Enabling Data-Driven Optimization of Quality of Experience in Internet Applications

Junchen Jiang

CMU-CS-17-119
September 11, 2017

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee
Hui Zhang, Chair
Vyas Sekar, Chair
Srinivasan Seshan
Peter Steenkiste
Ion Stoica, UC Berkeley

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under grant numbers CNS-1345305, CNS-1040801, and CCF-1536002, and by Intel Corporation Santa Clara under grant number 1011555. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Internet applications, Quality of Experience, Data-Driven Networking, CFA, VIA, Pytheas

Abstract

The Internet is an eyeball economy dominated by applications such as Internet video and Internet telephony, whose revenue streams crucially depend on user-perceived Quality of Experience (QoE). Despite intense research towards better QoE, existing approaches have failed to achieve the QoE needed by today's applications, because they are not acting on the right signal and in the right place: they either seek to rearchitect the in-network devices which have little information on user-perceived QoE, or rely on end-to-end protocols which have limited knowledge on network conditions.

The key contribution of this dissertation is to bridge the long-standing gap between the visibility to user-perceived QoE and the visibility to network conditions by a data-driven approach. Our thesis is that *one can substantially improve QoE by maintaining a global view of up-to-date network conditions based on the QoE information collected from many endpoints*. In essence, this thesis revisits the question of where to implement the functionality of QoE optimization. Unlike prior work which optimizes QoE by in-network devices or individual endpoints, our approach uses a logically centralized controller to optimize QoE, which retains endpoints' visibility to QoE while attaining a global view of real-time network conditions by consolidating information from many endpoints.

To prove the thesis, this dissertation provides a suite of solutions to address two fundamental challenges. First, we need expressive models to capture complex relations among QoE, decisions, and application sessions who share similar QoE-determining factors. Second, we need scalable platforms to make real-time decisions with fresh data from geo-distributed clients.

Our key insight is that there are *persistent critical structures* in the relations between QoE and session-level features. These structures allow us to build expressive models that can identify network sessions with similar QoE-determining factors, and their temporal persistence allows us to build scalable platforms by decoupling offline structure-learning processes and real-time decision making processes. We have developed algorithms and end-to-end systems, which integrate machine-learning techniques with our insight of persistent critical structures. We have used real-world deployment and simulation driven by real datasets to show that our solutions can yield substantial QoE improvement and consequently higher user engagement for video streaming and Internet telephony than existing solutions as well as many standard machine learning solutions.

Acknowledgments

First and foremost, I am immensely grateful to my advisors, Hui Zhang and Vyas Sekar, both of whom have been tireless mentors, counselors, co-authors, and career advisors. Vyas has been instrumental in teaching me how to focus on the critical aspects of any research project, and formulate my thoughts in the right context and in the right way. Most important of all, I learned from him how to approach research in a rigorous and efficient manner. Hui has been an unlimited source of insightful guidance and vision. I was deeply indebted to him for inspiring and encouraging me to pursue the research of application quality and its confluence with data-driven paradigm. Throughout the years, he has always offered insightful advises at the right moments, and remarkably, every meeting I recall with him ended with high spirit. I cannot imagine any team of academic advisors with a better match of personality, research approach, and advising style than Vyas and Hui. I am extremely lucky to have had the benefit of their vision and wisdom in these formative years of my research career.

In completing this dissertation, I am also intellectually and personally indebted to Ion Stoica. He gave tremendous help in every project I worked on with Conviva. He was among the first to point out the difference between the prior approach and data-driven networking lies in that the former is driven by single-flow information, while the latter by multi-flow information. His sharp questions and comments have always pushed me to dig deep into details. I am thankful for his honest feedback that improved me as a researcher.

I am grateful to other members of my committee, Peter Steenkiste and Srinivasan Seshan, who have been an unlimited source of honest and constructive comments. This dissertation is profoundly influenced by the comments that Srini drew from the parallel between this dissertation and his prior work of SPAND and congestion manager in the transport layer, and many of Peter's constructive comments that brought more coherence among the solutions of different applications. I am also thankful to them for giving me the opportunity to be the teaching assistant for their computer networks courses.

Throughout my PhD years, I have been greatly benefited from two fruitful collaborations with Conviva and Microsoft. I would like to give my gratitude to colleagues at Conviva, Jibin Zhan, Aditya Ganjam, Davis Shepherd, Henry Milner, Faisal Siddiqui, Yan Li, Rui Zhang, Saiguang Che and Florin Dobrian, for patiently helping me understand the state of the art of Internet video industry, and allowing me to run experiments on their platforms. I would like to thank colleagues at Microsoft Research, Ganesh Ananthanarayanan, Venkat Padmanabhan, Philip A. Chou, Rajdeep Das, Ranveer Chandra, Robert Grandl, Yinglian Xie, and Yu Fang, as well as

colleagues at Microsoft Skype team, Esbjorn Dominique, Marcin Goliszewski, Dalibor Kukoleca, and Renat Vafin. I would also like to thank Bruce Maggs in Akamai who had provided invaluable feedback on several chapters of the dissertation. This dissertation would not be possible without their earnest support.

I am thankful to the many graduate students at CMU, David Naylor, Matthew Mukerjee, Raja Sambasivan, Seyed K. Fayaz, Tianlong Yu, Soo-Jin Moon, Hyeontaek Lim, Dongsu Han, Yuchen Wu, Carlo Angiuli, Jay-Yoon Lee, Zhuo Chen, Jinliang Wei, Antonis Manousis, Yixin Luo, who had provided invaluable feedbacks to my early drafts and practice talks. I will always cherish the Tuesday seminars and XIA meetings, from which I received more honest comments than from any conferences or workshops. I was also fortunate to work with two extremely talented visitors to CMU, Yi Sun from ICT China, and Shijie Sun from Tsinghua University. Finally, I want to thank Deborah Cavlovich, Toni Fox, and Kathy McNiff for their invaluable help on logistics. My research would not be possible without the financial and intellectual support from the XIA project.

I would also like to thank my undergraduate advisor in Tsinghua, Bin Liu, for accepting me to his group and bringing me to networking research. During my undergraduate years, I was fortunate to work with Yi Tang, Chengchen Hu, Yi Wang, Yan Chen, Mingui Zhang, Beichuan Zhang, and Kai Chen. I learned something from each of them.

