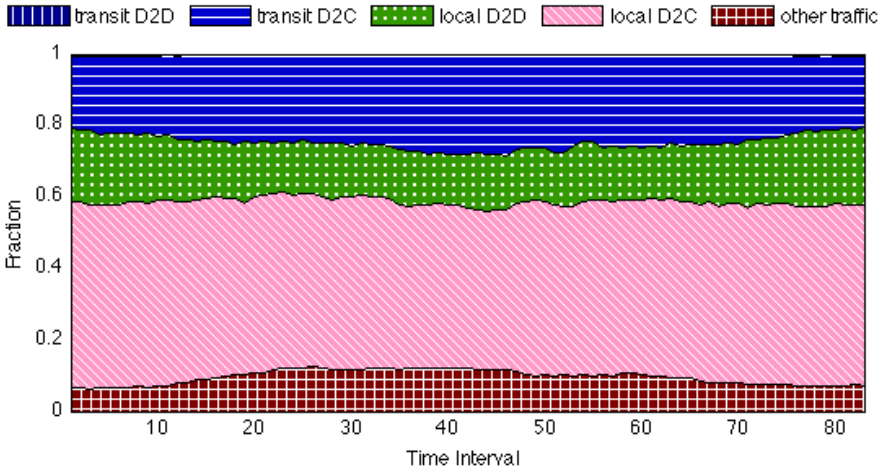

Figure 2.3: The fraction of different types of traffic at DAX data center.

As described in Sec. 2.3, we classify the traffic seen at the border routers into two categories: i) D2C traffic, and ii) D2D traffic. We further divide each category into two sub-categories, depending upon whether it is destined to the local data center or it is transit traffic seen at that location. The fraction of each type of traffic seen at the DAX data center is described in Figure 2.3. As seen in this figure more than 50% of the traffic is local D2C traffic at DAX, 20% of the traffic is local D2D, while transit D2C traffic contributes to 25% of overall traffic at DAX. Moreover, a very small amount of traffic is transit D2D. It shows that a significant amount of the D2C traffic seen at the DAX location belongs to the transit D2C, which is expected to be as small as possible. Furthermore, we are not able to classify the remaining 10% of the traffic. It is due to

the fact that we define client as all the IP addresses outside these five locations. Since there can not be any client to client traffic, this traffic must be transit D2C and D2D traffic destined to other Yahoo! locations.

2.5.2 D2C Traffic

Yahoo! provides multiple services including email, web-portal, instant messaging, news, music and video. These services are distributed across different data-centers, where each data-center does not necessarily serves all the services. Furthermore, different types of services are also likely to interact with each other. Some services are likely to be correlated with each other, while others may be independent of others. In the following we describe the traffic characteristics for each category of services, and the correlation among them.

D2C Service Classification

Table 2.3: Seven categories of D2C services.

D2C service	Port numbers
Email	110, 995, 465, 143, 587
SMTP	25
DNS	53
Messenger	5000, 5001, 5100, 5050, 5061
News	119
Video	1935
Game	11999
Web	80, 443

We identify the Yahoo! services by using the transport layer ports used in the traffic[3] . There are 17 popular server ports observed in our data, which contribute to more than 95% of the aggregate D2C traffic. As we see in Table 2.3 most of these ports are well-known such as web and email, while a few of them are specific to services provided by Yahoo! e.g. Yahoo! messenger and video ports. The ports which do not

[3] We consider port numbers for this classification, as no additional information such as application headers, packet payload, etc. is available to us. Nevertheless, it provides a coarser level classification of services and sufficient for understanding general characteristics of various services.

belong to well-known services are identified using entropy of the ports they talk to (see Sec. 2.5.3 for details), as well as from the publicly available sources [16, 17]. These 17 ports mainly fall into 7 service categories. The mapping of each service category and the corresponding ports providing this service is listed in Table 2.3.

Table 2.4: The number of IPs providing each D2C service and the overlapping number of IPs between each pair of services.

	email	DNS	IM	news	video	game	web	SMTP	unique
email	83	8	2	3	1	0	62	67	10
DNS	8	131	2	2	1	0	27	22	102
IM	2	2	235	60	1	1	163	64	71
news	3	2	60	66	0	0	64	64	2
video	1	1	1	0	87	0	67	2	20
game	0	0	1	0	0	2	1	0	1
web	62	27	163	64	67	1	3773	262	3333
SMTP	67	22	64	64	2	0	262	699	424

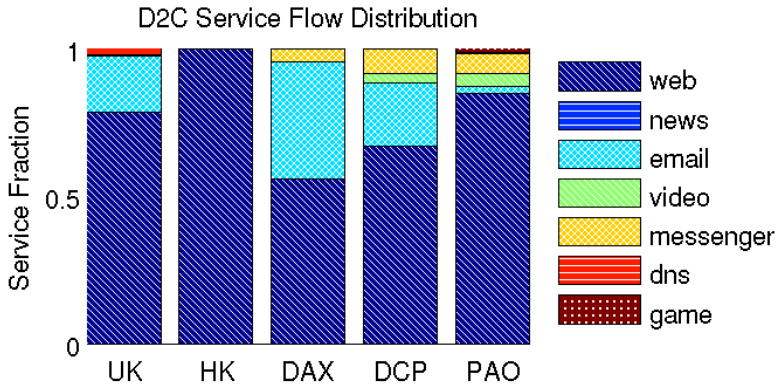

Figure 2.4: The distribution of D2C services in each location.

In Figure 2.4 we compare the fraction of traffic belonging to each D2C service for all the five data centers. As seen in this figure, the aggregate D2C traffic is mainly dominated by the web services, which is not surprising as most of the services provided by Yahoo! have web-based interface, and these services are provided at all five locations. On the other hand, instant-messaging (IM), video, and game services have smaller but significant contribution to D2C traffic at all three US locations. Moreover, the

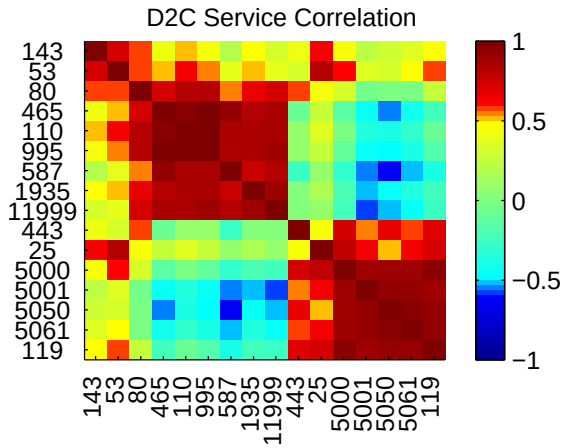

Figure 2.5: Cross-correlation between each pair of D2C services.

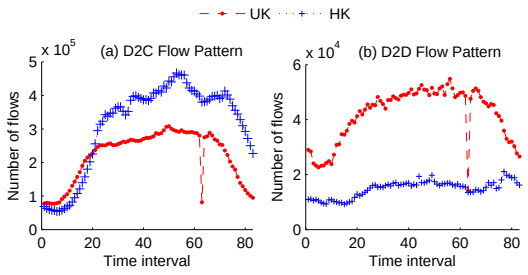

Figure 2.6: The D2C and D2D flow patterns during one day in UK and HK.

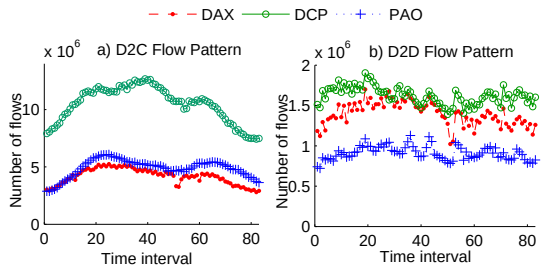

Figure 2.7: The D2C and D2D flow patterns during one day in US locations.

choice of location for different services can be affected by many factors such as regional demand, cost of infrastructure and the nature of service itself. Also location based services replicate content at multiple data centers to provide better performance [18, 19]. Table 2.4 shows the number of IPs providing each type of service in DCP data center. We separate port 25 (SMTP) from rest of the email category due to the fact that this is mainly used between Yahoo! mail servers, or between Yahoo! and other service providers' mail servers such as Gmail or Hotmail. On the other hand, other email port numbers are used by clients to directly interact with Yahoo!. The diagonal entries in the table show the number of IPs providing each service as specified in row or column, and the non-diagonal entries show the number of overlapping IPs between two services as specified per row and column. In the last column, we also list the number of unique IPs

providing each D2C service. As seen in this table, some of the IP addresses only provide one type of service (see the "unique" column), a large number of them provide multiple services on the same server IP address. From the table we learn that many of the web, SMTP, and DNS services are mostly served using a dedicated set of IP addresses, while the remaining services share IP addresses with other services[4] .

Cross-Correlation among D2C Services

Though D2C services can be categorized into 7 groups, we find that some of them are strongly correlated (positively or negatively) with each other, while others are independent of each other. We compute the pair wise temporal correlation of each service category to get a better understanding of the interplay among different types of D2C services. Figure 2.5 shows the correlation between each pair of D2C services in the PAO data center. In this figure, both x-axis and y-axis represent the list of D2C server ports observed in this location. The colored cell corresponding to a pair of services as specified in x-axis and y-axis shows the correlation between them. It turns out that the D2C service ports are clustered into 2 major traffic patterns. The first group consists of several email related ports, and the other messenger ports. These correlations among different services have important implications for the data center service providers (Yahoo! in this case) to distribute different services at multiple data centers. For instance, services with highly correlated traffic can be served from the same data center to minimize the inter-data center traffic. Further, knowing these correlations information may help them apply more efficient load balancing strategies, and therefore make better use of their computing resources.

2.5.3 D2D Traffic

In this subsection, we will first describe the frequency and entropy based technique to identify the popular server ports used in the D2D communication. Next, we describe the D2D traffic characteristics, and its correlation with the D2C traffic.

[4] It can happen due to a variety of reasons, such as a single host machine might be running multiple different server instances or a NAT based forwarding is used to divide the traffic to multiple physical(or virtual) servers. It is also likely that these IP addresses are simply frontend servers.

Identifying D2D port

Unlike most of the D2C ports, not all D2D ports are well-known or publicly available. However, the D2C and D2D traffic are exchanged in a similar fashion, namely, following the client/server communication paradigm. That is, in each flow one end-point uses a server port and the other uses a client port. Based on this observation, a port p is considered D2D port only if it meets two constraints. First, p is frequently used in D2D traffic. Second, entropy for the distribution of other ports it talks to is close to 1. We consider top N (in our case, 1000) frequent ports p talks to, and compute the entropy based on the frequency (P_i) of each port appearing on the other end of the flows for p. If it is close to 1, then it is considered as a server port used in D2D traffic, talking to a number of random client ports. By imposing these two constraints, we have found 37 such D2D ports, which cover more than 95% of the overall D2D traffic. Among the 37 ports, the top frequently used ports include 80, 25, 1971, 14011, 5017, 5019, 14020, and 14030.

D2D Communication Patterns

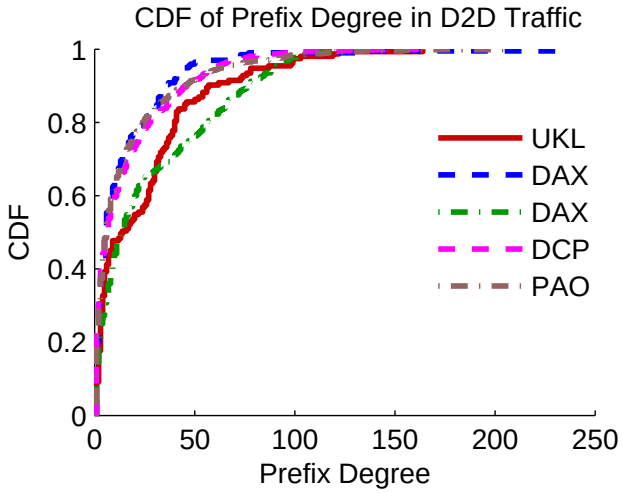

Figure 2.8: The prefix degree distribution.

To study the aggregate communication patterns among D2D prefixes, we look at the degree distribution for each Yahoo! prefix seen in the D2D communication. Here,

we define the degree of each prefix as the number of unique IP prefixes that it talks to. This can be useful in simulating various D2D traffic workloads, to evaluate the network performance. Figure 2.8 plots the cdf of the prefix-level degree distribution for the each location. As seen in this figure, the prefix-level degree distribution in D2D traffic follows a power-law distribution. Moreover, we observe that each D2D prefix mostly talks to the same set of D2D prefixes in other locations using the same set of D2D ports in our one-day data. This implies that communication patterns among D2D prefixes are quite stable.

Cross-Correlation between D2C & D2D Traffic

Table 2.5: Correlation coefficient between D2D and D2C traffic.

	DAX	DCP	PAO	HK	UK
Correlation	0.81	0.11	0.65	0.87	0.97

Figures 2.6, 2.7 show the distribution of aggregate D2C and D2D traffic seen in each of the data center locations over time. The x-axis here shows the series of time intervals (15 min for each time interval) during one day. There are 96 15-minute intervals in a whole day. However, the first three and a quarter hour of network data was lost during the collection. Therefore we only show 83 intervals in our analysis. The y-axis shows the number of flows seen in a given interval. The correlation coefficient between the two types of traffic is shown in Table 2.5. When compared with our inference results listed in Table 2.1, we see that D2C and D2D traffic are highly correlated at HK and UK data centers. On the other hand, they are less correlated at the DAX data center, and the PAO data center has only mild correlation, while there is no correlation between D2C and D2D traffic at the DCP data center. The larger the scale of the data center, the less correlated between the D2D and D2C traffic. Interestingly, most of the Yahoo! IP addresses seen at HK and UK data centers appear in both D2C and D2D traffic, which explains the strong positive correlation, as discussed in Sec 2.4.5. These act more like the "satellite" data centers in the sense that they have smaller scale and the D2D traffic is mostly triggered by the D2C demands. On the other hand, for the three US locations, the D2C traffic has shown varying trends at different times of the day, while D2D traffic does not show any dominant trends. Moreover, we observe that data

centers in US locations carry transit traffic for the UK and HK locations, as well as among themselves. In contrast, we do not see any transit traffic in UK, and only a little in HK.

The data centers in US seem to act more like a "backbone" data centers. As we have already seen in Sec 2.4.5, there are more IPs involved in the D2D traffic in these data centers. Intuitively, D2D traffic in the US locations may be affected by many factors. For example, it can be affected by both the D2C traffic in that location, and the D2C traffic in other locations. There may also exist some background traffic, e.g. regular maintenance or content replication, which might be independent of the D2C traffic. Based on the underlying causes of D2D traffic, we define two major types of D2D traffic:

a. *D2C-triggered* D2D traffic, which is triggered by D2C traffic. If it is triggered by the local D2C traffic, it is defined as local D2C-triggered D2D traffic. If it is triggered by the D2C traffic in other locations, it is foreign D2C-triggered D2D traffic.

b. *Background* D2D traffic, which includes the regular traffic exchanged among the back-end servers, and the traffic incurred by other network events, such as network failure, etc.

The difference between the two sub-types of D2C-triggered D2D traffic is that the local D2C-triggered traffic will actively generate request traffic from a local host to a remote host, i.e., the remote Yahoo! host uses D2D server ports. In contrast, foreign D2C-triggered traffic will trigger D2D traffic that is requested by a Yahoo! server from other data centers, implying that local Yahoo! host uses D2D server ports in the data exchange. We extract the D2D traffic that is triggered by (both local and foreign) D2C, via correlating the D2D traffic at each D2D port with D2C traffic at different ports in each location. The D2D traffic that uses the set of D2D ports that are highly correlated with the D2C ports are considered as the local or foreign D2C-triggered D2D traffic. The D2D traffic that does not use any of the ports that are highly correlated with the local and foreign D2C traffic, is considered as the background D2D traffic.

Our findings show that D2C services are only correlated with certain specific D2D ports. Furthermore, most of the D2C services that are highly correlated with the D2D ports are email-related services. This is quite reasonable, as the email service usually requires a lot of data stored at the back-end data center servers. While for services

such as messenger and game, they do not need such supporting data from the back-end servers.

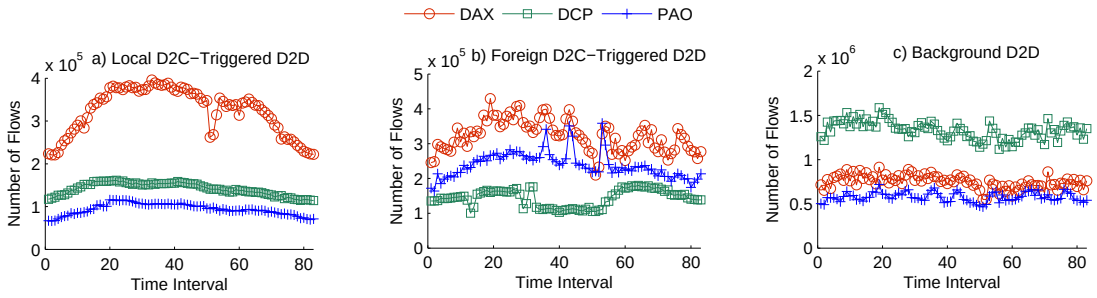

Figure 2.9: Comparing three types of D2D traffic.