I would like to express my earnest gratitude to my parents for their constant love and encouragement, without which none of my achievements would have been possible. Thanks to them for always having faith in me and providing me with the best education possible.

Last but not least, I would like to thank my beloved Xuezhi for her understanding and unreserved support throughout my PhD years.

I am truly grateful to all those who helped me complete this dissertation. If I neglected to mention you, I apologize. Please know that I greatly appreciate your support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today's Internet is an "eyeball economy" driven by applications, such as Internet video streaming (e.g., the share of video traffic of all consumer Internet traffic hit 70% in 2015 and is forecasted to reach 82% of consumer Internet traffic by 2020 [6]) and Internet telephony (e.g., Skype users spend over 2 billion minutes talking to each other every day [27]). As these applications rely on user engagement to generate revenues, it has become of paramount importance that application providers ensure high user-perceived *Quality of Experience* (*QoE*) in order to maintain high user engagement [51, 135]. For instance, recent research shows that even one short video buffering interruption can lead to 39% less time spent watching online videos and cause substantial revenue losses for ad-based video sites. Suboptimal QoE can negatively affect subscription-based service providers as well; e.g., our study with Microsoft Skype shows that most Skype users give low rating to calls when experiencing more than 1.2% packet loss rate [125].

Given the importance of QoE, understanding and improving the QoE of Internet applications have gained increasing attention from both academia and industry. This trend is best illustrated by the recent growth in the number of publications (e.g., [51, 88, 123, 135, 147, 214]), workshops (e.g., [31, 32, 33]), as well as commercial offerings (e.g., [5, 9]) for optimizing the QoE of Internet video streaming, Internet telephony, mobile apps, and web services.

Despite the intense research towards better Internet QoE, measurement studies show that existing approaches have failed to deliver the QoE needed by today's applications. For instance, several studies [122, 135, 147] showed that over 10% of video viewers spent more than 1% of session duration in re-buffering interruptions, which could significantly reduce user engagement, especially for live content [88]. Similar QoE problems are pervasive in Internet telephony as well. Recent measurement on the quality of Skype calls showed that 17% calls experienced over 1.2% packet loss in the call's duration [125], which can cause frustrating user experience [7, 19]. (We elaborate on these quality problems in Chapter 2.)

1.1 Fundamental Limitations of Prior Approaches

These suboptimal QoE issues stem from fundamental limitations of prior approaches. There are two broadly defined classes of prior approaches, whose key difference lies in *where to implement the functionality for optimizing QoE*.

- *In-network* approaches seek to improve QoE by improving the quality of service (QoS) of ISPs and in-network services through better designs of in-network devices (e.g., routers, switches, and middleboxes) and routing schemes. Although in-network approaches have inspired influential projects and enormous intellectual legacy (e.g., [86, 96, 97, 117, 195, 205]), they are fundamentally limited by in-network devices' lack of visibility to user-perceived QoE, and thus it cannot react to QoE problems that are not reflected by lower-level QoS metrics. Moreover, it is difficult, and increasingly so, to make substantial changes to the network core, despite recent efforts to facilitate it [94].
- *Endpoint* approaches seek to improve QoE by using intelligent logic running at individual endpoints to react to changes in network conditions, in order to fully utilize the existing network resources. These approaches are pervasively used in application-level protocols (e.g., [66, 121, 130, 190, 218]) and transport-level protocols (e.g., [89, 107, 116, 217]). Unlike in-network approaches, endpoint-based approaches have direct insight to user-perceived QoE and is arguably more deployable. However, endpoint-based approaches are fundamentally limited by individual endpoints' local visibility to network conditions. As a result, each endpoint can react to the changes in network conditions and resource availability only after the changes have affected the QoE, and when it reacts, it relies on trial-and-error strategies driven by only local information which has limited view on the network conditions. Both aspects lead to suboptimal QoE when network conditions change constantly (e.g., flashcrowds) or when quality of the beginning of a session is critical (e.g., short video clips)

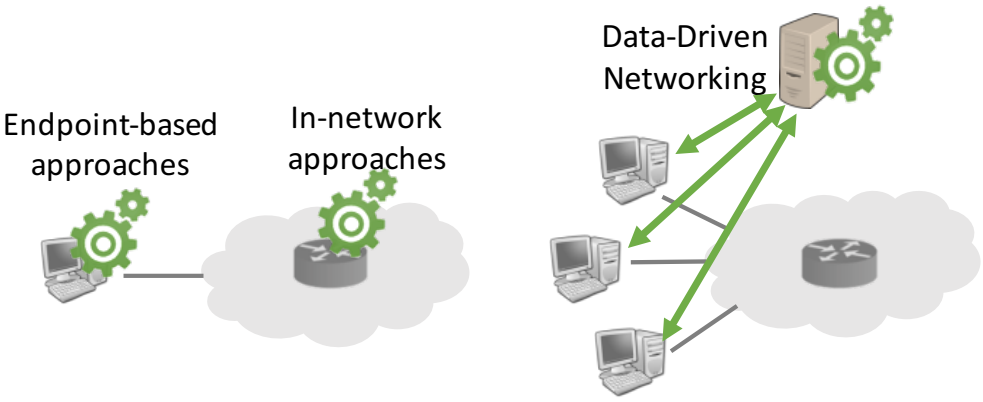
In essence, both approaches do not act on the right signal: they have limited visibility to either *user-perceived QoE* or *network conditions*, both of which are critical to achieving desirable QoE in practice. In contrast, this dissertation approaches the question of where to implement QoE optimization with a radically different answer, and demonstrates that the new approach can get the best of both worlds.

1.2 New Paradigm: Data-Driven Networking

This dissertation is inspired by a recent paradigm shift in computing and tries to bring it to networking research. In essence, *Data-Driven Networking* (*DDN*) offers a different answer to the placement of the functionality of QoE optimization: instead of optimizing QoE at endpoints or in-network devices, we could substantially improve QoE by using a logically centralized controller (as illustrated in Figure 1.1) which maintains a global view of real-time network conditions by gathering QoE measured from many application sessions[1] and uses this global view to make optimal decisions regarding the adaptation of individual sessions [127]. This design choice brings two key advantages of DDN (Table 1.1).

DDN is driven by the right signal: Compared to in-network approaches, DDN can monitor client-side applications and thus can directly optimize user-perceived QoE, rather than indirect low-level metrics. Compared to endpoint-based approaches, DDN compensates the lack of visibility of network conditions at one endpoint by a real-time, global view of QoE observed from

[1] We use "client" to denote where a "session" is actually run.



(a) Classic approaches (in-network approaches and endpoint-based approaches).

(b) Data-Driven Networking (DDN).

Figure 1.1: Contrasting the DDN paradigm with classic approaches. The key distinction lies in where to implement the functionality of QoE optimization (symbolized by the gears): prior approaches implement it in the in-network devices or individual endpoints, whereas DDN implements it in the controller that maintains a real-time global view of QoE of millions of endpoints.

Approaches	**Where data is collected?**	**What data?**	**Who make decisions?**
In-network	In-network devices	Packet-level statistics *no visibility to user-perceived QoE*	In-network devices *not readily deployable*
Endpoint-based	Single endpoint	Performance of a single flow *limited visibility to network conditions*	Endpoints *readily deployable*
Data-Driven Networking	Many endpoints	In-situ QoE of many endpoints *visibility to QoE & network conditions*	Centralized controller *readily deployable*

Table 1.1: Advantages of DDN over in-network approaches and endpoint-based approaches.

many endpoints, thus addressing the key limitation of the endpoint adaptation.

DDN is readily deployable: Unlike in-network approaches or the precursors of data-driven quality optimization (e.g., [193]), DDN is fortuitously aligned with several recent technology trends: Many application providers (e.g., [5, 88, 125, 135]) today have widely deployed client-side instrumentations that can collect real-time in-situ QoE data en masse from clients. The emergence of large-scale data analytics platforms and cloud infrastructure provides the ability to extract insights efficiently from large corpses of data (e.g., [28]) and streams of updates (e.g., [227]). Finally, logically centralized control platforms are commonly employed by many application providers (e.g., [101, 146]) and CDN providers (e.g., [74, 156]).

1.3 Making Data-Driven Networking Practical

While prior work has shown that DDN can potentially improve QoE, it is not clear how to fully realize this potential in practice. *The main contribution of this dissertation is a suite of*

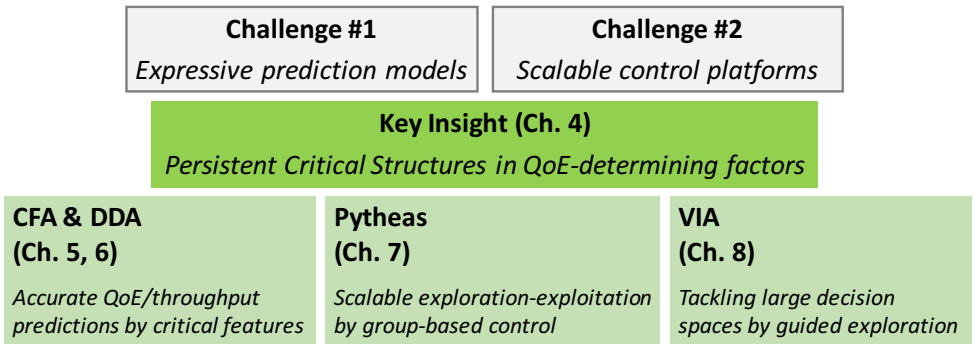


Figure 1.2: The main contribution of this dissertation is to present a suite of solutions (bottom) to address the key challenges of DDN (top). The key insight (middle) is that QoE-determining factors exhibit persistent critical structures.

algorithm and architectural solutions driven by domain-specific insights to make DDN practical for QoE optimization. In particular, we identify key algorithmic and architectural challenges to apply DDN to QoE optimization, address these challenges by novel algorithm and system designs that integrate machine-learning techniques with domain-specific insights, and use real-world deployment and large-scale emulation to demonstrate that our solutions can substantially improve the QoE of video streaming and Internet telephony.

1.3.1 Key Challenges

To unleash DDN's full potential in any Internet-scale application, one must address two fundamental challenges.

1. **Expressive models:** DDN needs to turn QoE measurements of millions of different application sessions into actionable insights. Thus, we need expressive models to capture the complex network-level and application-level factors that affect QoE.
2. **Scalable platforms:** DDN needs to turn the actionable insights into real-time control decisions to be performed by geo-distributed clients. Thus, we need scalable platforms that can respond to geo-distributed clients in real time with decisions based on the most up-to-date insights extracted from information of other sessions.

1.3.2 Unifying Insight

We address these challenges by integrating machine learning techniques with the unifying insight that there are *persistent critical structures* in the relationship between session-level features, decisions, and QoE (as illustrated in Figure 3.1). At a high level, these structures have two distinctive features: (1) they allow us to build expressive models that can identify network sessions with similar QoE-determining factors, and (2) because these structures tend to be persistent, we can build scalable platforms by decoupling offline structure-learning processes and real-time decision making processes. To see an intuitive example of a persistent critical structure, let us consider a group of video sessions bottlenecked by a congested link. The quality of

these sessions may vary over time, but the fact that these video sessions are bottlenecked by the same congested link remains true for the whole duration of the congestion event. In this example, the correlation among the quality of these sessions is a persistent critical structure, which manifests the underlying congestion. (We will give a formal definition of persistent structures in Chapter 3.)

1.3.3 Proposed Solutions

Building on the insight of persistent critical structures, this dissertation develops three concrete solutions to address the two aforementioned challenges in the context of Internet video and Internet telephony. Next, we briefly describe the three components that constitute this dissertation.