Table 2.6: The normalized standard deviation for D2C-triggered D2D and background D2D traffic.

	DAX	DCP	PAO
D2C-triggered D2D traffic	0.1429	0.0887	0.1427
background D2D traffic	0.0994	0.0761	0.0897
$\frac{D2C-triggeredD2D}{backgroundD2D}$	1.4373	1.1669	1.5903

Finally, we extract the background D2D traffic by excluding the local D2C-triggered D2D traffic as well as the foreign D2C-triggered D2D traffic from the aggregate D2D traffic seen at each location. In Figure 2.9, we compare the background D2D traffic with the two types of D2C triggered traffic. It shows that the background D2D traffic is dominant in the aggregate D2D traffic. Moreover, D2C triggered traffic has increasing or decreasing trends depending upon the time of the day. On the other hand, background D2D traffic does not have any significant trends over the day, but it has smaller variance compared with the D2C triggered traffic. To quantify the variance of the two types of D2D traffic, we use the metric of normalized standard deviation, which normalizes the standard deviation by the mean value of the flow. The results are summarized in Table 2.6. As seen in this table D2C-triggered D2D traffic has larger normalized standard deviation than background D2D traffic, which implies more stable behavior for background D2D traffic over time.

2.6 Discussion and Implication

Our findings in this chapter have important implications not only to network researchers, but also to the network operators or data center designers. In this section we discuss the various findings made by our study, and their various implications.

Data Inference: There are very limited number of publicly available datasets to understand the inter-data center traffic characteristics. However, most of these datasets are anonymized due to various concerns related to privacy of users, security of data center infrastructure etc. These obstacles severely limit the usefulness of these datasets. To overcome these challenges, we developed some simple and intuitive heuristics, which proves to work far better in terms of accuracy than some complicated ones, such as correlating the timestamp between different flows etc., which is commonly used in traffic analysis and correlation [20]. Because of its simplicity, the alogrithms can be easily adjusted or directly applied to any other anonymized datasets from other content providers.

Flow Classification: Since most of the existing work related to network traffic analysis focuses on single data center, and content providers are usually not willing to publicize their data center locations and internal topology, little is known about the types of traffic we might see among different data centers within one content provider. Our study shows the presence of various types of traffic, such as client to server traffic, traffic among servers at different data centers, etc. However, it is a challenging task to separate these flows from the aggregate network traffic. Using the correlation based techiques developed in this chapter, we have provided an initial estimate of such traffic and their characteristics.

Traffic Correlation: In general, data centers are used to provide various services with different characteristics. Due to the co-existence of several services, it becomes difficult to understand how the traffic for different services interact with each other. On the other hand, a better understanding of these interaction can help in developing better strategies to deploy various services across data centers, to optimize their network performance. For instance, D2C services with highly correlated traffic can be served from the same data center to minimize the inter-data center traffic, which has shown to be quite large in Yahoo!. Moreover, by correlating D2D and D2C traffic, we infer that

Yahoo! uses a tiered structure in deploying their data centers, with several "satellite" data centers mostly distributing services, and "backbone" data centers having huge amount of background D2D traffic going on. By inferring and extracting background D2D traffic, we are able to estimate how much background traffic may exist within a content provider. By analyzing its characteristics, we show that background D2D traffic exhibits quite irregular, often varying, patterns and trends. These characteristics have important implications for data center operators or designers, and can help them in designing efficient schemes for deploying/managing data centers, doing content replication, as well as a lot of other background operations.

2.7 Summary

Understanding data center traffic dynamics is critical for designing and managing large scale data centers. Besides a few recent studies [4, 5] of traffic within a single data center, little is known about inter-data center traffic dynamics. In this chapter, using the network traces collected at five major Yahoo! data centers, we provided the first study on traffic dynamics among multiple data centers within a large global content provider. Our results indicated that Yahoo! employs a hierarchical way of deploying its data centers. In "satellite" data centers, D2D traffic is closely correlated with D2C traffic. For the three US locations, we identified two types of D2D traffic: i) D2C triggered traffic and ii) Background D2D traffic. By applying port based correlation, we separated these types of D2D traffic. Our findings showed that background D2D traffic is quite dominant in the aggregate D2D traffic. At the same time, it shows no significant trends, and has smaller variance compared with the D2C triggered traffic. On the other hand D2C triggered traffic shows varying trends over the day which are mainly governed by the user dynamics, and has larger traffic variance. Also, generally these data centers provide multiple services which may be located(and replicated) at different data centers. We also showed that several of Yahoo! services have correlated traffic. These correlations have important implications for distributing different services at multiple data centers. In addition, we also developed simple traffic feature based inference techniques to separate the Yahoo! and non-Yahoo! IP addresses using the anonymized NetFlow traces. The proposed techniques not only perform really well on the Yahoo! NetFlow datasets, it is simple, intuitive, and general, therefore can be applied to anonymized NetFlow traces of other providers as well.

Chapter 3

Front-end Server Roles in Dynamic Content Distribution

3.1 Introduction

More and more content on the Internet is now stored at powerful, large-scale data centers in the cloud. A significant portion of this content is *dynamic* in that in response to a user's request for content, the content returned to her is generated dynamically and sometimes personalized. Web search is one common example of such dynamic content generation. With the emergence of cloud computing and cloud-based services, we expect that more data will be stored in the cloud, and more dynamic content will be generated on the fly in response to user requests. Because of the sheer scale and cost of building and operating large-scale powerful data centers, their number is few and far between. Hence they are generally far away from a large majority of users.

One way to mitigate this effect and improve the user-perceived performance (e.g., the overall response time) is to deploy "proxy" servers – hereafter we refer to them as *front-end* (FE) servers – closer to users. The usefulness of such an approach for *static* content distribution (e.g., video) is obvious because of *content caching*. FE servers can also be exploited to improve the *user-perceived* performance of *dynamic* content distribution due to the following two key aspects [21, 22]: i) a portion of the (dynamic) content may be static; thus can be cached and delivered immediately from the FE servers; and more importantly, ii) via *split* TCP connections, a FE server can establish a *persistent* TCP connection with the data center which not only eliminates the effect of TCP slow-start congestion window ramp-up on the throughput of the TCP connection between the FE server and the back-end (BE) data center, but also mitigates such effect on the throughput of the TCP connection between the user and the FE server (due to the reduced RTT). Nonetheless, the *overall* user-perceived *end-to-end* performance likely depends on a confluence of various factors such as RTT, loss rate and throughput of the connections between users and FE servers and between FE servers and BE data centers, the load on FE servers, the processing time at BE data centers to generate user-requested dynamic content, and so forth.

To investigate the roles of FE servers in improving user-perceived end-to-end performance of *dynamic* content distribution, we conduct an active measurement-based *comparative* study of Google and Microsoft Bing web search services. Both services utilize a number of FE servers that are placed closer to users to assist *dynamic* content (i.e.,

search results) distribution: Google deploys a set of its own FE servers, whereas Bing relies on Akamai's content distribution network (CDN). Using the PlanetLab nodes, we perform extensive measurements of Google and Bing search services by emulating and generating a variety of keyword search queries of varying popularity, granularity and complexity, and collect a large amount of dynamically generated content and application-layer measurement data. Through *content analysis* and *temporal clustering* of packet-level events, we confirm that both Bing and Google search results contain a *static* portion, such as the HTTP header, HTML header, etc., which is cached and delivered immediately by the FE servers upon receiving a user request. As the effect of the aforementioned second key aspect cannot be *directly* measured, we develop a novel *model-based* inference framework: we classify and separate the content (i.e., search results) into two parts – the *static* portion that is cached and directly delivered by FE servers, and the *dynamic* portion that is generated by the BE data centers and then passed onto the FE servers for delivery. We define several *directly measurable* parameters to characterize and *predict* the delivery performances of static and dynamic portions. These predictions are indeed borne out by our measurement data, and thus enable us to deduce that despite that one utilizes a third-party CDN (as FE servers) and the other does not, both Bing and Google employ FE servers in a similar fashion.

Furthermore, our inference framework allows us to *bound* the *FE-BE fetch time* (T_{fetch}) of dynamic content – namely, the overall time it takes for a FE server to forward user query to a BE data center, for the data center to dynamically generate the user-requested content and deliver it to the FE server – we note that *this time cannot be directly observed and measured at the end systems*. Comparing Bing and Google search services, we find that the fetch time between Google FE servers and BE data centers tends to be smaller and more stable for Google; in contrast, the fetch time between the Akamai FE servers and Bing data centers tends to be larger and shows higher variability. Hence, despite Akamai FE servers are generally placed closer to users (and their number is larger than that of Google FE servers), user-perceived performance of the Bing search service tends to vary significantly from queries to queries. While it is known that placing FE servers closer to users can generally improve the user-perceived performance (e.g. [21]), our study demonstrates a critical trade-off between placement of FE servers and the FE-BE fetch time which limits this improvement: there is a

distance threshold within which placing FE servers further closer to users is no longer helpful; instead, the end-to-end performance is now determined solely by the FE-BE fetch time. Thus, to improve the end-to-end performance, it is also crucial to optimize the FE-BE fetch time. Lastly, we develop heuristics to factor the FE-BE fetch time so as to estimate the back-end processing time (T_{proc}) and the BE-FE round-trip delivery time (RTT_{be}) separately.

Related Work. Most prior works in this area focus on understanding the distribution of static content. For instance, in [23], authors study the assignment of clients to the CDN edge servers, in order to maximize the performance for each user. Similarly, several other studies such as [24, 25, 26] develop techniques to use a peer-to-peer based model to distribute the content, which is assumed to be static. In [27] authors study the various caching mechanisms used by CDN networks. Besides, a recent OSN study [28] show that placing more proxy servers can enhance the content distribution for Facebook users sharing similar interests. However, the focus of the study was to exploit the redundancy in the data accessed by users in a given geography, and reduced the user-perceived delay by caching the content at nearby proxy servers. A study [21] that is more closely related to our work compares the performance of cloud service with and without tcp-splitting, and therefore dealt with only the indispensability of TCP-splitting. On the other hand, in this chapter, by reverse engineering the strategies used by Google and Bing to distribute the dynamic content, we shed light on the trade-offs among different underlying factors in designing TCP-splitting for dynamic content distribution.

3.2 Problem Setting & A Simple Model

In this section, we describe the basic infrastructure with FE servers for dynamic content distribution, and present a simple abstract model to capture the interactions between users and FE servers and between FE servers and BE data centers. This model will guide us in the measurement and analysis of dynamically generated search results from Bing and Google.

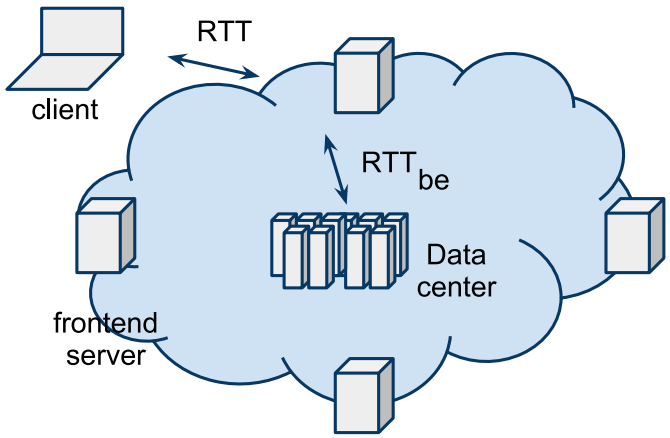

Figure 3.1: Content distribution infrastructure.

Figure 3.1 depicts a typical infrastructure set-up for dynamic content delivery consisting of FE servers that are deployed at the "edge of the cloud" (thus relatively closer to users) and BE data centers "deep in the cloud". When serving static content, FE servers often function as caches. When serving dynamic content, FE servers can play two key roles, as mentioned in the introduction: i) they can cache certain portion of static content that is common to all dynamically generated content, and deliver it immediately upon receiving a user's request; ii) by splitting the end-to-end TCP connection, FE servers can establish persistent TCP connections with BE data center to speed up the delivery of the dynamically generated content between them.

Unlike the static content distribution, there are several key factors affecting the user-perceived performance of dynamic content distribution: the latency or round-trip time (RTT), available bandwidth and loss rate between a user and a FE server, the load on a FE server, the latency or round-trip time, available bandwidth and loss rate between a FE server and the BE data center, the processing time at the data center

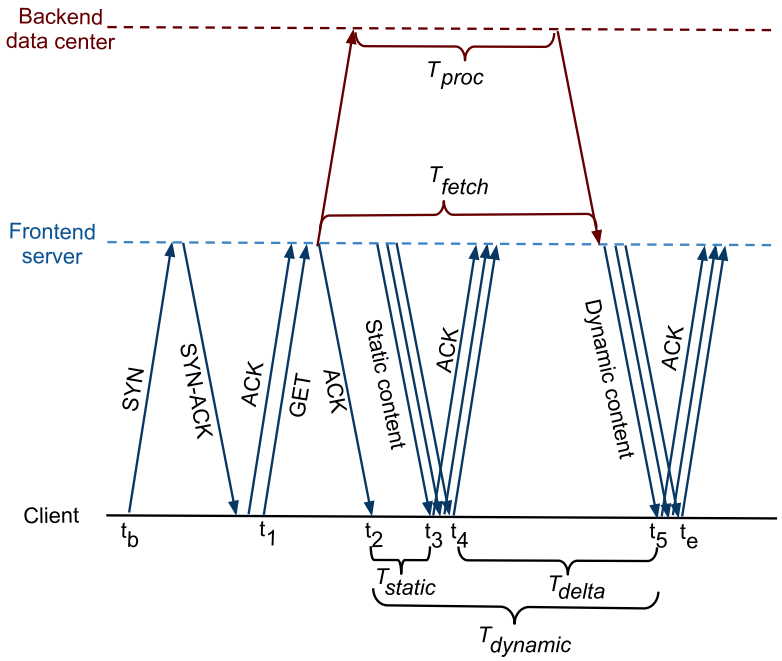

Figure 3.2: Modeling search query timeline.

to generate dynamic content in response to a user's request, the load on servers at the data centers, and so forth. Unfortunately most of these latter factors cannot be *directly observed and measured at the end hosts*. To address this, we develop a novel inference framework which allows us to indirectly measure and quantify the overall (search query) processing and delivery time between the data center and a FE server.

As shown in Figure 3.2, we model the packet-level generation and reception process and define several (*measurable*) parameters to capture the events of the *static* and *dynamic* portions of the content distribution. The process starts at t_b with a TCP three-way handshake[1] . At time t_1, a user (client) sends an HTTP GET request, and receives the ACK packet from the FE server after one RTT at t_2. At t_3 the client receives the first packet containing the *static* portion of the content, and at t_4 receives the last packet containing the static content. At time t_5, the first packet containing dynamic content is received, and at time t_e, the final packet of the entire content is received. The correctness of the model is validated in later sections, and is also quite consistent with the descriptions given in [29, 21, 22].

[1] DNS resolution time is not included, as it is negligible as compared to the overall user-perceived response time.

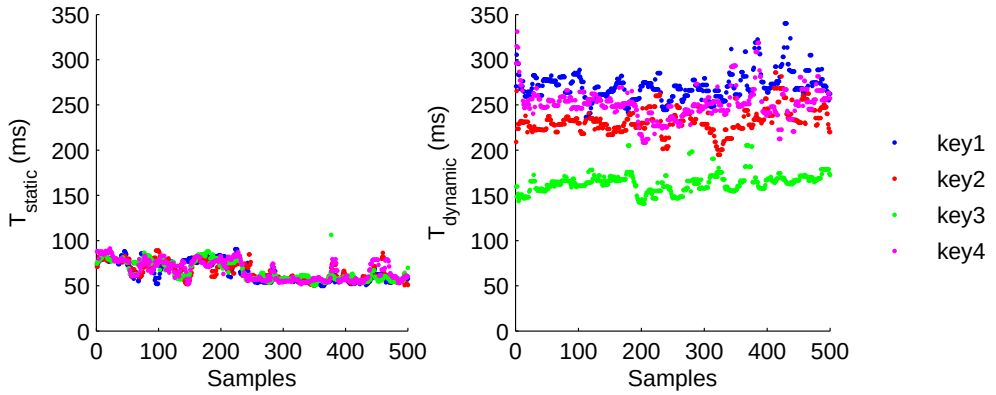

Figure 3.3: T_{static} and $T_{dynamic}$ for different keywords.