- **Expressive QoE Prediction Using Critical Features** (Chapter 5 and 6). Prior work has shown a substantial room for improving video QoE by dynamically selecting the optimal CDN and bitrate for individual video sessions based on a real-time global view of network conditions [147]. To realize this promise, we have developed CFA [126] (a video QoE prediction system that can accurately predict the quality of a video client if it uses certain CDN and bitrate), and DDA [124] (a throughput prediction system to accurately predict end-to-end throughput at the beginning of a video session to help determine the highest-yet-sustainable initial bitrate). Both techniques are inspired by the domain-specific insight of *persistent critical features*, an instantiation of persistent critical structures, that each video session has a small set of critical features that ultimately determines its video quality, and these critical features change much more slowly than video quality, and thus can be practically learned from history data.
- **Group-Based Exploration-Exploitation at Scale** (Chapter 7). While CFA and DDA show promising QoE improvement by formulating the data-driven QoE optimization as a prediction problem, this formulation is necessarily incomplete, as it suffers from a biased visibility and cannot respond to sudden changes. Drawing on a parallel from machine learning, we argue that data-driven QoE optimization should instead be cast as a process of *real-time exploration-exploitation*. To scale the real-time exploration-exploitation process to millions of application sessions running in geo-distributed clients, we have developed a control platform called Pytheas [128], which relies on another illustration of persistent critical structures that the sessions that exhibit similar QoE behaviors have similar network-level features (e.g., IP prefix), and thus their fresh data could be collected by the same geo-distributed front-end cluster close to the clients of these sessions. Inspired by this insight, Pytheas uses a scheme called *group-based exploration-exploitation*, which decomposes the global exploration-exploitation process of all sessions into subprocesses, each managing a group of similar sessions and running in the geo-distributed front-end cluster that has the fresh measurement data of these sessions.
- **Tackling Large Decision Spaces via Guided Exploration** (Chapter 8). The last project tackles a challenge of large decision spaces, which is particularly relevant in Internet telephony. In the first large-scale study on VoIP[2] quality, we found that there is substantial

[2]We use the terms Internet telephony and VoIP interchangeably.

room for improving Skype quality by routing each call through the optimal relay clusters in Microsoft's cloud. However, identifying a close-to-optimal relay for each Skype call in practice is challenging, due to the sheer number of possible relay paths (in hundreds) and their dynamic performance (which could change on timescales of minutes). Neither prediction-based methods (e.g., CFA) nor those based on exploration and exploitation (e.g., Pytheas) would suffice to handle such a large decision space. Our key insight to address this challenge is another manifestation of the persistent critical structures that, for each pair of caller AS and callee AS, there is a *small and stable subset* of relays that almost always contains the best relay path. We have developed VIA [125], a Skype relay selection system that achieves close-to-optimal quality using the concept of *guided exploration*, which, instead of exploring the whole decision space of all possible relay choices, learns a small set of promising relays for each AS pair based on long-term (e.g., daily) historical data, and explores these promising relays using most calls in near real time.

Generalizability beyond Internet video and telephony: Finally, while this dissertation has mostly focused on Internet video and Internet telephony, we observe similar data-driven opportunities in other applications (e.g., CDN overlay routing [156], web proxy selection [146]) where solutions proposed in this dissertation are readily applicable. Besides application-specific QoE, the insight of persistent structures can also help optimize other performance metrics, such as network throughput. For instance, DDA (Chapter 6) shows the feasibility of predicting end-to-end throughput by aggregating information of multiple network flows. These observations indicate the potential *generalizability* of our solutions to realize more data-driven opportunities in a broad set of scenarios in networking and distributed systems.

1.4 Summary of Results

In evaluating the solutions proposed in this dissertation, we focus on answering two questions.

- *How much can QoE be improved by the proposed solutions?* The ultimate goal of DDN is to improve QoE. We examine the contribution of each solution by evaluating the incremental QoE improvement of adding one component at a time. For instance, by predicting the best CDN and bitrate selections based on a global view of network conditions, CFA reduces video re-buffering time on average by 32% compared to a state-of-the-art client-side logic using local information (Chapter 5); and by re-casting the data-driven QoE optimization as a real-time exploration-and-exploitation process, Pytheas further reduces the re-buffering time on average by 30% over CFA (Chapter 7). In Chapter 8, we show that VIA achieves better VoIP QoE than CFA and Pytheas by addressing the new challenge of large decision spaces in Internet telephony. Moreover, this dissertation also tries to identify the circumstances under which the proposed solutions achieve more QoE improvement. For instance, in Chapter 8, we observe improvement of VIA on both international and domestic Skype calls, but international calls have a higher magnitude of improvement than domestic ones.
- *Can the proposed solutions be deployed at Internet scale?* Scalability is another critical metric to evaluate the proposed solutions. Any proposed solution must operate at the scale of a large application provider. In Chapter 5, we show that CFA can update video QoE

prediction every tens of seconds with sub-second response time to the scale of a large video content provider (i.e., 10 million sessions every day), and in Chapter 7, we show that Pytheas throughput scales horizontally with more machines in the controller, and that 30 CloudLab instances can make decisions for the population of a site like YouTube (5 billion sessions per day) with measurement data of concurrent sessions with less than a second of delay.

To demonstrate the benefit of our solutions in realistic settings, our evaluation methodology combines real-world pilot deployment and emulation/simulation driven by large-scale datasets collected from real users. For instance, in Chapter 5, we integrated CFA in a production system [101] that provided video optimization service for major content providers in the US. We deployed CFA on one of these content providers to improve QoE for 150,000 sessions each day. We performed A/B tests (where each algorithm was used on a random subset of clients) to evaluate the improvement of CFA over baseline random decision makers, which many video optimization services use by default (modulo business arrangement like price).

Finally, in order to evaluate application QoE in a reliable and scalable fashion, this dissertation does not use subjective QoE metrics (e.g., user-provided score), but focuses on the metrics that can be objectively measured and known to have great impact on user satisfaction and engagement. For instance, video QoE is measured by buffering time, start-up delay, and average bitrate, each of which has been shown to have strong correlation with user engagement in multiple studies [88, 135]. While subjective metrics can directly reflect user satisfaction, we choose to use these objectively measurable metrics as a proxy for real user satisfaction for two reasons: (1) they can be passively collected en masse by instrumentation code running in client devices without any user input, and (2) they are less noisy than subjective metrics which can be affected by factors (e.g., content or personal preference) beyond the scope of this dissertation.

1.5 Organization

The rest of this dissertation is organized as follows. Chapter 2 begins with the background information of Internet video and Internet telephony, including their QoE problems today and current distribution infrastructures. It then discusses two recent research directions closely related to this dissertation: quality optimization of Internet applications and application of data-driven techniques in networked systems. In particular, it introduces a taxonomy of prior work on quality optimization, which emphasizes the trade-offs between more visibility of QoE feedback and more visibility of network conditions.

Chapter 3 presents an overview of the main insight and ideas of this dissertation. It begins with a formal description of the DDN paradigm, concrete example applications that can benefit from DDN, and a perspective on its advantages over prior approaches. It then elaborates the key technical challenges in making DDN practical. Finally, it describes our key insight of persistent structures in QoE-determining factors, and how the insight inspires our solutions to address the DDN's challenges. Chapter 4 presents a large-scale structural analysis on the video and VoIP QoE problems in the wild. It provides empirical evidence of the persistent structures in QoE-determining factors.

Chapters 5, 6, 7, and 8 describe four important components of this dissertation: (1) CFA and

DDA optimize video streaming quality by predicting video QoE and end-to-end throughput using a global and real-time view of network conditions; (2) Pytheas optimizes quality of Internet-scale applications by re-casting the DDN process as a real-time exploration-exploitation process over millions of geo-distributed clients at scale; and (3) VIA addresses the challenge of large decision spaces, and uses VoIP as a use-case where it optimizes network performance for VoIP calls by selecting the optimal relay clusters.

Chapter 9 summarizes the contributions of the dissertation, discusses the limitations of the proposed solutions, and ends with future work.

Chapter 2

Background

Before we embark on the solutions to improve QoE, it would be helpful to (1) motivate the need for improving QoE by shedding light on today's QoE problems in the wild, and (2) understand the fundamental trade-offs in prior approaches to QoE optimization.

The first part of this chapter uses empirical measurement studies to show that there is a substantial fraction of sessions in both Internet video and Internet telephony with bad QoE (Section 2.1), and then discusses some salient aspects of distribution infrastructures of these applications today (Section 2.2).

The second part of this chapter discusses two research directions, which are closely related to this dissertation: optimization of Internet applications quality and network performance (Section 2.3) and application of data-driven techniques to improve networked systems (Section 2.4). In particular, it presents a taxonomy of prior approaches to QoE optimization, which emphasizes the fundamental trade-off between visibility to user-perceived QoE and visibility to network conditions. The taxonomy also helps to crystallize the contrasts between prior work and this dissertation (Section 3.1.2).

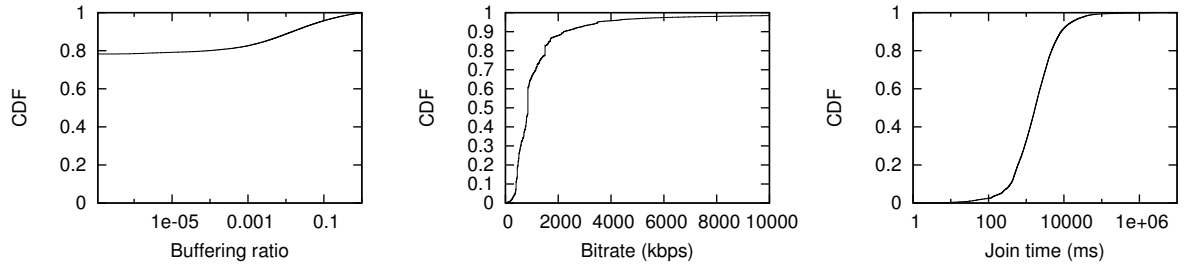
2.1 How Good is QoE Today?

We begin with large-scale measurement studies based on QoE observed by real users to shed light on the how good (or bad) QoE is today for Internet video (Section 2.1.1) and Internet telephony (Section 2.1.2) in the wild. For each application, we first introduce the QoE metrics and the dataset, and then present empirical session-level QoE distributions for different quality metric.

2.1.1 Video QoE

Video QoE metrics: We focus on four key video QoE metrics that are common across different content providers and have been shown to be critical for measuring quality as well as user engagement:

1. *Buffering ratio:* Given a video session of duration T seconds, if the player spent B seconds in buffering (i.e., waiting for the buffer to replenish), the buffering ratio is defined as $\frac{B}{T}$. Prior work has shown that buffering ratio is a key metric that impacts user engagement [88].



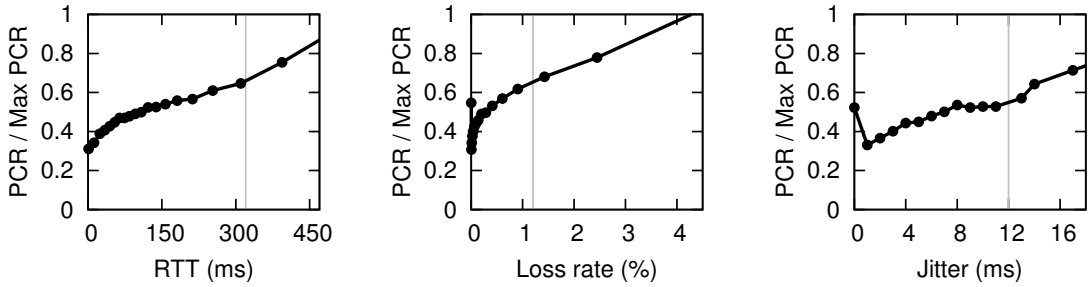
(a) Buffering ratio (b) Bitrate (c) Join time

Figure 2.1: Distributions of observed video QoE metrics – buffering ratio, average bitrate, and join time. We see that a non-trivial number of sessions suffer quality problems. For instance, more than 5% of sessions have a buffering ratio larger than 10%.

2. *Join time:* This is the time taken for the video to start playing from the time the user clicks on the "play" button. While join time may not directly impact the view time of a specific video, it does have long-term effects as it reduces the likelihood of repeated visits [88, 135].
3. *Average bitrate:* Many video players today support adaptive bitrate selection and midstream bitrate switching to adapt to changing bandwidth availability. The average bitrate of a session is the time-weighted average of the bitrates used in a given video session.
4. *Join failures:* Some sessions may not even start playing the video; either the content is not available on the CDN server or the CDN is under overload or other unknown reasons. We mark such a session as a join failure if no content was played during this session.

Dataset: Our dataset is based on client-side measurements of video quality from over 300 million sessions over a duration of two weeks. The unique feature of our dataset is that it is collected over 379 distinct content providers spanning diverse genres, both live and video-on-demand content, different content delivery platforms, different types of bitrate adaptation algorithms, and device/browser platforms. Though US viewers dominate the dataset ($\sim$55%), there are a fair number of European ($\sim$12%) and Chinese ($\sim$8%) users in the dataset. This is especially relevant as it provides us with a panoramic view of state of Internet video delivery today. More details on the datasets can be found in [122].