In our model, the time when the first and last packets of the *static* content portion are received should, *by definition*, hinge on the factors involving only the FE server and client, namely, they are independent of the BE data center. We define $T_{static} := t_4 - t_2 (= t_4 - t_1 - RTT)$ which bounds the processing and delivery of the static content portion *at the FE server side* (assuming a constant RTT). We define $T_{dynamic} := t_5 - t_2$ and $T_{delta} := t_4 - t_2$. Then $T_{dynamic}$ upper-bounds the overall fetch time T_{fetch}, while $T_{delta} := t_5 - t_4$ serves as a (potentially loose) lower-bound on this overall time. Meanwhile, T_{fetch} is mainly consisted of the time it takes for the FE server to send the user request to a BE data center, for the data center to process and dynamically generate a response (the *dynamic content* portion) to the user request, and deliver it to the FE server. Namely,

$$T_{delta} \leq T_{fetch} \leq T_{dynamic} \tag{3.1}$$

$$T_{fetch} = T_{proc} + C * RTT_{be} \tag{3.2}$$

where C is constant, which depends on the TCP window size on the BE data center. Moreover, fixing a FE server, T_{fetch} should be a constant, assuming the variability introduced by FE server and data center loads, the available bandwidth between them, etc., is small and negligible relative to the RTT between the FE server and a client. In other words, the time it takes for the FE server to receive the delivery of the *dynamically generated* portion of the content from the BE server is (roughly) a constant. On the other hand, the delivery time $(t_4 - t_3)$ for the static content is a function of RTT. Hence our model predicts that as RTT increases, T_{delta} decreases. With sufficiently

large RTT, $T_{delta} = 0$; thus the last packet of the static content portion and the first packet of the dynamic content portion will be delivered back-to-back or even coalesce as a single packet.

3.3 Active Measurement & Content Analysis

For our study, we develop an in-house user search query emulator, which performs exactly the same functionality as the web-based search box. We deploy the emulator on globally distributed[2] PlanetLab nodes as well as on our lab and home machines. The number of Planetlab nodes participating in each of our experiments ranges from 200 to 250, depending on their reachability at the time being. We conduct extensive measurements by submitting the same search queries to both Bing and Google search engines, and collect detailed TCPdump with full application-layer payloads. We perform two sets of experiments: 1) In the first set, search queries are launched from all measurement nodes to their *default*[3] FE servers every 10 seconds. 2) In the second set, we fix one FE server (of Bing or Google respectively) at a time, and launch queries from all measurement nodes to this server. We repeat these two sets of experiments using different sets of keywords and over different times. We refer to the data collected in the first set/type of experiments as *Datasets A*, and the second as *Datasets B*. Due to space limitation, we omit the details of the active measurement platform and experiment design and execution.

Parsing Application-Layer Packet Traces and Identifying the Static Content Part. Using the packet traces collected via TCPdump, we perform detailed application layer content analysis as well as transport layer temporal classification of packet generation and reception events. We find that in the search results returned by both Bing and Google, there is a portion of the content that is *static*, namely, independent of the search keywords submitted. This *static* content portion includes the HTTP header, HTML header, CSS style files, and the static menu bar, e.g. "Videos," "News," "Shopping," etc. that are placed on top of each search result page. The remaining *dynamic* portion includes the *keyword-dependent* dynamic menu bar, search results and ads. The temporal analysis of the packet-level events confirms that this static content portion is most likely cached at FE servers and delivered immediately, as its delivery time is largely a function of RTT, and does not vary significantly with, say, the types and complexity of search queries as does the dynamic content portion (see below and Section 3.4).

Choice and Effect of Search Queries. Since the dynamically generated content

[2] Although users are distributed globally, the size of the returned search results are quite similar.
[3] The default server is whatever server IP address the DNS resolution returns to the client.

portion is search query dependent, we use different sets of search keywords with varying *popularity, granularity, and complexity*. For instance, the Bing main page provides a list of most popular keywords at the current time. In terms of granularity, we generate search queries with concatenated keywords which gives us increasingly refined search results (e.g., "Computer Science Department" and "Computer Science Department at University of Minnesota"). In terms of complexity, we use long and complex search queries and mixtures of keywords that are not highly correlated (e.g., "computer *and* potato").

As an example, Fig. 3.3 illustrates the effect of 4 search keywords of different types on the Bing search performance: the left and right panels plot T_{static} and $T_{dynamic}$, respectively, for the 500 sample queries made in chronological order. As the performance is susceptible to short-term fluctuations, we plot the moving median with the sample window size being 10. (The results using Google have similar distributions.) We observe that $T_{dynamic}$ varies significantly with the types of search keywords used, whereas T_{static} is mostly insensitive to the search keywords used.

Do FE Servers Cache Search Results? To answer this question and mitigate the effect of this type of caching on our measurement analysis, we conduct a series of experiments. First, we collect a list of commonly searched keywords (e.g., "mobile cloud computing") as listed in the drop-down "search suggestion box" used by both Bing and Google. We also generate a list of search words not listed by the suggestion bar. A total of 40,000 keywords are used in the experiments. We perform two sets of experiments. In the first set, all measurement nodes submit the same search query sequentially to a fixed FE server. In the second set, each node submits a different search query to a fixed FE server. We repeat the experiments with different search queries and vary the FE server used. Analysis and comparison of the measurement results (in particular, the characteristics and distributions of $T_{dynamic}$) suggest that FE servers do *not* appear to cache any (dynamically generated) search result. This may not be too surprising, as most search engines attempt to *personalize* search results for individual users.

3.4 Dissecting End-User Performance

In this section, we present the analysis of search query traces collected by us using Bing and Google. In particular, we use the model described in Sec. 3.2 to extract the T_{static} and $T_{dynamic}$ from the traces. We present our methodology to understand how the round trip delay between user and FE web servers affects the distribution of T_{static} and $T_{dynamic}$.

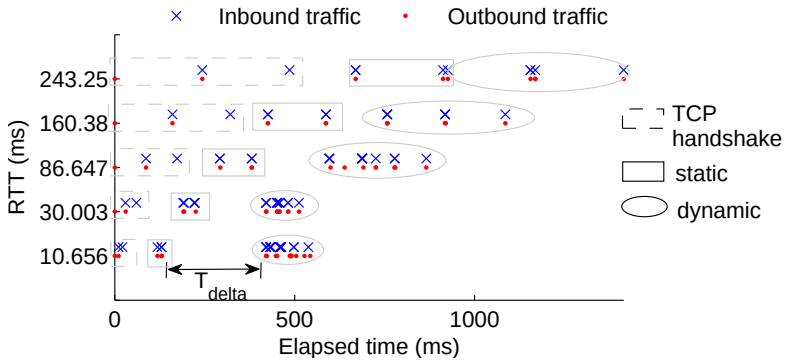

Figure 3.4: Inbound and outbound traffic events triggered by a single search query.

3.4.1 Extracting & Analyzing T_{static} and $T_{dynamic}$

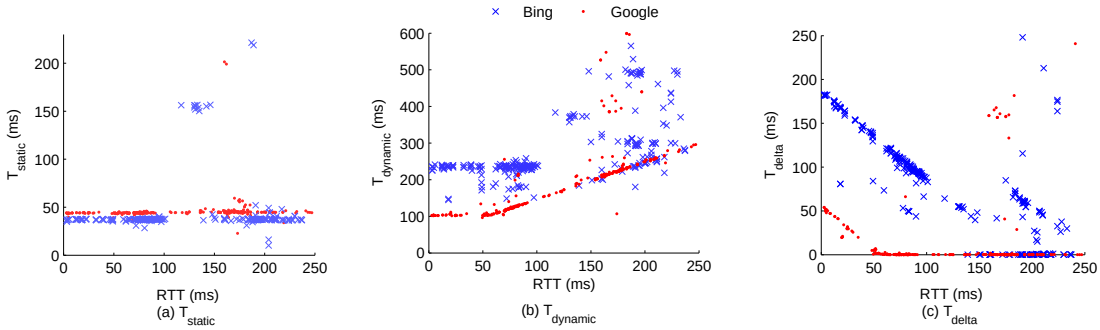

Figure 3.5: Distribution of T_{static}, $T_{dynamic}$, and T_{delta}.

Guided by the basic abstract model in Sec. 3.2, we perform temporal analysis of the packet-level events at the user (client) side using the collected packet traces. Figure 3.4 plots five sample timelines of packet generation and reception events at the client side,

where five PlanetLab nodes are used as clients, each submitting the same search query to the Bing FE server. The x-axis represents the elapsed time since the start of the session, when the first TCP SYN packet is sent to the FE web server. The y-axis represents the round trip time (RTT) between the client and the FE server. Each horizontal array of dots/markers represents the timeline of packet-level events, where each blue cross/red dot marker indicates the sending/receiving time of a TCP packet. When RTT values are small, the temporal clusters of packet events are clearly visible: correlating with the application-layer packet payloads, we find that the first cluster represents the three-way TCP handshake between the client and the FE server; the second and third cluster represent the delivery of static and dynamic contents, respectively, from the server to the client. As the RTT increases, the gap between the end of the second and the beginning of the third clusters decreases, and eventually the two are lumped together, as predicted exactly by our model.

Using the *datasets B* collected via the second type of experiments (see Sec. 3.3) conducted for both Bing and Google, we extract and analyze the parameters T_{static}, $T_{dynamic}$ and T_{delta}. Figure 3.5 shows the distribution of these parameters using a sample of measurement data for one Bing FE server (IP address 198.189.255.208) and one Google FE server (IP address 74.125.224.18), where 720 repeated experiments using the same search query are performed over time at each PlanetLab node. In these plots, x-axis represents the RTT between a PlanetLab node and the FE server, while y-axis represents the *median* value of T_{static}, $T_{dynamic}$ and T_{delta} observed at each of the PlanetLab nodes. With a few outliers, the leftmost plot shows that for both Bing and Google, T_{static} is relatively stable and is largely independent of the PlanetLab node that had sent the query. Similarly, $T_{dynamic}$ is roughly a constant when RTT is small. However, when RTT is large, $T_{dynamic}$ increases linearly with RTT. In the case of T_{delta}, when RTT is small, it decreases linearly with RTT; and it becomes zero when RTT is beyond a certain threshold (for Google, this threshold is around 50ms to 100ms, for Bing, around 100ms to 200ms).

All these observations can be explained using our simple abstract model. First note that in the definition of T_{static}, we have subtracted the (initial) effect of RTT. Hence T_{static} depends mostly on the time to generate and deliver the (same) static content portion at the FE server (assuming the available bandwidth and the server load seen by

all the PlanetLab node is roughly the same). When RTT is small, the delivery of the static content portion will be finished before the FE receives the dynamically generated search result from the BE data center (the time at which this is received at the FE server is independent of where the client is). Hence when RTT is small, $T_{dynamic}$ is roughly a constant while T_{delta} decreases as a function of RTT. When RTT increases beyond a certain threshold, the dynamic content portion will be received by the FE server before the static content portion is entirely delivered to the client. Hence $T_{dynamic}$ increases as a function of RTT (due to the TCP window mechanism), while T_{delta} becomes zero. The observations therefore match the prediction by our simple abstract model. Our analysis also suggests that below a certain threshold, reducing the RTT further will not drastically improve the overall user perceived performance.

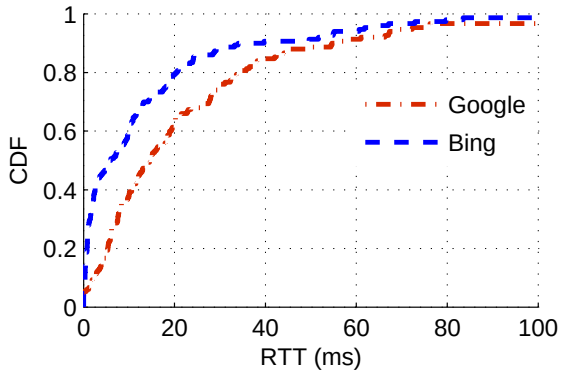

Figure 3.6: RTT distribution.

3.4.2 Comparing Bing & Google Performances

We now compare the performances of Bing and Google by examining various factors and time components in affecting the overall user-perceived response times using the *datasets A* collected via the first type of experiments (see Sec. 3.3).

Comparing RTT Distributions. In Figure 3.6, we compare the RTTs between the PlanetLab nodes and their *default* FE servers (as determined by the DNS resolution). As seen in this plot, in general, Bing has FE servers (Akamai CDN servers) which are closer to PlanetLab nodes than for Google. In particular, more than 80% of PlanetLab nodes observe an RTT of less than 20ms for reaching the Bing FE servers. On the other

hand, only 60% of PlanetLab nodes observe this latency for Google.

Comparing T_{static} and $T_{dynamic}$ Distributions.

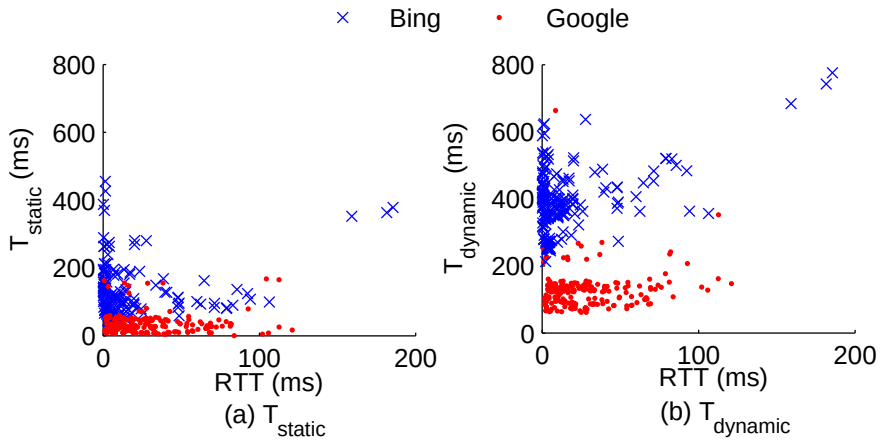

Figure 3.7: T_{static} and $T_{dynamic}$ for Planetlab nodes using default frontend servers.

Next, we extract and analyze T_{static} and $T_{dynamic}$ seen by each client when the default FE server is used for search queries. Figure 3.7 shows the distribution of T_{static} and $T_{dynamic}$ for both Google and Bing. As seen in this figure, although the Bing FE servers are generally closer to the clients (PlanetLab nodes), it has significantly higher value of T_{static} and $T_{dynamic}$ than Google. In contrast, Google has slightly farther FE servers from the clients, but has significantly lower T_{static} and $T_{dynamic}$. In addition, Bing exhibits more *variable* performance (i.e., higher variances) in the measured values of T_{static} and $T_{dynamic}$ than Google. These results illustrate that placing FE servers closer to clients does not necessarily reduce T_{static} and $T_{dynamic}$. We speculate that a plausible reason that Bing has higher and more variable T_{static} values may be due to the higher and more variable loads at the Akamai FE servers, as they are shared with a number of other services; while that Google FE servers have smaller and more stable T_{static} values may be attributed to the fact that these servers are likely dedicated to distribution of search results. Similarly, the $T_{dynamic}$ values for Bing FE servers are larger, and have more variability. The contributing factors may involve the processing capability and load fluctuations on the BE data centers, the search algorithm being used, the quality of the connection between FE and BE servers, e.g. loss rate, jitter, throughput, etc. A dedicated connection between FE and BE servers via "internal" network usually

provides better connection than that built on the general Internet connections.

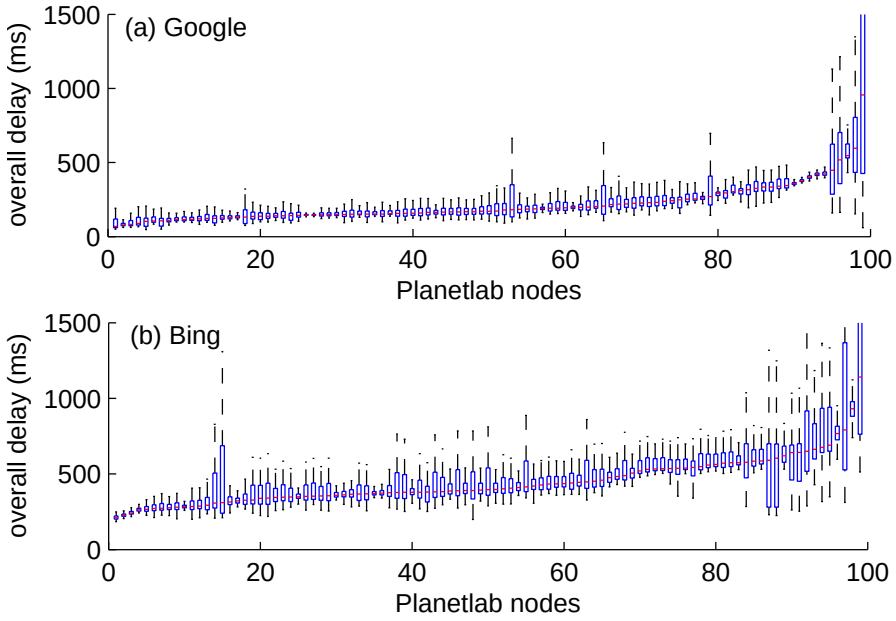

Figure 3.8: Overall delay performances.