QoE distributions of different metrics: Figure 2.1 shows the distributions of the first three quality metrics over the dataset. (Join failures are binary events; it is not meaningful to look at a distribution.) The results reconfirm prior observations that there are a non-trivial number of sessions with less-than-ideal quality [88, 147]. The key difference here is that these past efforts only considered a small set of 3–4 content providers. In contrast, we are considering the aggregate data from over 300 content providers, and our results suggest that the QoE problems might be more pervasive than what we realized. For instance, more than 5% of all sessions have a join time greater than 10 seconds; i.e., users had to wait for 10 seconds before the video even started playing! Similarly, more than 5% of sessions had a buffering ratio that was greater than 10%. This is particularly bad as past studies show that even a 1% increase in buffering ratio can lead to 3-4 minutes of lost viewership [88]. Finally, we also see that more than 80% of sessions observe an average bitrate less than 2 Mbps; i.e., less than the lower end of today's "HD" content. Note that the dataset from which these observations are made is dominated by US-based content



(a) PCR vs. RTT (0.97) (b) PCR vs. Loss (0.95) (c) PCR vs. Jitter (0.91)

Figure 2.2: Network performance metrics have considerable impact on VoIP QoE (poor call rate or PCR); y-axis normalized to the maximum PCR. Vertical gray lines show the thresholds for poor network performance. Numbers in the brackets show the correlation coefficients.

providers where viewers in general have good broadband penetrations, so QoE could be even worse in other less developed regions.

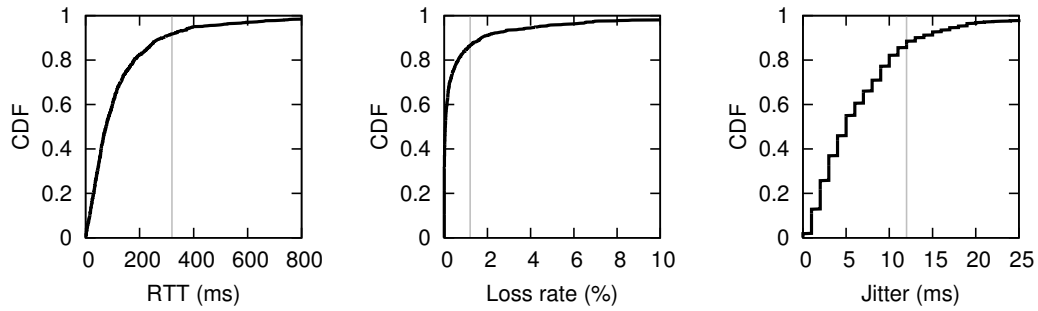
2.1.2 VoIP QoE

VoIP QoE metrics: Like video QoE metrics, VoIP QoE ideally should be based on some user-provided scores, but getting such information directly from users is not scalable. Fortunately, for a small random fraction of calls in Skype, users label the call quality on a discrete 5-point scale, ranging from 1 (worst) to 5 (best). This allows us to study the correlation between these user-provided scores and network metrics that can be objectively measured in a scalable fashion. We can then use these network metrics to study the QoE problems across a large number of users.

Consistent with the operational practice in Skype, we deem the calls with a rating of 1 or 2 as "poor", and use the fraction of such calls, termed as the *Poor Call Rate (PCR)*, as an empirical metric of QoE. Figure 2.2 shows the impact of the three network performance metrics (RTT, loss rate, jitter) on the (normalized) user-derived PCR[1]. For each network metric, we bin calls based on their network performance and show the PCR of the calls within each bin. For statistical significance, each bin has at least 1000 samples. The figures show PCR significantly increases with all the three network metrics (correlation coefficients of 0.97, 0.95, 0.91), confirming that user-perceived quality is indeed sensitive to network performance. Interesting, PCR is sensitive to the *entire* spectrum of network metrics. This suggests that any improvement in RTT, loss or jitter is likely to improve PCR.

Dataset: The dataset from Skype consists of a sampled set of 430 million audio calls drawn from a seven month period. The sampled set includes both calls that use the default path (e.g., BGP-derived) between the caller and the callee as well as calls that are relayed through managed relay nodes distributed across datacenters in different locations. Note that today such relaying

[1]Besides PCR, prior work also has provided analytical models to translate the network metrics into a measure of audio call quality, called the *Mean Opinion Score (MOS)* (e.g., [80]). Our study also showed that MOS is similarly correlated with these network metrics [125].



(a) RTT (b) Loss (c) Jitter

Figure 2.3: Distributions of observed network performance metrics of Skype calls – RTT, loss rate, jitter. Vertical grey lines show the thresholds for poor network performance.

is typically employed for connectivity (e.g., firewall or NAT traversal); i.e., the only instances of relaying in our passively collected dataset correspond to the caller and callee being unable to establish a direct connection. Despite this bias, the dataset offers a *panoramic* view across diverse end-points from 1,905 ASes across 126 countries.

VoIP QoE distribution: Figure 2.3 shows the distribution of network performance experienced by calls using default routes (BGP-based routes). The results show that a significant fraction of calls (over 15%) occur on paths with RTT over 320ms, or loss over 1.2%, or jitter more than 12ms, which we pick as our thresholds for poor performance. These thresholds are in line with literature from industry and standards bodies that recommend one-way end-to-end delay of no more than 150 ms and a packet loss rate of no more than 1% for good call quality [7, 19]. Note that these thresholds are on the *average* values over the call's duration during which there may be transient spikes (e.g., loss burst) in bad performance.

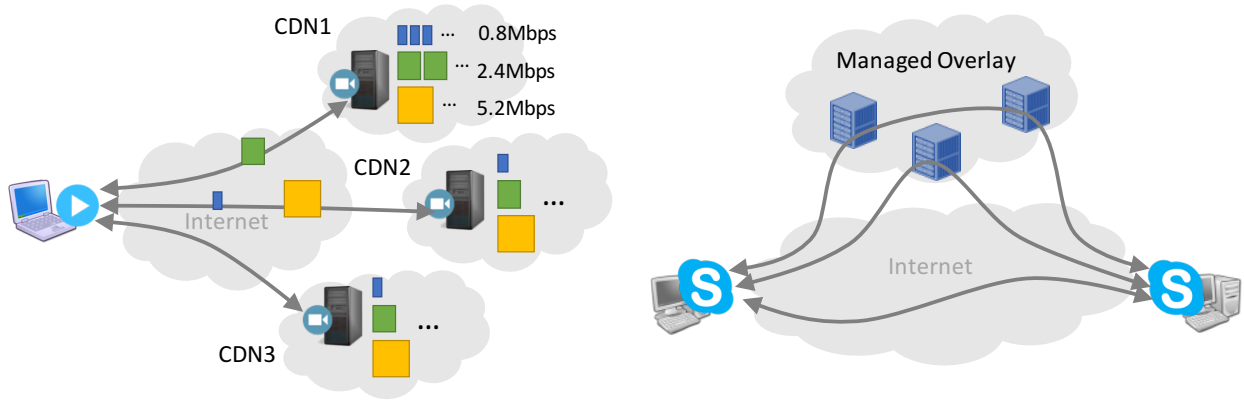
2.2 Today's Application Distribution Infrastructures