Comparing Overall User Search Experiences. Finally, we compare the overall responsive time for individual search queries performed on both Bing and Google, as shown in Figure 3.8. The x-axis represents the PlanetLab nodes, and the y-axis represents the box-plot for the distribution for different samples. The results show that comparing Google, users using the Bing search service tend to experience slightly longer and more variable overall response times. In conclusion, our study shows that simply placing FE servers closer to users may not be entirely effective in improving the overall user-perceived performance in dynamic content distribution. Other key factors such as the processing time and server loads at both FE servers and BE data centers as well as the (physical and TCP) connections between them also play a critical role. Improving and optimizing these factors are therefore important in improving the overall user-perceived performance in dynamic content distribution such as dynamic generation of search results in response to user queries.

3.5 Factoring FE-BE Fetch Time

As discussed in Sec. 3.2, T_{fetch} consists of two key components, namely, T_{proc} and RTT_{be}. They represent the search query processing time at the BE data center and the delivery time of search results from the BE server to the FE server respectively. As part of our ongoing work, we are exploring various mechanisms to separate these two components.

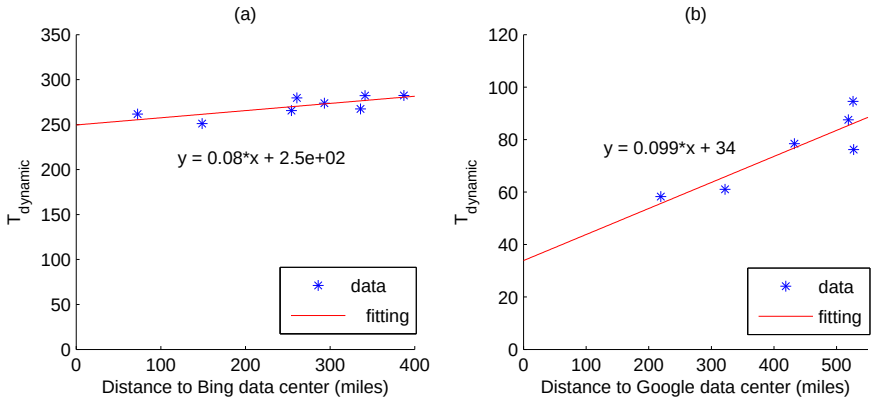

Figure 3.9: Correlating $T_{dynamic}$ and the distance to BE data center.

To get a better understanding of these components we conducted the following analysis. We first get a list of possible locations for Bing and Google data centers from [30, 31]. Next for each of these data centers we consider the geographically closest FE servers, and plot the distribution of $T_{dynamic}$ with respect to geographical distane between FE and BE. As explained in Sec 3.4 for smaller values of RTT, $T_{dynamic}$ can be considered as an approximation for the T_{fetch}. Figure 3.9 shows the distribution of T_{fetch} time for Bing and Google with respect to the geographical distance between the FE and BE locations. For this plot, we consider the Bing data center located in Virginia (US), while we pick the Lenoir, North Carolina data center for Google. As seen in these figures, the T_{fetch} time increases linearly as the distance between BE and FE increases. We perform a linear regression to fit a straight line for these data points, which is shown using the red continuous lines in the figure. As seen in this figure the Y-intercept for the regression is 260ms for Bing, while it is only 34ms for Google data center. This intercept actually represents the computation time for a given search query for Bing and Google data centers. Similarly, the slope of the line represents the contribution of network delay

in T_{fetch}, which are similar for both Bing and Google. For the different keywords used in our search queries, we get very similar slope, but pretty different intercept values. Though our initial results show interesting characteristics of factors affecting the FE-BE fetch time, we are currently conducting extensive experiments and analysis to gain a better understanding behind the factors affecting the FE-BE fetch time, and thus will potentially guide us in designing better content placement and delivery strategies for dynamic content distribution.

3.6 Discussions

In this chapter we have focused on the roles of frontend servers on the end-to-end performance of search queries using the standard search functions of search engines. More recently, some search engines such as Google has introduced more advanced search features such as the *interactive* "search as you type" feature. Our preliminary investigation of this new feature shows that our basic model and key observations still hold. We find that using the interactive search feature, after each letter a user has typed, a separate query (using a *new* TCP connection) is sent to the FE server. The delivery of each query hence still fits our basic model; although we believe that it is likely that the search query processing times at the BE data centers are generally reduced because the subsequent queries are highly correlated with previous queries. We are in the process of conducting more thorough measurements and analysis on this and other search features.

As most the Planetlab nodes are located within or close to the University campus networks (and it is known that some Akamai frontend servers are placed closer to University campus networks), we realize that using the PlanetLab as the testbed may introduce some biases. For instance, the RTT between PlanetLab and Akamai FE servers may not be of all users. In addition, in our measurements we do not see any significant packet losses. In an environment where the loss rates are high (e.g., in a wireless network), placing FEs closer to users in fact may significantly improve the user-perceived end-to-end performance by reducing the total time needed to deliver the query result (that has been delivered to the FE server from a BE data center) to a user. As part of ongoing work, in addition to the PlanetLab, we are utilizing other testbeds (e.g., Seattle testbed [32]). We are also investigating the trade-offs between RTTs and loss rates (e.g., in a WiFi environment) in the placement of FE servers.

3.7 Summary

In this chapter we investigated the roles of FE servers in improving the user-perceived performance of dynamic content distribution. Using Bing and Google search services as case studies, we conducted extensive application-layer active measurement and data analysis. Our results demonstrate that there is a critical trade-off between the placement of FE servers and the FE-BE fetch time. While placing FE servers closer to users can help reduce latency, other key factors such as processing times and loads at both FE servers and BE data centers as well as the quality of (physical and TCP) connections between them also play a critical role in determining the overall user-perceived performance.

Chapter 4

Understanding and Diagnosing Search Response Time

4.1 Introduction

Web services are the dominant enabler for a wide range of online user activities such as searching and accessing content, shopping, and social interactions. Their performance is critical; even small increases in response time can hurt the experience for users, and affect the monetization ability for service providers [33, 34]. As we will illustrate in § 4.2, performance of modern cloud-based Web services hinges on a variety of *diverse, interacting factors* that span various servers in back-end data centers, CDN edge servers, networks, client machines, web browsers, and user behavior. It is therefore a challenging, albeit extremely important, task for service providers to understand the various key factors that contribute to performance variations so as to quickly detect and diagnose any service degradation (e.g., due to failures) and ensure good user experience.

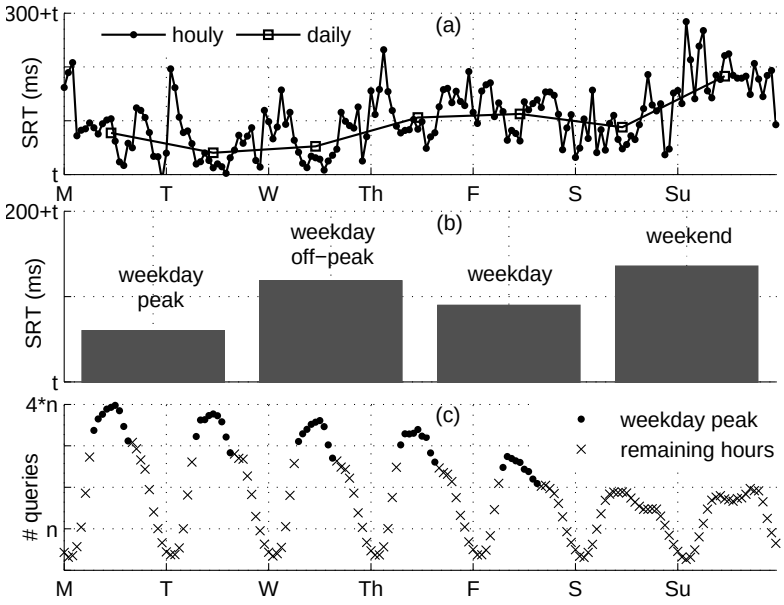

Figure 4.1: Search response time and query load at a large search provider.

Using a large search service as a case study, in this chapter we highlight the challenges in dissecting the various factors affecting the end-to-end *search response time* (SRT) – defined as the time between sending the query to receiving the complete response page at the browser – experienced by users, and in diagnosing performance anomalies. Using real measurement data, we find that the average SRT not only varies widely with hour of the day and day of the week, but exhibits *surprising* behavior. As an example,

Figure 4.1(a) shows the average SRT for the provider we study over the course of a week (the average SRT is computed for queries issued by client machines located in US – the primary geographical area of interest in our study – taking into account the time zones). Each data point is an average over all queries received in an hour. We hide the absolute SRT values to ensure confidentiality. We see that the SRT varies significantly over time: *counter-intuitively,* the average SRT is actually higher during off-peak hours and weekends (Figure 4.1(b), when the query load is in fact much lower (see Figure 4.1(c)). We have confirmed that this behavior is not unique to our provider but exists for another large search provider as well.

To resolve this paradox and understand the SRT variations in general, we conduct a systematic study of massive amount of measurement data collected over 6 months using detailed client- and server-side instrumentation. Using time series decomposition [35] and other analytics, we develop a general analysis framework to identify and separate the observed SRT variations into two components: *systemic* variations that are caused by periodic and other systemic changes in the key underlying factors that vary over time, and *anomalous* variations that are more likely due to unexpected system issues, e.g., server failures, network congestion, or even malicious attacks against the service.

Through careful and detailed analysis, we demonstrate that the systemic variations in SRT can be attributed to three key (*interacting*) factors: the network characteristics of clients, the nature of queries, and the web browser speed (which also includes the possible effects of client machines). The complex interactions of these three key factors over different times of the day and the days of the week explain the apparent *paradoxical* SRT behavior we see in Figure 4.1: Although the number of queries issued by users decreases significantly during off-peak hours and weekends, most of these queries come from slower (residential) networks (as opposed to faster enterprise networks during the peak hours). Furthermore, a greater fraction of the queries during off-peak hours result in media-rich response pages that take longer to download and load by browsers. In contrast to mostly work-related queries (e.g., queries for "correlation coefficient") which result in response pages containing most text-based content (e.g., a list of URLs), during off-peak hours users are more likely to search, for instance, celebrities (e.g., "Lady Gaga") or travel destinations (e.g., "Bali") during off-peak hours. Response pages to such queries often contain embedded images or even short clips of videos. While many

users in fact use "faster" browsers during the off-peak hours, this factor is not sufficient to offset the slower networks and queries that lead to richer response pages. The net effect is higher average SRT during the off-peak hours and during the weekends. We also find that the query processing time at the server is relatively stable and has only a small impact on SRT variations.

Without properly accounting for the systemic variations in SRT, detecting and diagnosing performance anomalies can be difficult, as they may get masked by the systemic variations, resulting in both false positives and negatives. Using the manual inspection and existing tools, the operators inform us that it sometimes takes them multiple days to notice and detect anomalies, often only after users complain. Building on the analysis framework, we have developed a performance diagnosis tool that detects performance anomalies by factoring out the systemic variations. Results from a five-month deployment shows that our tool detects over 90% of the anomalies that operators detected using a mixture of user complaints, visual analysis and other means, three times more than the existing detection tools. Further, our tool not only detects many anomalies that the current practice does not, it also correctly pinpoints where the root cause is most likely to lie.

In summary, the main contribution of our work is three-fold:

• We provide a detailed look at SRT variations seen by a Web search provider, and uncover some surprising, apparently paradoxical behaviors.

• We develop a general analysis framework that tease apart the *systemic* and *anomalous* variations in SRT, and identify the key factors and their complex interactions that contribute to the systemic variations.

• We develop a performance diagnosis tool that can effectively detect performance anomalies and help pinpoint the plausible locations of the root causes. It has been deployed for more than five months, with promising results.

To our best knowledge, our study is the first systematic performance analysis of a complex, large-scale *operational* Web service. While it focuses on search service, we believe our analysis framework, tools and insights are also applicable to other cloud-based Web services with similar components and system/user interactions.

4.2 Background: Search Services

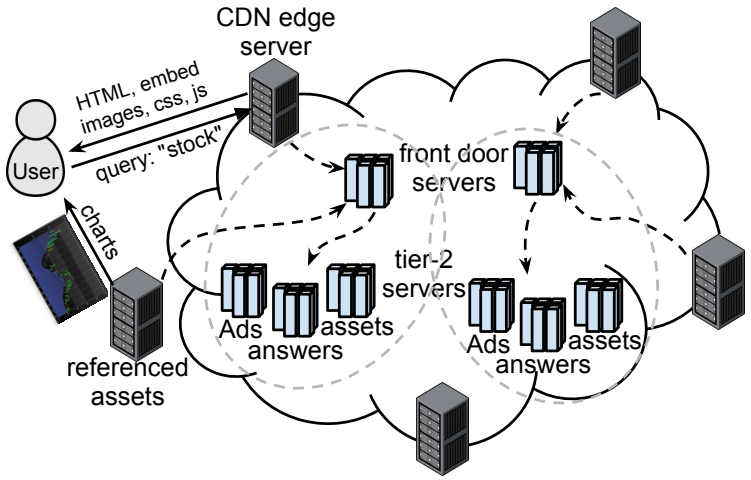

Figure 4.2: Infrastructure of a large Web service.

Figure 4.2 shows a typical infrastructure for large-scale Web services such as search. It consists of three main entities: users that send queries, data centers deep in the cloud that store indexes and compute query responses, and the content distribution network (CDN) servers that sit close to the network edge and serve as intermediaries between users and the data center.

The data center has a number of servers that play different roles in serving the requests and are organized in tiers. The tier-1 servers, also known as the front door servers, parse the request, invoke tier-2 servers, and rank and aggregate the answers from tier-2 servers into the response page that is sent to the user. Different tier-2 servers index different types of content such as Web pages, images, news, videos, and advertisements, and their answers are specific to that content. The front door decides during aggregation which types of content answers are most relevant for a given query, based on hints provided by the tier-2 server, and includes only those answers in the response page.

Because of their high cost, large data centers with all types of servers are located in only a few locations. For performance, a large number of CDN edge servers are deployed. These servers help improve performance by terminating TCP connections closer to the user (leading to faster growth of the congestion window). They establish one or more long-lived, high-throughput TCP connections to front door servers and

multiplex users on these connections. Due to the large diversity of queries across users and the personalization of responses, the CDN edge servers do not cache query results but fetch them from a data center. However, the results of popular queries may be cached in the data center.

User queries start with a DNS lookup, which returns IP addresses of one or more nearby CDN edge servers. The user then opens a TCP connection to an edge server and sends her query. Edge servers relay queries to a close data center and relay responses to the users.

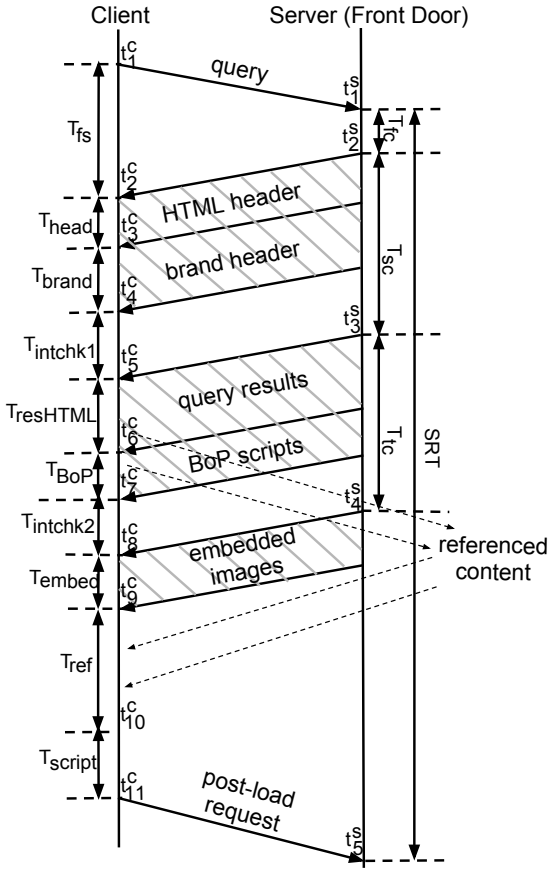

Figure 4.3: Timeline of a search query.