Next, we provide the necessary background on Internet video and telephony, focusing on the salient aspects of today's protocols and distribution infrastructures.[2] Specifically, we want to answer two questions: (1) what are the tunable "knobs" in these applications (Section 2.2.3 and 2.2.2)? and (2) how much is the rooms for improving QoE by optimally tuning these "knobs" (Section 2.2.3)?

2.2.1 Internet Video

Early Internet video technologies (e.g., Apple QuickTime [23], Adobe Flash RTMP [25]) were based on connection-oriented video transport protocols, which maintain a session abstraction

[2]This section is not meant for a detailed documentation of their end-to-end delivery systems–readers may refer to related work for a comprehensive overview of these applications' infrastructures; e.g., [183] for Internet videos and [57] for Internet telephony.



(a) Internet video

(b) Internet telephony

Figure 2.4: Today's architecture of Internet video and Internet telephony. Components relevant to this dissertation are highlighted; other details are omitted for clarity. The figures depict the configurations ("control knobs") that can be adaptively tune on a per-session/-call base in order to improve QoE.

between the client and the server, and use (proprietary) stateful control protocols to manage the data delivery. The new generation of Internet video technologies such as Microsoft Smooth-Streaming [2], Apple's HLS [171], and Adobe's HDS [16], however, are HTTP-based adaptive streaming protocols.

In these HTTP-based protocols, each video is typically encoded at multiple bitrates, and is broken into 1-10 seconds chunks stored in multiple CDNs as individual files. When a client streams a video, the player uses the HTTP protocol to fetch the chunks sequentially as individual web files from the server. Figure 2.4a gives a (simplified) depiction of the architecture of how HTTP-based adaptive streaming protocols work in practice. The video chunks are stored in web servers hosted by content delivery networks (CDNs). The video player first receives from the content provider a manifest file (not shown) which enumerates a list of CDNs from which the content can be fetched, as well as a list of available bitrates in which the content has been pre-encoded. Then the player fetches video chunks sequentially, and can switch between bitrates and CDNs at the boundary of any two chunks. Since each chunk is fetched with an independent HTTP GET, there is almost no cost to switch the CDN and bitrate. (Note that the fetches may reuse the same persistent connection if they are from the same CDN.) A video is typically encoded in 3-8 bitrates, and is available from 2-4 CDNs.

Compared with connection-oriented protocols, HTTP-based adaptive streaming protocols enjoy several advantages [121]. (1) The reliance on HTTP provides more ubiquitous reach and support as this traffic can seamlessly traverse enterprise and home NATs and firewalls [167]. (2) The video servers are web servers and caches widely available from commercial CDNs with significantly lower cost than streaming content from dedicated servers that support the early-generation connection-oriented video protocols. (3) Finally, the use of HTTP as the underlying transport protocols allows video streaming to benefit from many techniques proposed recently to enhance web performance and security (e.g., [214]). These practical and performance benefits

have been a key driver for rapid growth of HTTP-based adaptive streaming protocols.

2.2.2 Internet Telephony

Like Internet video, Internet telephony (or audio-video conferencing services) has evolved for more than a decade. The early architecture of Internet telephony relied on peer-to-peer (P2P) overlay systems. This key technology, called UDP hole punching, uses well-connected peer-to-peer users with public IP addresses as *supernodes* to enable connections between clients who did not have public IP addresses or were behind firewall. This technology led to the early success and a dramatic growth of VoIP services.

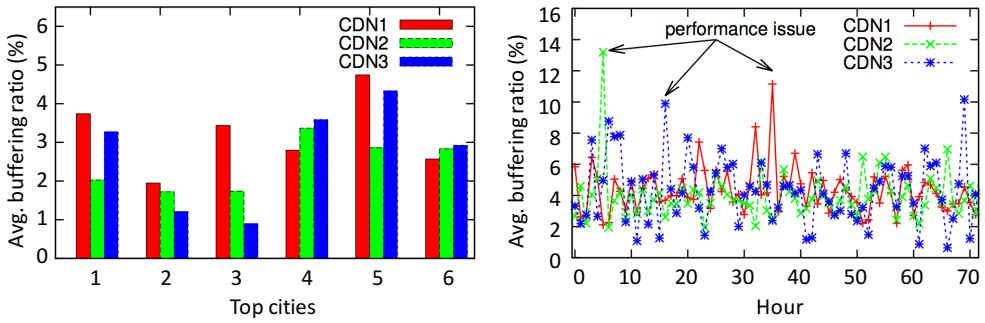
Over the past decade, VoIP services such as Skype and Google Hangouts, have been evolving from the traditional P2P-based overlay towards using *managed overlays* which leverage well-connected and well-provisioned cloud servers. A case in point is Skype, which started off with a peer-to-peer approach to NAT and firewall traversal [133]. In the recent years, Skype has adopted managed overlay [221], with some supernodes hosted in the cloud [1]. It has been reported that Google Hangouts uses relays in the cloud for all calls, and moreover also has streams traverse the cloud backbone from one relay to another [221]. Figure 2.4b depicts a (simplified) architecture of how Internet telephony work over an managed overlay network. Each call can take either the default path through the public Internet or a *relayed path* that routes the traffic through one or more relay nodes in the data centers. Relayed paths could include a single relay to "bounce off" traffic or a pair of relays to enable traffic to "transit through" the private backbone of the managed overlay network.