Figure 4.3 shows the interaction between the front door server and the user, after abstracting out the CDN edge server. As soon as the query is received, the front door server relays it to various tier-2 servers. In parallel, it starts preparing the response which goes out in three chunks [36], shown as shaded areas in the figure. The first

chunk contains the part of the result page that is the same across all types of queries, including the HTML header elements and the header portion containing the brand information (e.g., Google or Bing logo image). Personalized user information is also sent in this chunk.

The second chunk begins after the front door server has finished ranking and aggregating the results from the tier-2 servers. It contains the HTML portion of the response followed by BoP (bottom of page) javascript that is executed right after the whole HTML page is loaded, to refresh cookies, set image hovering properties, etc. The content of the chunk is typically compressed and must be decompressed by the browser before it can be parsed.

The response commonly contains pointers to additional *assets* that are needed to render the page. These include images, CSS, and more javascript. Some of these images are sent as objects *embedded* in the response itself. The front door server decides which images should be embedded and transmits them as the third chunk after fetching those images from their locations. The remaining assets, which are *referenced*, are fetched by the user by issuing additional queries to servers that belong to the search provider or third-parties. These queries are pipelined and issued in parallel to the reception and parsing of the HTML content.

After all the content is received and objects in the page are fully loaded (t^c_{10} in Figure 4.3), the browser fires the "onload" event and starts executing the javascript that corresponds to this event. After the script finishes executing (t^c_{11}), the browser finishes rendering the page and issues a post-load request for a tiny (1x1 pixel) image to the front door server using the same TCP connection. As described below, the time at which this request is received by the front door server is used to estimate the SRT for the query.

4.3 Data collection

To gain insight into SRT behavior and factors that impact it, we collect detailed data from one of the largest search providers in the world. The granularity of the data is at the level of individual queries. For each query, a set of server-side and client-side metrics are collected. The server-side metrics include all timestamps shown on the right in Figure 4.3 (t^s_*). The client-side metrics include the type of browser (user-agent) and the timestamps on the left (t^c_*), which are collected using javascript and HTML instrumentation. The timestamps are taken by the browser and returned to the server as part of the post-load request (at t^c_{11}). Through controlled experiments, we have verified that the overhead of collecting this data, including its impact on SRT, is negligible.

We use the collected timestamps to compute the time delay measures (T_*) that are shown in the figure. Each measure captures the time taken for a certain activity to complete on the client or the server side. For instance, T_{fc} is the time it takes for the front door server to transmit the first byte of the first chunk after receiving the query and $T_{intchk1}$ is the time the client has to wait between the last byte of the first chunk and the first byte of the second chunk.

Some of the measures that we need for the analysis cannot be directly computed from the timestamps and we thus approximate them. The first of these is SRT itself. While considering SRT, we ignore the time for performing the DNS lookup and establishing the TCP connection with the edge server. Both these activities are relatively quick as they involve one round trip to a nearby server, and DNS names would be cached in many cases; we are able to confirm this for a subset of our clients using a separate data source. Ignoring these two factors, the SRT for a query is $t^c_{11} - t^c_1$. But our data does not have t^c_1 because HTML instrumentation does not allow us to log the time when the query is sent. We thus estimate SRT as $t^s_5 - t^s_1$, which closely approximates $t^c_{11} - t^c_1$ when the one-way delay of the original query is similar to that of the post-load query [37, 38].

Another measure that we approximate is network round trip delay, T_{net} (not shown in Figure 4.3). We approximate it as $T_{net} = (t^s_5 - t^s_2) - (t^c_{11} - t^c_2)$. The first parenthetical quantity measures the time between the server starting to send the first chunk to receiving the post-load query. The second parenthetical quantity subtracts from it the time

between the client starting to receive the first chunk and sending the post-load query. Their difference approximates the round trip network delay. This difference includes the time taken by CDN edge servers to relay queries and responses between clients and front door servers. Data from the CDN servers confirm that this time is negligible compared to network delays (as relaying involves minimal processing and uses passthrough techniques). Thus, we ignore it and attribute T_{net} entirely to the network.

A final measure that we approximate is T_{fs}, the time it takes for the first byte of the first chunk to arrive at the client. We approximate it as $T_{fs} = T_{fc} + T_{net}$.

In addition to the query-level metrics, we obtain from our CDN information about the clients. This includes the client's location and the ASN (Autonomous System Number) of its ISP. The CDN also provides us a coarse estimate of the client access link bandwidth, categorized into six buckets: [1, 56], [57, 256], [257, 999], [1000, 1999], [2000, 5000], and [5000, ∞] Kbps. The CDN measures access bandwidth of a client, which is not susceptible to fast change, once in two months.

The data used in this chapter was continuously collected for a span of over 6 months. We only consider Web search, which is the most common type of search. We exclude queries from outside the USA or from mobile devices because these clients experience disparate conditions and the responses to their queries are also different. In compliance with the privacy and confidentiality conditions for using the data, all results in this chapter are anonymized and normalized.

4.4 Systemic SRT Variations

To identify various key factors that contribute to the observed SRT variations and dissect their complex interactions, we develop a general analysis framework based on time series decomposition and other analytics. This framework enables us to separate the observed SRT variations into two components: *systemic* variations and *anomalous* variations. Int this section we present the general analysis framework, with the focus on systemic variations. Detecting and diagnosing the anomalous variants will be discussed in §4.5.

4.4.1 General Analysis Framework

Table 4.1: Factors that impact each measure.

Measure	Impact factors
T_{fs}	network, server
T_{head}	browser, network
T_{brand}	browser, network
$T_{intchk1}$	query, server
$T_{resHTML}$	browser, query, network
T_{BoP}	browser, query
$T_{intchk2}$	query, server
T_{embed}	query, network
T_{ref}	browser, query, network
T_{script}	browser, query
T_{fc}	server
T_{sc}	query, server
T_{tc}	query, server
T_{net}	network

The observed SRT variations are likely due to a confluence of complex, interacting factors, e.g., network bandwidth, end-to-end latency, server-side processing time, browser speed. To isolate and identify the key contributing factors, we start by decomposing the overall SRT into individual, *measurable/computable* components as shown in Table 4.1 (see Figure 4.3 for what each measure captures). We note the relationship between the individual measures and underlying factors of interest (e.g., network) is in general not one-to-one (see the right column in Table 4.1). Most measures are

impacted by multiple factors, and each factor impacts multiple measures. Each factor may also influence each of the measures in different degrees (i.e., their effects are unbalanced). These complex dependencies make it hard to tease out which factors that cause most of the variations in SRT; we cannot simply correlate SRT to measures that cleanly capture individual factors. Through the decomposition, we nonetheless reduce the number of potential key factors that contribute to each individual measure, and limit their potential interactions.

Methodology

Mathematically, let Y be a random variable (the so-called *response* variable) representing the observed SRT, and let X_i, $i = 1, 2, \ldots, n$, be a set of *explanatory* random variables X_i, each of which represents one of the constituent measures in Table 4.1. We build a linear model $Y = a_0 + \sum_k a_k X_k$ *plus* random noise in the measurements. Note that due to X_i's are not independent, in general $a_k \neq 1, 1 \leq k \leq n$; the model learns the a_k's that best match the model using the measurement data. Given this model, our analysis framework proceeds in three major steps. First, we separate the variance due to random noises in the measurement from the the variances captured by the model (the *systemic* variances). Given the model/systemic variances thus extracted, we then quantify the contribution of each individual measure X_i, dissect and identify the key contributing measures that account for the most systemic variances. In these first two steps, we apply ideas and techniques from *analysis of variance*) (ANOVA) [39]. In the following, we will briefly described these two steps. The results of these two steps are presented in § 4.4.1: most of the model variances can be attributed to three primary factors, *network characteristics*, *nature of queries* and *browser types*. In the *last* step of our analysis, we investigate the contribution of each of these three key contributing factors by controlling the other two factors, and identify the *systemic* variations in SRT due to the interactions of these three key factors. These investigations are detailed in § 4.4.2 to 4.4.4.

The first step of our analysis proceeds as follows. We first compute the variance in the response variable, $SS_T = \sum_i (y_i - \bar{y})^2$, where y_i is an individual (independent) observation of the response variable and $\bar{y}$ is the mean value across all observations. We then partition, SS_T into SS_R and SS_E ($SS_T = SS_R + SS_E$), where SS_R the variance

explained by the model, and SS_E the variance that is not explained by the model. Let $\hat{y_i} = a_0 + \sum_k a_k x_{ki}$, where x_{ki} is an individual observation of the explanatory variable X_k. Then $SS_R = \sum_i (\hat{y_i} - \bar{y})^2$, and $SS_E = \sum_i (y_i - \hat{y_i})^2$.

Given the model variances SS_R computed above, in the second step of our analysis, we further partition SS_R into n components that are attributable to each of the explanatory variables, identify and quantify those measurements that can explain most of the systemic variances in SRT. For this purpose, we focus in particular on two important metrics[1] : i) the *1st order* variance, $SS_R(X_k)$, *as if* the model consists of only one explanatory variable, $Y = a_0 + a_k X_k$; and ii) the *n-th order variance*, $SS_R(X_k|X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n)$, namely, the variance that is captured by X_k but cannot be explained by the interactions of the other $n-1$.

We apply the method above on 1-hour averages of SRT and the individual delay measures listed in Table 4.1. For computation efficiency and to further minimize redundancy, we consider only those individual measures that have a noticeable impact on SRT by excluding from our analysis those measures that either 1) constitute a minuscule fraction of SRT, or 2) are highly correlated with another measure. The first criterion excludes $T_{intchk2}$ and T_{embed}; they represent less than 1% of the SRT. The second excludes T_{fs}, $T_{intchk1}$, and T_{brand}; they are highly correlated with, respectively, T_{net}, T_{sc}, and $T_{resHTML}$ (the Pearson's correlation coefficients are 0.99, 0.79, 0.99). High correlation between T_{fs} and T_{net} indicates that T_{fs} is largely determined by network latency, as the server-side processing time to generate the first chunk (T_{fc}) has only a minute effect.

Results: Key contributing factors

Figure 4.4 shows the amount of variation in SRT that is explained by each of the 9 remaining measures. Focusing on the 1st order variance first, we see that the two measures that explain the most variations (roughly 60% each) are T_{net} and T_{BoP}. T_{net} is impacted by the network latency between the client and the (tier-1) server. T_{BoP} is impacted mainly by the browser speed and query type (e.g., results pages with more images have more complicated bottom-of-page scripts). The next three measures explain

[1] Mathematically, $SS_R(X_k) = \sum_i (x_{ki} - \bar{y})^2$, and $SS_R(X_k|X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n) = SS_R - SS_R(Y_{\bar{k}})$, where $Y_{\bar{k}} = a_0 + \sum_{j \neq k} a_j X_j$ represents the linear model consisting of only the $n-1$ explanatory variables (with X_k excluded), and $SS_R(Y_{\bar{k}})$ is the model variances of this linear model, i.e., $SS_R(Y_{\bar{k}}) = \sum_i (\sum_{j \neq k} x_{ji} - (\bar{y}))^2$.

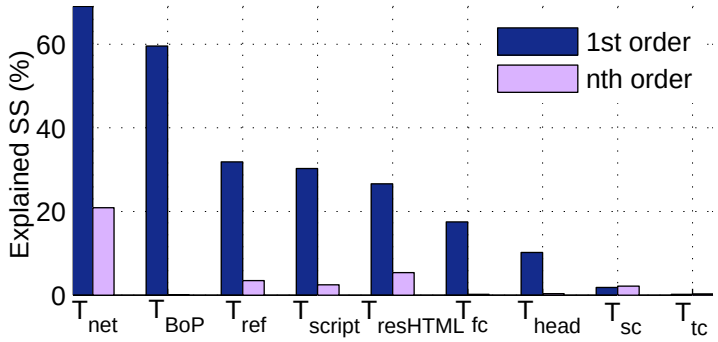

Figure 4.4: Analysis of variance results.

roughly equal amounts of variations (roughly 30% each). Of these, $T_{resHTML}$ and T_{ref} are impacted by network latency and bandwidth since they involve downloading, respectively, of the results page and referenced content, by the query type since size of the results page and referenced content depend on query, and by the browser's speed of rendering the results and referenced content. T_{script} depends mainly on the browser's speed of javascript execution.

The measures that are impacted by the server processing time, T_{fc}, T_{sc}, and T_{tc}, explain relatively small amount of variations in SRT. (Some of these measures are impacted by the query type as well.) Thus, we see that the top several measures that best explain SRT variations are impacted by network, query, and browser, but not by the server-side processing time.

Interpreting these results requires some care. Our results do not imply that server-side processing time is not an important factor in determining overall SRT. They only imply that server time is not a primary culprit that leads to variations in SRT. Server time could be a big fraction of SRT and yet not be responsible for its variations if it is relatively stable.

Next, looking at the n-th order variance, we see that most variables explain only a small amount of variance that cannot be explained by other variables. This underscores the high degree of interaction among various variables. T_{net} is the only variable that has a noticeable amount of n-th order variance.

Though not shown in the figure, we also find that collectively these measures capture almost all of the variance in SRT. The amount of SRT variations that cannot be explained by them (SS_E) is only 0.66%.

In summary, we find that the variations in SRT stem largely from network characteristics, query type, and browser speed; server-side processing time has a relatively small impact. That these three factors lead to systemic SRT variations implies that they must be systemically varying across times of day and days of week. The following sections investigate how and why these three factors vary.

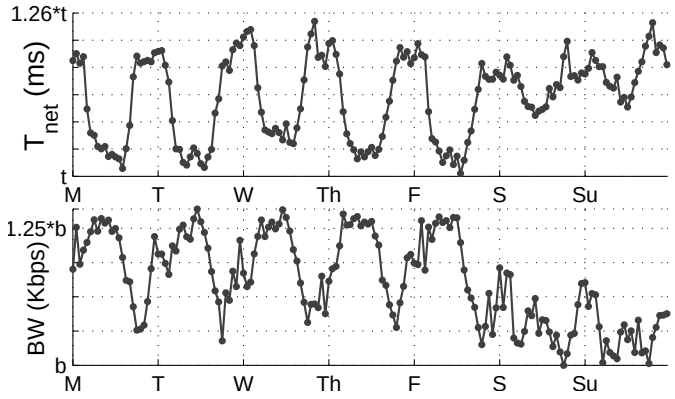

Figure 4.5: variations in network characteristics.

4.4.2 Variations in Network Characteristics

We begin by studying variations in network characteristics. We show that this variation stems from changes in the relative proportion of queries coming from residential and enterprise networks.

Figure 4.5 shows the network characteristics observed across all of our clients. For a week-long period, it plots the hourly average of network latency (T_{net}) and the bandwidth (BW) reported by our CDN. All the graphs in this chapter that show a week's worth of data correspond to the same week; unless there is a weekday holiday, other weeks have very similar behaviors.

We see the network characteristics do vary systemically over time. During off-peak hours, network latency increases by as much as 20% and the bandwidth decreases by a similar percentage. These systemic changes explain why network characteristics is an important factor underlying SRT variations.

Now the question is why the network characteristics vary as shown. If anything, we would have expected the opposite: due to possible congestion during peak hours, network latency should be higher during peak hours, but it is in fact lower. As we

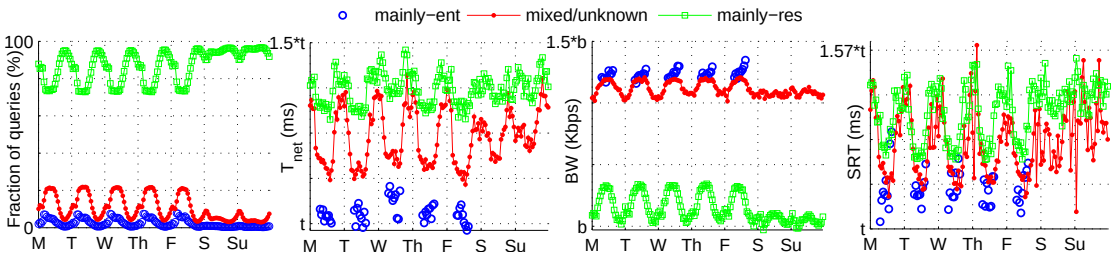

Figure 4.6: Comparing mainly-enterprise, mixed-or-unknown, and mainly-residential ASNs.

explain below, the variations we see are due to the variations in the fraction of queries that come from residential networks which tend to have longer latencies and lower bandwidths compared to enterprise networks.

We use a simple heuristic to classify a network as residential or enterprise, based on the expectation that enterprise networks send relatively few queries during weekends. For each of the 13,349 ASNs observed in our data, we compute the ratio of the number of queries they send during the weekend (Saturday and Sunday) to the number during weekdays. Across all ASNs, this ratio varies between 0 and 10. For a conservative classification, we deem one-third of ASN with the lowest ratios as *mainly-enterprise* and one-third of the ASNs with the highest ratios as *mainly-residential*. The middle third is deemed as *mixed-or-unknown*. The weekend to weekday queries ratio of all mainly-enterprise ASNs is lower than 0.0074. We do not expect the ratio to be perfectly zero for many enterprises if some employees work on the weekends or connect their laptops through the corporate VPNs (Virtual Private Networks). We manually verified the classification of many ASNs that are deemed mainly-residential or mainly-enterprise, based on their names. For instance, Microsoft (ASN 3598) is classified as mainly-enterprise.[2]

With this classification in place, we can now explain the variations in network characteristics observed in Figure 4.5. As shown in Figure 4.6, the mainly-residential ASNs send a greater proportion of queries during off-peak hours (left graph) and they have poorer network characteristics (middle two graphs). Their share is 70% during peak hours but almost 100% during the off-peak hours, and their network latency is 25%

[2] We tried initially to use ASN names as a primary means for classifying ASNs, but dropped that effort because of the large number of ASNs and the fact that many ASN names are hard to interpret.

higher than that of mainly-enterprise ASNs.

The poorer network characteristics of mainly-residential ASNs translate, expectedly, to higher SRTs (right graph). On average, the SRT for these ASNs is 11.2% higher than that of mainly-enterprise ASNs. Due to the low traffic volume from mainly-enterprise ASNs during off-peak hours, we only plot their performance during weekday peak hours. To illustrate this behavior without conflating with the impact of browser speed and query type on SRT, the graph is plotted using data that corresponds to only one type of browser and queries that generate similar pages (i.e., pages that only contain ten plain text results, with no images). Other browsers and query types show a similar effect.

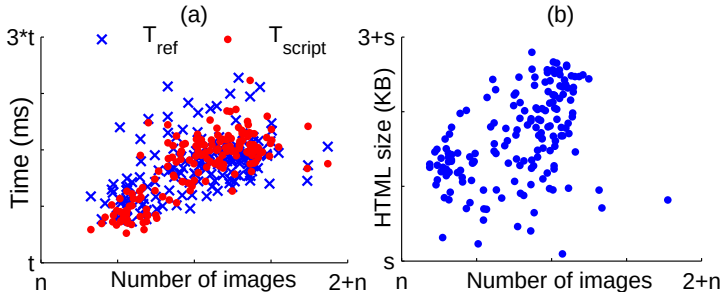

Figure 4.7: Image count vs. other measures of response page complexity.

4.4.3 Variations in Query Type

We now investigate the variations in query type. The nature of the query can impact SRT in two ways. First, it can impact the time it takes the server to compute the results. Second, because different queries have different response pages, it can impact the time it takes to download all content (which includes HTML and multimedia content) and for the browser to load the page (which includes javascript execution and rendering). We showed earlier that server processing time is not a primary factor in SRT variations, and detailed measurements based on different types of queries confirms that the variation in server processing time is small.

Thus, variations in SRT due to query type must largely stem from the diversity and variations in type of responses generated. While most queries generate ten Web answers (HTML links), depending on the query, the response page can also contain other types of answers such as news, images, videos, and maps. These differences lead to high degree

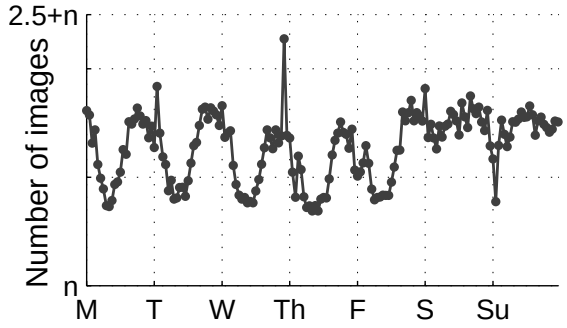

Figure 4.8: Image count variation.

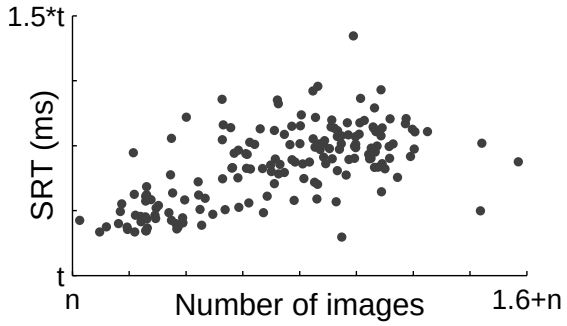

Figure 4.9: SRT vs. image count.

of diversity in response pages. The number of HTML bytes in the response page varies by an order of magnitude across individual queries, and the number of embedded or referenced images vary by more than a factor of two. Consequently, different response pages take different times to download, parse, and render.

Though researchers have recently proposed general measures of Web page complexity [40], we use a simple metric that is specific to the domain of search. Because most non-Web answers contain images, we use the number of images on the response page as a metric for page complexity. It turns out that, as shown in Figure 4.7, this metric is correlated with other possible measures of page complexity such as time to download all referenced content (T_{ref} in left graph), time to load the page after all content has been downloaded (T_{script} in left graph), and HTML size (right graph).

Figure 4.8 shows that the number of images in query responses varies systematically across times of day and days of week. Further, Figure 4.9 shows that the SRT is higher for queries with more images in their responses. The spread in SRT values is about

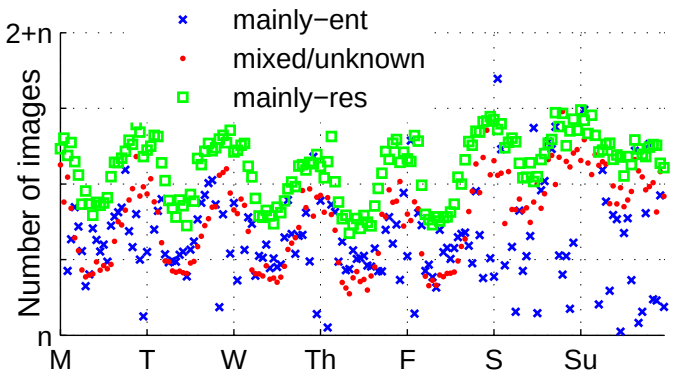

Figure 4.10: Image count in diff. nets.

30%. To factor out the impact of network and browser type, this figure is plotted for one type of browser and mainly-residential ASNs. We obtain qualitatively similar results for other browser and ASN types.

Taken together, these two effects—variations in query richness and dependence of SRT on query richness—explain how the nature of the queries is an important factor in SRT variations.

One interesting question is *why* query richness varies with time. An in-depth look at the data reveals that richer queries (which generate more images) tend to be news-oriented or entertainment-oriented such as queries regarding celebrities and holiday events. Users tend to issue more of such queries during off-peak hours or when they are at home. As shown in Figure 4.10, the number of images contained in the queries sent from mainly-residential ASNs are higher than those from mainly-enterprise ASNs. Therefore, the shift of the "user intent" from peak to off-peak hours is what leads to variations in query richness.

4.4.4 Variations in Browsers

We now show that browsers are a factor behind SRT variations because *i*) the relative mix of browsers accessing the search service varies over time; and *ii*) different browsers have different speeds.

There are eight types of browsers in our data that each issues at least 1% of the queries. Different major versions of the same browser (e.g., Internet Explorer 8 and 9)

are considered as different types of browsers because they can have significantly different rendering and script execution engines.

Figure 4.11 shows the variations of the fraction of queries for the two most popular browsers, Browser-X and Browser-Y. These two browsers account for 35% and 40% of the total queries, respectively. We see that the fraction of queries from these browsers varies substantially with time. Their relative popularity swings by over 25%: Browser-X goes from generating 15% more queries during peak hours to generating 10% fewer queries. The relative popularity of other six major browser types varies as well. Two of them vary like Browser-X, i.e., they send a higher fraction of their queries during peak hours. The remaining four vary like Browser-Y.

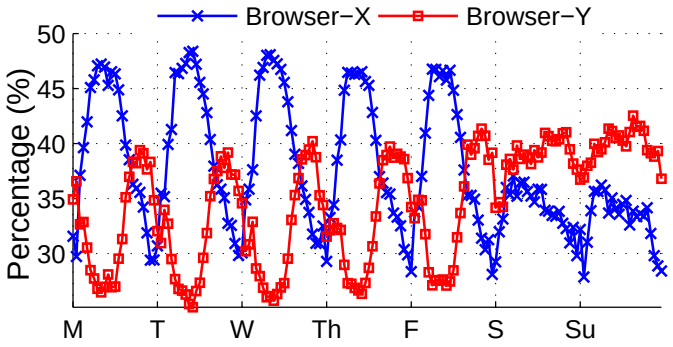

Figure 4.11: %-age of queries from the top two browsers.

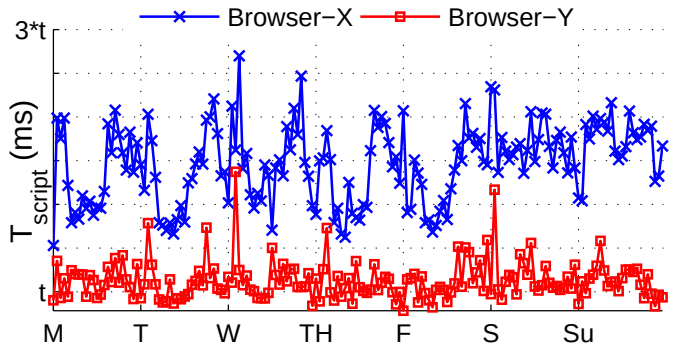

Figure 4.12: T_{script} for the two major browsers.

In addition to generating different fractions of queries over time, different browsers have disparate performance. As an indicator of browser performance, we use T_{script}, which is not impacted by the time it takes to download the page because it captures the time from downloading all content to fully loading the page. Figure 4.12 shows

the performance of the two popular browsers. On average, Browser-X is slower by a factor of 1.82. To minimize the impact from network and query type, we only plot it for queries that do not contain images and come from mainly-residential ASNs; other types of queries and ASNs show a similar behavior. As we can see from the figure, the two browsers have disparate performance. Controlled experiments on the same machine with different browsers confirm that the differences in T_{script} times that we see in the wild reflect real differences in browser performance, and not just differences in the capabilities of host machines.

The combination of the two observations—variations in the browser mix and the differences in browser speed across different types—explains why browsers play an important role in SRT variations.

To understand *why* the mix of browsers varies with time, we investigated where their users come from. Figure 4.13 shows the fractions of queries coming from three different categories of networks for the two popular browsers. We see that the vast majority of the Browser-Y's queries come from mainly-residential networks. For Browser-X, however, there is a significant shift (around 40%) during peak hours away from mainly-residential networks. We speculate that these trends stem from the fact that Browser-X, at the time of this study, is more likely to be the standard browser that is adopted and supported officially by many enterprises. This speculation is supported by the fact Browser-X is an older browser, while Browser-Y is newer; many enterprises in the USA do not immediately upgrade to newer browsers as they must first test compatibility of the new browser with their internal services (often called line-of-business applications). On the other hand, residential machines (at least in the USA, the geographic region to which we limit our study) are likely upgraded sooner due to automatic updates. Thus, as the users of the search service move from work to home, the service sees a move from Browser-X to Browser-Y.

Finally, we observe that the impact of browsers on SRT is the opposite of the impact of network and query type. During off-peak hours, a greater fraction of queries come from Browser-Y, which has better performance. This should have a positive impact on SRT during off-peak hours. But the negative impact on SRT that comes from network and query type variations dominates, and we see higher SRTs as the net effect. Without the corrective effect of browsers, the SRTs during off-peak hours would have been even

higher.

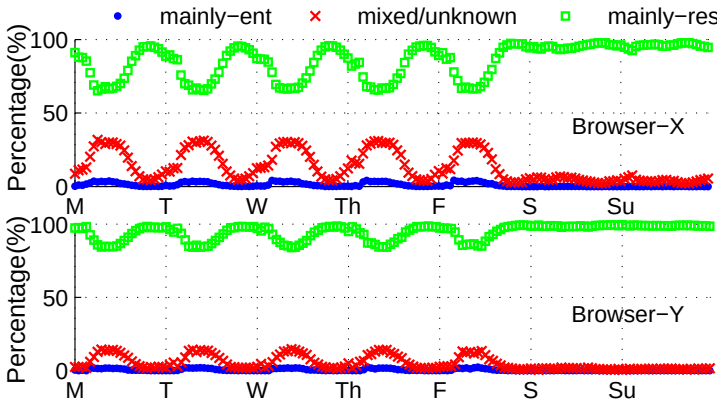

Figure 4.13: %-age of queries from diff. nets.

4.5 Anomalous SRT Variations

In addition to systemic variations that we study above, SRT also experiences irregular variations. These variations, which we call anomalies, stem from events such as failures in the network or data center, congestion, and attacks on the search infrastructure. For a good user experience, anomalies must be detected and resolved quickly. However, operators inform us that it frequently takes them several days before they even detect that the SRT is anomalous, as the systemic variations often hide the real anomalies. Once detected, diagnosing the root cause of an anomaly is also challenging as it can lie in any part of the infrastructure.

We present an online detection and diagnosis tool to assist the operators in quickly resolving anomalies. We first describe how we detect anomalies in SRT and then how we localize their root causes. Our tool has been running on the real system for five months.

4.5.1 Detecting Anomalies

The main challenge in accurately detecting SRT anomalies is that they co-exist with systemic variations due to other factors, including the weekly, daily and hourly fluctuations, and the long-term evolution of SRT. As we show later, due to these variations, common anomaly and outlier detection methods both fail to detect many anomalies and flag events that are not anomalies.

Methodology

We use an approach based on time series decomposition [35], which we call WoW (week-over-week) analysis. The basic idea is to view SRT as a composition of three components: *i*) the long-term trend; *ii*) the seasonality or periodic behavior; and *iii*) fast variations or noise.

Consider the SRT time series, where SRT_t is the average over the t-th hour. The long-term trend component, denoted by L, of this series can be computed as a centered moving average with the window size set to T:

$$L_t = \frac{1}{T+1} \sum_{i=-T/2}^{T/2} SRT_{t+i} \tag{4.1}$$

T should be greater than or equal to the maximum periodicity in the data. We use T=168 hours (1 week).

Let Y_t=SRT_t–L_t be the time series after removing the long-term trend. Then, the seasonal component, S, can be computed as seasonal moving average:

$$S_t = \frac{1}{M+1}\sum_{i=0}^{M} Y_{t-iT}, M = \lfloor t/T \rfloor \tag{4.2}$$

What remains now is the noise component, N, which can be computed by removing the long-term trend and seasonality components:

$$N_t = SRT_t - L_t - S_t \tag{4.3}$$

By definition, this component is neither part of the long term trend nor a periodic event. It captures the irregularity that cannot be explained by the other two factors.

We deem as anomalous time instances where the noise is abnormally high. To infer high abnormality, we assume that noise follows a Gaussian distribution. However, due to the diurnal patterns and the day of week effect, the distribution parameters (i.e., mean and standard deviation) can differ for different times of week.

Thus, based on historical data, we learn 168 Gaussian models, one for each hour of the week. We flag as anomalous values outside the 95th percentile of the distribution, i.e., the SRT at time t is anomalous if:

$$\frac{|N_t - \mu_{\hat{t}}|}{\sigma_{\hat{t}}} > 1.96 \tag{4.4}$$

where $\mu_{\hat{t}}$ and $\sigma_{\hat{t}}$ are the mean and standard deviation of the Gaussian distribution built from all data collected at the same hours as t within a week.

The quantity on the left in Eq. 4.4 is the *severity* of the anomaly, and we report it in the notifications that are generated for the operators. It captures the extent to which current SRT has deviated from expectation. Larger values indicate more serious anomalies.

We repeat the above steps for each incoming data point to conduct online anomaly detection. If an hour is anomalous, its data is excluded while learning the Gaussian model. We found that including data from anomalous hours leads to less robust anomaly detection.

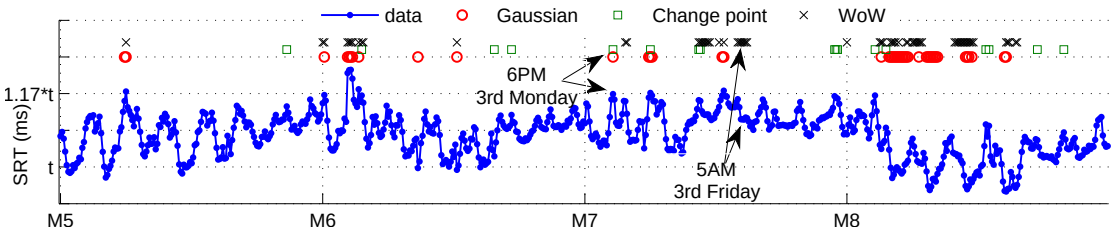

Figure 4.14: Comparing anomaly detection results for three different techniques.

Results

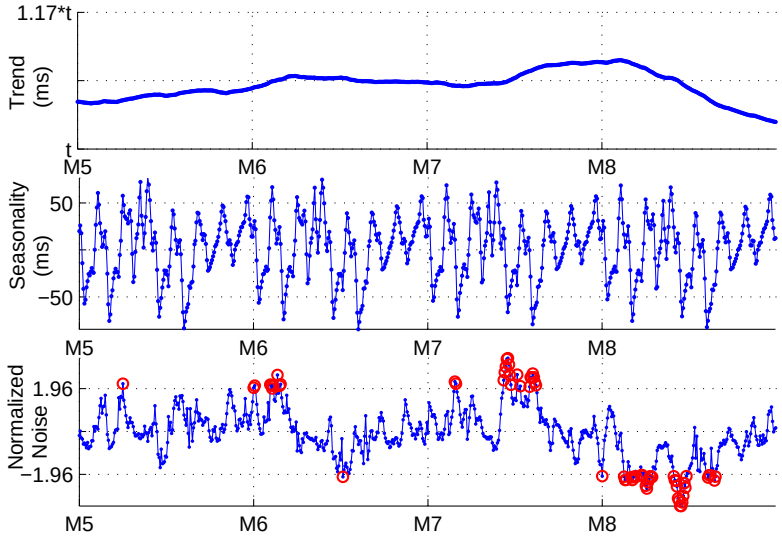

Figure 4.15: Time series decomposition results. Circles in the bottom graph denote anomalies.

To evaluate our method, we compare it to two common methods for anomaly detection that have frequently been used in prior work. The first method builds a single Gaussian model of SRT using recent history, and it detects anomalies when the current value is an outlier. The second method is based on change point detection [41]. To eliminate the potential trend change, sudden changes in SRT are detected as change points using cumulative sum and bootstrapping techniques [41]. A Gaussian model is built for all the hourly data points between each pair of change points. In both methods, a value v is deemed anomalous if $\frac{|v-\mu|}{\sigma} > 1.96$, where μ and σ are the parameters of its learned model.

We first illustrate the behavior of the three approaches and then quantify their overall performance. For an examplary four-week period, Figure 4.14 shows the anomaly

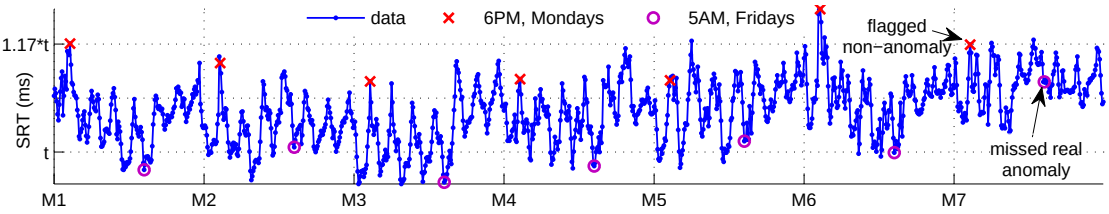

Figure 4.16: Historical SRT behaviors of the anomalies flagged or missed using Gaussian and change point techniques.

detection results (overlaid on top of SRT) for each approach. Figure 4.15 shows the time series decomposition results of WoW analysis for the corresponding period.

As we see in Figure 4.14, the three approaches do behave differently. The simple Gaussian model approach only detects globally huge spikes, which can overlap with the systemic variations. The change point approach detects only local spikes between any two change points. The WoW approach can detect not only globally huge spikes, but also anomalies that are not visibly apparent due to interference from systemic variations. As one example, for 5AM on the 3rd Friday, an anomaly was detected by our WoW approach but was missed by the other approaches. This was a real anomaly; historical behavior of SRT, shown in Figure 4.16, reveals that the SRT at 5AM on Fridays is generally much lower.

The conventional approaches are fooled by systemic variations from the other perspective too. That is, they flag non-anomalies. As one example, these approaches detected an anomaly at 6PM on the 3rd Monday in Figure 4.14, but WoW did not flag that time period. As we can see from the historical data in Figure 4.16, this time period does not have significantly degraded SRT compared to 6PM on past Mondays.

We now perform a more systematic evaluation. The performance of anomaly detection can be quantified using false negative and positive rates. False negatives are cases in which a real anomaly is missed, and false positives are cases in which a non-anomaly is flagged.

False negatives: Quantifying the false negative rate requires as a reference a complete list of all anomalies in the system. However, such a list is rarely available for a complex, real system. We use instead the ticket database that is maintained by the search provider. The anomalies documented in this database have been manually detected by the operators (e.g., based on visual inspection of the data or user complaints), or they

have been flagged by an existing tool (which does not account for SRT variations) and later verified manually.

We find that over the course of five months, 90%, 65%, and 60% of all the anomalies present in this database are identified by WoW, Gaussian, and change point approaches. That is, while WoW missed 10% of the anomalies (as the user complaints can be caused by other issues, *e.g.*, availability other than the SRT degradation), the other approaches miss 3-4 times as many.

Comparing the ticket database and our tool, we find that the anomalies in the database tend to have high severity values (> 2.5, or outside the 99th percentile of the Gaussian model). This implies that with the current practice, operators could detect only highly anomalous events. Further, our tool flags many anomalies that are not in the ticket database. If these anomalies are not false positives (which we study next), they represent anomalies that the current practice misses.

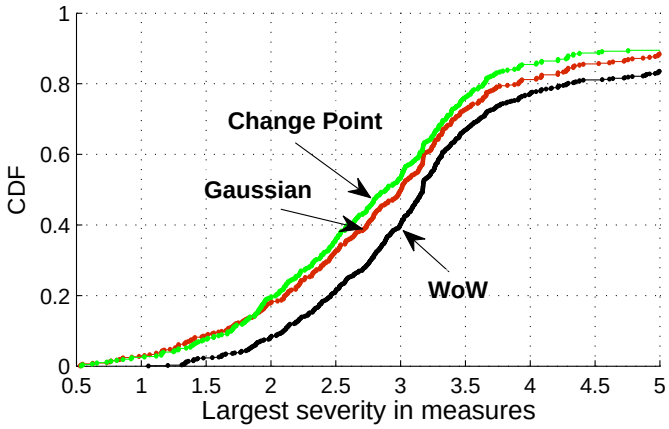

Figure 4.17: Distribution of largest severity across all measures.

False positives: Estimating the false positive rate of an approach tool is challenging. Investigating an anomaly can take a huge amount of effort and thus operators do not investigate each anomaly. For instance, short-lived anomalies that disappear before an operator has had a chance to investigate are not investigated. (However, operators still want to detect and log all anomalies; this record helps quantify service reliability and determine if some components fail repeatedly.) Thus, we cannot be certain if a certain anomaly that was flagged by our tool but not investigated is a true or a false positive.

To estimate the false positive rate, we emulate the method used by the operators

as a first step towards investigating an anomaly. The operators observe the behavior of other, fine-grained measures (e.g., T_{net}) during the anomaly period. If one or more of those measures is anomalous as well, the anomaly is deemed as likely real and further investigation is conducted. If none of those measures is anomalous, the anomaly is deemed as false. The assumption is that if the anomalous behavior in SRT correlates to anomalous behavior in at least one of the fine-grained measures, then we can be confident that this is a real anomaly.

Thus, we apply the same WoW technique to the 14 fine-grained measures (§4.4) and compute the largest severity value across all measures. Figure 4.17 shows the distribution of the largest severity value as an SRT anomaly is detected using three different techniques. We consider an SRT anomaly to be a false positive, if the largest severity value does not indicate an anomaly, that is, it is under 1.96. Based on this criteria, we see that the false positive rate of WoW is 7%, while that of Gaussian (17%) and change point (19%) approaches is at least twice as much.

4.5.2 Anomaly Diagnosis

In addition to detecting anomalies, our tool helps operators diagnose the root cause of those anomalies, by identifying at a coarse-level the most likely source of anomalies in SRT. For our purposes, possible sources are client behavior, the data center, and the network between the clients and the data center (which includes the CDN servers). Client behaviors can be a source of anomalies due to attacks (e.g., bots generating a lot of queries), among other possibilities.

The inference of our tool is then combined with other lower-level measures (e.g., output of tools that monitor network or server health and utilization) to localize the root cause at a finer-granularity. These low-level measures are by themselves insufficient for root cause localization because they are noisy and their impact on SRT is otherwise unclear [42].

Methodology

The anomaly diagnosis functionality of our tool is invoked whenever an SRT anomaly is detected. Its starting points are the time series of the 14 measures that we used to study systemic SRT variations.

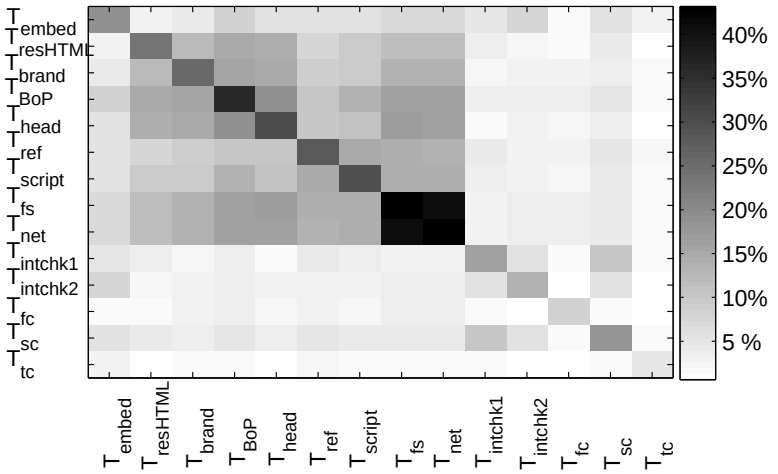

Figure 4.18: %-age of anomalies for each pair of measures.

We face two main challenges. First, due to complex interactions among different measures, simply using the anomaly severity value of individual time series does not suffice. For example, when an anomaly happens in the network, both $T_{resHTML}$ and T_{net} can be anomalous. Second, during anomalies, the relationships among the different measures may vary from the normal case. For example, under normal circumstances, T_{fs} (which is $T_{net} + T_{fc}$) and T_{net} are highly correlated usually, but if T_{fc} is anomalous due to a failure in the data center, this correlation disappears.

Thus, a better understanding of the relationships among the various measures *during* SRT anomalies is important. To obtain insight into these relationships, we investigate how often two measures are simultaneously anomalous during SRT anomalies. Figure 4.18 shows the percentage of SRT anomalies in which a pair of measures, specified on the x- and y-axis, appear as anomalies at the same time. The diagonal values are the percentage of SRT anomalies when the measures, specified on the x-axis (or the y-axis), appear as an anomaly by themselves. The graph is symmetric along the diagonal.

We make the following observations from this figure. First, focusing on T_{net}, a pure network-side measure, we see that T_{fs} is almost always anomalous along with it. The server-side measures (T_{fc}, T_{sc}, T_{tc}) and client-side measures that are highly dependent on the server behavior ($T_{intchk1}$ and $T_{intchk2}$) are rarely anomalous in conjunction with T_{net}. Other browser-side measures are sometimes anomalous in conjunction with T_{net}. Second, let us focus on pure server-side measures (T_{fc}, T_{sc}, T_{tc}). They appear anomalous mostly on their own, except for *i*) $T_{intchk1}$, which is a browser side measure that is highly

dependent on server-side latency; and *ii*) $T_{intchk2}$ and T_{embed}, which we can ignore as they contribute less than 1% of the total SRT. Third, the anomalies in browser-side measures ($T_{resHTML}$, T_{brand}, T_{BoP}, T_{head}, T_{ref}, and T_{script}) are weakly correlated with server- and network-side measures, indicating their anomalies likely stem from client issues.

Based on the observations above, to diagnose anomalies, we first divide the measures into three classes: Network (T_{net} and T_{fs}), Server (T_{fc}, T_{sc}, T_{tc}), or Client ($T_{resHTML}$, T_{brand}, T_{BoP}, T_{head}, T_{ref}, and T_{script}). If only one class of measures is anomalous, then we deem the problem likely lies in the corresponding class. We saw above that, frequently, only one class is anomalous when SRT is anomalous.

If measures in more than one class are anomalous, we use the following decision logic to determine the likely root cause (or prioritize the investigation). First, if any Server measures are anomalous, we deem that the anomaly is in the data center. This heuristic is based on the second observation above and the fact that the server-side measures are collected using a separate instrumentation (not Javascript), which is not supposed to be affected by the impact from the network or client-side behaviors.

Second, if T_{net} is anomalous, but not any Server measure, then we consider it as an anomaly associated with the network. The underlying rationale is that because of the way we compute T_{net} (§4.3), it is not intimately dependent on client behavior. This is also supported by the first observation above. Finally, we consider that the remaining cases are due to client behaviors (e.g., bot attacks).

Results

Using the diagnosis method described above, we determine the root cause for each of the detected anomalies and compare them with what is inferred and logged by the operators in the ticket database. We find that our results are perfectly consistent with the ticket database for those anomalies that are common to both methods.

4.5.3 Experience

We now report briefly on the experience of running our tool for five months. We first describe three anomalies that were detected and diagnosed, and then present overall

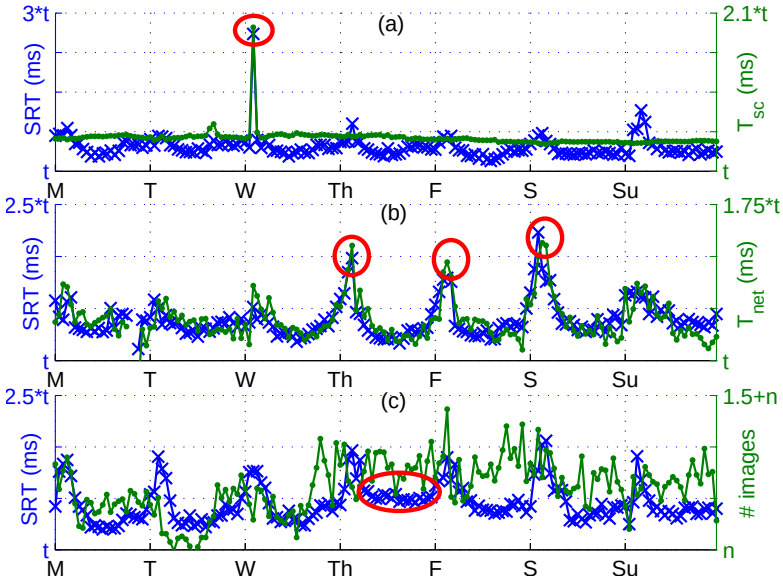

Figure 4.19: Example anomalies.

statistics. Figure 4.19 shows the three example anomalies. For each, SRT uses (blue) crosses and the left y-axis, another metric of interest uses (green) dots and the right y-axis, and the anomalous periods are marked using (red) circles. Figure 4.19(a) shows the first anomaly, which was detected soon after the SRT became abnormally high. For this anomaly, our tool diagnosed that the backend data center was the likely culprit because T_{sc} was anomalous at the same time. Using this information, the operators were able to quickly determine the exact root cause. They were then able to mitigate the impact of the failure, which resulted in SRT returning to normal levels.

Figure 4.19(b) shows the second anomaly that was caused by a re-configuration of the routing weights between the CDN edge servers and the backend data centers. The SRT at one of our data centers was detected to be extremely high in the early morning hours of Thursday. The anomaly was diagnosed as a network related issue using our tool, as it was accompanied by anomalous T_{net}. To fully diagnose this anomaly, because our tool blamed the network, the operators combined our results with a log of operational events from the network. They found that a network reconfiguration had led to a substantial decrease in the overall client BW to this data center at the early morning hours.

Figure 4.19c shows the third anomaly. The culprit here was client behavior. More specifically, clients were sending queries that generated richer response pages. It turns out that this was because of a major holiday event. Although no action needs to be

taken to resolve this performance degradation, our tool helped the operators eliminate the possibilities of failures in the data centers or the network.

Across all SRT anomalies that our tool detected, we find that the fractions of the anomalies that were attributed to the wide-area network, data center, and client behaviors are 37%, 27%, and 36%, respectively. That the culprits are almost evenly distributed (with the data center being slightly lower), means that there is no silver bullet to significantly reducing the bulk of the SRT anomalies. The provider must work on reducing the impact of failures and unexpected behaviors on all three fronts. Perhaps the most surprising aspect is that client behaviors account for a third of the anomalies. Most of these are attacks on search infrastructure, but some are also caused by sudden changes in query richness (e.g., following a significant newsworthy event).

4.6 Implications and Discussion

Our work has several implications for understanding, diagnosing, and managing the performance of large-scale Web services. We now discuss a few of them.

End-to-end performance management: Contrary to the common belief in research and practitioner communities [43, 44], our findings show that focusing on improving server processing time alone is insufficient. Much of the variation in user performance now stem from factors such as network paths, browsers, and query types. While these factors are not under the direct control of the service provider, there are several ways in which their impact may be minimized.

For network paths, the service provider can send simpler pages for queries that come from clients behind slow last mile links. Using a data source that is different from the one used in this chapter we find that the middle mile between the data center and the edge exhibits significant performance variations as well. This variations can be controlled through dedicated capacity between these points. One major provider appears headed in that direction already [45].

To ensure consistently fast SRTs across browsers, the providers can customize the responses and scripts to account for the strengths and weaknesses of individual browsers. Our data indicates that customization for the top three or four browsers would cover the vast majority of the queries.

To ensure fast SRTs for rich queries, the CDN edge servers can help reduce the burden on the user's end (browser) by simplifying dynamically the layout of the page and pre-processing some of the scripts. They can also cache more of the CSS, images, and javascript. Caching on the edge is not used heavily today because bits of the response pages are personalized. Small amount of computational capabilities on the edge servers (for generating personalized portions of the page) and a different layout of the response page can improve the amount of the content that users can fetch from the edge servers.

Monitoring via performance-equivalent request classes: Our work highlights the difficulty of reasoning about the service performance when it is measured across all users because such a measure is tainted by systemic changes in behaviors and characteristics of user population. This holds not only for search but to other Web services as well. For

instance, Zander et al. [46] observed different fraction of users with IPv6 capabilities on weekends vs. weekdays for many services. (They could not fully explain this variation. Our results confirm their suspicion that it stems from residential and enterprise users having different characteristics.)

To provide greater insight into service performance, we are developing a monitoring engine to partition requests into equivalence classes that account for expected performance differences across network paths, browsers, and query types. This enables the detection of performance degradation for each class, which otherwise gets hidden under the variations across classes. Our provider's current query classification is based solely on user intent, e.g., travel-related and person search, to help generate meaningful responses. It does not help with performance analysis, which is the goal of the classification that we are developing.

Diagnosis using high- and low-level measures: A final difficulty highlighted by our work is that of detecting and diagnosing anomalous events such as failures. Past research has highlighted the importance of using end-to-end performance metrics, rather than relying on low-level metrics such as server processing time, CPU or network utilization, because the anomalies in low-level measures may or may not cause anomalies in user experience [42]. While we agree with this position in principle, our work shows the difficulty of using end-to-end measures as they exhibit complex behavior due to reasons other than failures (e.g., user behavior). Prior work has looked at systems with roughly stationary response time characteristics because of a lack of significant diversity in query types, browsers, and network paths, as may be the case in enterprise networks [42, 47]. But in the Web services context, effective diagnosis of response time requires factoring out the impact of systemic variations and appropriately combining high- and low-level metrics.

4.7 Related Work

Our work builds upon much prior work on the performance of Web services. One thread of work takes a client-side perspective to monitor the performance that a client experiences to various Web sites or to uncover factors that impact Web page load times. For example, WebPagetest [48] provides a tool to measure the performance to any Web site and provides a timeline of relevant events that occur while the page is being loaded; and Butkietz et al. [40] examines the impact of page structure and server location of various Web sites on a client's performance. In contrast, our work takes a provider-side perspective, to uncover factors that impact the performance delivered by one Web service to all its active clients.

Another thread of work focuses on improving different aspects of Web services such as request processing, page design, and content delivery. For example, several works seek to make request processing predictable in a data center [49, 50]; tools exist to help developers follow the best practices for site layout and design [51, 52]; and many researchers have studied several facets of CDN design such as placement and TCP behavior of edge servers [2, 53, 54, 55, 56] and proposed enhancements such as better redirection and caching, and hybrid CDNs [57, 58, 59, 60, 61]. In contrast to this body of work, we take an end-to-end view, which includes all relevant aspects, of the performance of a Web service. Further, instead of focusing on specific design enhancements, we seek to explain and diagnose performance variations given the design of a large-scale service.

4.8 Summary

Our work is motivated by the observation that SRT, as observed by a large provider, varies widely and, surprisingly, increases during off-peak hours when the query load is lower. Through a detailed analysis, we found that this variation stems from systemic shifts in user characteristics—the source networks, the nature of queries, and the browsers used. In contrast, server processing times are relatively stable. To help operators detect and diagnose anomalous SRTs that often get hidden by these systemic changes, we developed and deployed a diagnostic tool that works by isolating the impact of such changes from anomalous events. We showed that it is significantly more effective than methods that ignore systemic variations. Over a five month period, it detected 30% more real anomalies.

Overall, our work highlights the importance of factoring in user characteristics when monitoring, optimizing, and diagnosing performance of user-facing services. We believe that its findings and methods are relevant to Web services beyond search.

Chapter 5

Conclusion and Discussion

5.1 Conclusion

In this thesis, we conducted extensive measurement studies on the existing content distribution systems using Yahoo! and search services as case studies. Our aim is to understanding the current practices and limitations in the existing content distribution framework, and the challenges we face to further improve the performance in terms of its latency, scalability, and robustness.

For this, we picked three studies that focus on back-end data centers, edge servers, and user characteristics, respectively. The conclusions drawn in these three studies not only helped us better understand the impact of each individual entity on the overall end-to-end user experience, but also how they interact with each other in leading to future key design decisions.

First, to understand the impact from the back-end data centers, we presented a first look at the inter-data center traffic characteristics in Yahoo!. As one of the largest content providers both inside and outside US, Yahoo! faces great challenges in scaling to large user population. Our study shows that to address this challenge, Yahoo! employs a hierarchical way of deploying its data centers. In the meantime, by teasing out different types of traffic being carried at Yahoo! back-end, we found that background D2D traffic is quite dominant in the aggregate D2D traffic. Our investigation over different types of D2C traffic further suggested that there are strong correlations among different D2C traffic.

Second, to characterize the roles of the front-end edge servers in the end-to-end dynamic content distribution, we compared two different practices currently adopted in Bing and Google. While Google has much smaller number of servers being deployed, and they sit not as close to the users as Akamai – the main CDN for Bing search service, Google search service exhibits much better latency and robustness performance. Our study highlights the trade-off between the placement of front-end servers and the front-end to back-end fetch time.

Third, to have a deeper understanding of the impact from user characteristics, and how it interacts with the server-side processing time, and network time in affecting the overall latency performance, we studied one of the largest search services in US by collecting massive amount of measurement data through detailed client- and server-side instrumentation. Our findings are that the server processing time plays a much less important role in the overall SRT systemic variations. Rather, the user-side variations including network characteristics, types of queries, and types of browsers from peak to off-peak hours play much bigger roles. The study culminated in the discussion of an anomaly detection and diagnosis tool that accounted for the systemic variations. It proved to be an useful tool that can significantly help the content providers improve their robustness.

5.2 Discussion

The studies conducted in this thesis not only reveal the current industrial practices in dealing with performance issues such as latency, scalability, and robustness, but also shed lights on the design of future content distribution systems.

The Yahoo! study showed that background D2D traffic is quite dominant in the aggregate D2D traffic. This is an indication of great potential to optimize the background traffic such as the periodic data replications, background search index computation, etc. The same study also revealed that there are high correlations among different application services. The observations have important implications on distributing different services at multiple data centers, for instance, placing highly correlated services in the same data center in order to reduce the inter-data center fetch time. These design principles can greatly improve both the latency performance and its scalability.

The findings in our second study suggested that while placing front-end servers closer to users can help reduce latency, other key factors such as processing times and loads at both front-end servers and back-end data centers as well as the quality of (physical and TCP) connections between them also play a critical role in determining the overall user-perceived latency performance. Therefore, instead of deploying huge number of edge servers, and keeping them as close to the users as possible as most existing CDNs do, a careful design of the front-end edge server placement strategy can potentially lead to improved end-to-end user-perceived latency.

The observations achieved in our last study highlighted the importance of factoring in user characteristics when monitoring, optimizing, and diagnosing performance of user-facing services. To minimize the impact of user characteristics in the overall latency performance, content providers can send simpler pages to users coming from slower network, customize responses and scripts to account for the strengths and weaknesses of individual browsers, and utilize the CDN edge servers to cache CSS, images, and javascript, to pre-processs some of the scripts, and to simplify the layout of the page dynamically. To enhance the robustness of the system, our last study also suggested that operators should take into consideration the user characteristics for achieving better false positives and false negatives in detecting and diagnosing the anomalies residing in the system.

References

[1] Y. Chen, S. Jain, V.K. Adhikari, Z.L. Zhang, and K. Xu. A first look at inter-data center traffic characteristics via yahoo! datasets. In *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011.

[2] Y. Chen, S. Jain, V.K. Adhikari, and Z.L. Zhang. Characterizing roles of front-end servers in end-to-end performance of dynamic content distribution. In *IMC*, 2011.

[3] Y. Chen, R. Mahajan, B. Sridharan, and Z.L. Zhang. Understanding and diagnosing search response time: A provider-side view. In *SIGCOMM*, 2013.

[4] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. The nature of data center traffic: measurements & analysis. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 202–208, New York, NY, USA, 2009. ACM.

[5] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Understanding data center traffic characteristics. In *WREN '09: Proceedings of the 1st ACM workshop on Research on enterprise networking*, pages 65–72, New York, NY, USA, 2009. ACM.

[6] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. Moving beyond end-to-end path information to optimize cdn performance. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 190–201, New York, NY, USA, 2009. ACM.

[7] Vijay Kumar Adhikari, Sourabh Jain, and Zhi-Li Zhang. Youtube traffic dynamics and its interplay with a tier-1 isp: An isp perspectives. In *IMC '10: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement conference*, New York, NY, USA, 2010. ACM.

[8] Yahoo! research webscope program. `http://labs.yahoo.com/Academic_Relations`.

[9] T. Brekne, A. Årnes, and A. Øslebø. Anonymization of IP Traffic Monitoring Data: Attacks on Two Prefix-Preserving Anonymization Schemes and Some Proposed Remedies. In *Privacy Enhancing Technologies*, pages 179–196. Springer, 2005.

[10] J. Fan, J. Xu, M.H. Ammar, and S.B. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks*, 46(2):253–272, 2004.

[11] S. Coull, C. Wright, F. Monrose, M. Collins, M. Reiter, et al. Playing devil's advocate: Inferring sensitive information from anonymized network traces. In *Proceedings of the Network and Distributed System Security Symposium*, pages 35–47. Citeseer, 2007.

[12] S. Coull, MP Collins, CV Wright, F. Monrose, MK Reiter, et al. On web browsing privacy in anonymized netflows. In *Proceedings of the 16th USENIX Security Symposium*, pages 339–352, 2007.

[13] Routeviews. University of oregon route views archive project. *http://archive.routeviews.org*, 2010.

[14] J. Rexford. Route optimization in IP networks. *Handbook of Optimization in Telecommunications*, pages 679–700, 2005.

[15] Neil Spring, Ratul Mahajan, and Thomas Anderson. The causes of path inflation. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 113–124, New York, NY, USA, 2003. ACM.

[16] List of tcp and udp port numbers. `http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers`.

[17] TCP/IP Ports. In *url http://www.chebucto.ns.ca/~rakerman/port-table.html*.

[18] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. Moving beyond end-to-end path information to optimize cdn performance. In *IMC '09*, pages 190–201, New York, NY, USA, 2009. ACM.

[19] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. Measuring and evaluating large-scale cdns (paper withdrawn). In *IMC '08*, pages 15–29, New York, NY, USA, 2008. ACM.

[20] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D.A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, page 24. ACM, 2007.

[21] A. Pathak, Y. Wang, C. Huang, A. Greenberg, Y. Hu, R. Kern, J. Li, and K. Ross. Measuring and evaluating TCP splitting for cloud services. In *Passive and Active Measurement*, pages 41–50. Springer, 2010.

[22] Mukarram Tariq, Amgad Zeitoun, Vytautas Valancius, Nick Feamster, and Mostafa Ammar. Answering what-if deployment and configuration questions with wise. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pages 99–110, New York, NY, USA, 2008. ACM.

[23] M.H. Bateni and M.T. Hajiaghayi. Assignment problem in content distribution networks: unsplittable hard-capacitated facility location. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 805–814. Society for Industrial and Applied Mathematics, 2009.

[24] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, 2004.

[25] M. El Dick, E. Pacitti, R. Akbarinia, and B. Kemme. Building a peer-to-peer content distribution network with high performance, scalability and robustness. *Information Systems*, 2010.

[26] H. Jiang, Z. Wang, A.K. Wong, J. Li, and Z. Li. A replica placement algorithm for hybrid CDN-P2P architecture. In *2009 15th International Conference on Parallel and Distributed Systems*, pages 758–763. IEEE, 2009.

[27] J. Ravi, Z. Yu, and W. Shi. A survey on dynamic Web content generation and delivery techniques. *Journal of Network and Computer Applications*, 32(5):943–960, 2009.

[28] M.P. Wittie, V. Pejovic, L. Deek, K.C. Almeroth, and B.Y. Zhao. Exploiting locality of interest in online social networks. In *CoNEXT*, page 25. ACM, 2010.

[29] D.M. Lewin, A.T. Davis, S.D. Gendler, M. Kagan, J.G. Parikh, and W.E. Weihl. Dynamic content assembly on edge-of-network servers in a content delivery network, July 6 2010. US Patent 7,752,258.

[30] Microsoft online services data center locations. `http://www.championcloudservices.com/data-center-locations/`.

[31] Google data centers. `http://www.google.com/corporate/datacenter/locations.html`.

[32] Seattle. `https://seattle.cs.washington.edu/html/`.

[33] Official google webmaster central blog: Using site speed in web search ranking. `http://googlewebmastercentral.blogspot.com/2010/04/using-site-speed-in-web-search-ranking.html`.

[34] R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web: Listen to your customers not to the hippo. In *Proc. of SIGKDD*, 2007.

[35] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002.

[36] Progressive rendering via multiple flushes. `http://www.phpied.com/progressive-rendering-via-multiple-flushes/`.

[37] Measure page load time. `https://developers.google.com/speed/docs/pss/LatencyMeasure`.

[38] How to measure page load time with google analytics. `http://blog.yottaa.com/2010/10/how-to-measure-page-load-time-with-google-analytics`.

[39] Analysis of variance. `http://www.stat.columbia.edu/~gelman/research/unpublished/econanova.pdf`.

[40] M. Butkiewicz, H.V. Madhyastha, and V. Sekar. Understanding website complexity: Measurements, metrics, and implications. In *IMC*, 2011.

[41] W.A. Taylor. Change-point analysis: a powerful new tool for detecting changes. *preprint, available as http://www. variation. com/cpa/tech/changepoint. html*, 2000.

[42] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D.A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *ACM SIGCOMM Computer Communication Review*, 2007.

[43] L.A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. *Micro, IEEE*, 2003.

[44] T.F. Abdelzaher, K.G. Shin, and N. Bhatti. Performance guarantees for web server end-systems: A control-theoretical approach. *Parallel and Distributed Systems, IEEE Transactions on*, 2002.

[45] R. Krishnan, H.V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize cdn performance. In *IMC*, 2009.

[46] S. Zander, L.L.H. Andrew, G. Armitage, G. Huston, and G. Michaelson. Mitigating sampling error when measuring internet client ipv6 capabilities. In *IMC*, 2012.

[47] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *ACM SIGCOMM Computer Communication Review*. ACM, 2009.

[48] Webpagetest - website performance and optimization test. `http://www.webpagetest.org/`.

[49] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006.

[50] R.W. White and S.T. Dumais. Characterizing and predicting search engine switching behavior. In *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009.

[51] Pagespeed tools family. `https://developers.google.com/speed/pagespeed/`.

[52] Yslow. `http://yslow.org/`.

[53] M. Al-Fares, K. Elmeleegy, B. Reed, and I. Gashinsky. Overclocking the yahoo!: Cdn for faster web page loads. In *IMC*, 2011.

[54] K.L. Johnson, J.F. Carr, M.S. Day, and M.F. Kaashoek. The measured performance of content distribution networks. *Computer Communications*, 2001.

[55] C. Huang, A. Wang, J. Li, and K.W. Ross. Measuring and evaluating large-scale cdns. In *Proceedings of IMC*, 2008.

[56] S. Triukose, Z. Wen, and M. Rabinovich. Measuring a commercial content delivery network. In *Proceedings of the 20th international conference on World wide web*. ACM, 2011.

[57] C. Gkantsidis and P.R. Rodriguez. Network coding for large scale content distribution. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*.

[58] Y.J. Song, V. Ramasubramanian, and E.G. Sirer. Optimal resource utilization in content distribution networks. *Cornell University*, 2005.

[59] H. Jiang, J. Li, Z. Li, and X. Bai. Performance evaluation of content distribution in hybrid cdn-p2p network. In *Future Generation Communication and Networking*. IEEE, 2008.

[60] J. Kangasharju, K.W. Ross, and J.W. Roberts. Performance evaluation of redirection schemes in content distribution networks. *Computer Communications*, 2001.

[61] M.J. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with coral. NSDI, 2004.