While both generations of Internet telephony technologies are based on the same technology of UDP hole punching for NAT and firewall traversal, the managed overlay approach enjoys several practical benefits. (1) Global-scale managed overlays use the cloud infrastructure which already exists and need not be built up from scratch, while P2P overlays involved building up overlay networks from scratch, which limited their scale. (2) Supernodes in managed overlays are cloud servers that have high available bandwidth and low latency to edge clients, while in P2P overlays, supernodes were regular clients, and they sometimes became performance bottlenecks due to their limited last-mile bandwidth or computational capacity. (3) In managed overlays, communications between supernodes are through well-provisioned private backbone networks, while in P2P overlays, all communications must compete bandwidth resources of public networks with other traffic.

2.2.3 Room for Improving QoE

While Internet video and Internet telephony services have been evolving towards protocols that have less tunable knobs (e.g., the HTTP-based streaming protocol cannot change bitrate arbitrarily as in traditional protocols, and managed overlays provide less overlay choices than P2P-based ones) for practical considerations, these protocols still offer enough flexibility and recent research has shown a substantial room for improving QoE by optimally selecting the best configuration for each application session.

- *Internet video:* Prior research has shown that video QoE can be significantly improved by better CDN and bitrate configuration for each video session. For instance, Figure 2.5 shows



(a) Spatial diversity

(b) Temporal dynamics

Figure 2.5: Spatial diversity and temporal dynamics of CDN performance [147]. The figures suggest the opportunity of cross-CDN optimization: video buffering ratio can be significantly reduced by dynamically picking the best CDN for each location and at any point of time.

that there is significant spatial diversity and temporal dynamics of CDN performance [147]. Note that most video players today start with a statically configured CDN or a random CDN. This suggests a great opportunity of cross-CDN optimization, e.g., video buffering ratio can be significantly reduced by dynamically picking the best CDN for each location and at any point of time.

- *Internet telephony:* Similarly, it has also been shown that the number of Skype calls whose QoE is negatively affected by network performance can be reduced by over 50% by judiciously selecting supernodes to form a relay path for each Skype call [125]. (We will elaborate on this in Section 8.2.)

These measurement results suggest that application QoE is sensitive to these configurations, and by customizing for each application session with the optimal parameters, we can substantially improve QoE over default or static configurations which are commonly found in today's implementations.

2.3 Prior Work on Quality Optimization

The evolution of the Internet has been driven largely by the need for better quality of a variety of applications. Around early 2000s, many application providers of video streaming, VoIP, and web services started discovering monetization strategies, allowing them to scale with reduced costs. Since this inflection point, Internet applications have been growing and proliferating at an unprecedented pace. Not surprisingly, given the importance of Internet applications, understanding and improving their quality have a long history of intense research. This section introduces a taxonomy of these prior efforts that emphasizes on on their inherent trade-offs.

Before introducing the taxonomy, I would like to clarify that we use the term "Internet quality" to include both quality of service (QoS) and quality of experience (QoE). While QoS is different to QoE, it is closely relevant to this dissertation for two reasons. First, QoS has a strong (albeit non-linear) correlation with QoE. For instance, video buffering highly depends on packet

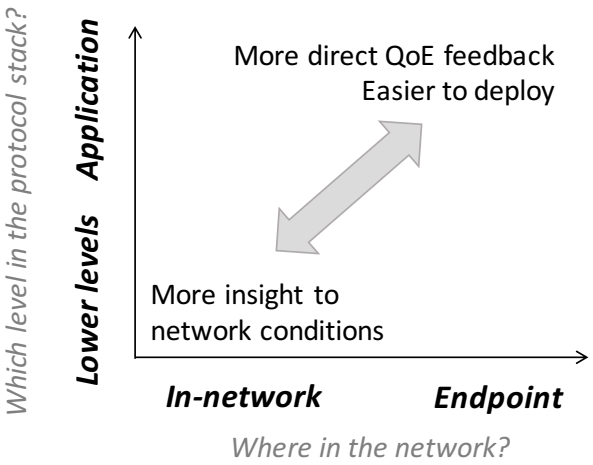


Figure 2.6: Prior approaches can be categorized by their placement of the functionality of quality optimization along two dimensions. These design choices must make fundamental trade-offs between more visibility to QoE and more visibility to network conditions.

loss, and VoIP call experience is very sensitive to network latency. Second, though ultimately we care about QoE, much influential work has focused on providing QoS in the IP and transport layers. Readers may refer to, for example [76], for more discussions on the relationship between QoS and QoE.

Taxonomy: The prior approaches to quality optimization can be categorized based on their answers to a key architectural question: *where should the functionality of quality optimization be implemented?* There are two dimensions in the answers to the questions:

- *Where in the network?* There are two natural options: *endpoint-based* solutions which rely only on endpoints (e.g., clients, servers, caches), and *in-network* solutions which require assists from in-network devices (e.g., switches, routers). To avoid any ambiguity, solutions that involve both endpoints and in-network devices (e.g., router-assist congestion control) fall into in-network solutions under such dichotomy, because they share similar advantages and disadvantages with other in-network solutions. Note that although early in-network solutions only operate in or below the IP layer, this has since changed as in-network devices move up in the protocol stack to provide richer services (e.g., router-assist video streaming [58]).
- *Which level in the protocol stack?* Because our ultimate objective is to improve application-level QoE, it is natural that the solution should have access to the applications themselves, i.e., at the application level. That said, the layering nature of the protocol stack means that any improvement in the lower-level protocols (e.g., TCP, routing, wireless adaptation) may benefit the application-level quality.

Design trade-offs: The choices regarding these two dimensions involve a key architectural trade-off, illustrated in Figure 2.6.

- *More visibility to user-perceived QoE:* On one hand, optimizing quality at endpoints and the application layer has the advantage of more direct and accurate information on user-

perceived QoE, which usually is available only to client-side applications. This allows their optimization logic to be driven directly by the QoE as feedback. Implementing optimization functionality in endpoints at the application layer also carries the practical advantage of more flexibility since their software is upgraded more frequently than in-network devices or lower layer protocols.

- *More visibility to network conditions:* On the other hand, while in-network solutions in the lower layers have to rely on indirect metrics to infer QoE from encrypted application-level communications [41, 52, 108], they enjoy the architectural advantage that they have more accurate and finer-grained information on the network conditions (e.g., per-packet congestion loss). Applications running in user-space sandboxes (e.g., browsers) are often oblivious to changes in these low-level network conditions.

Next, we use the taxonomy to put prior work in perspective.

2.3.1 In-Network Solutions

The in-network approach has long been a focus of intense research. In-network solutions can provide better services at multiple layers on the protocol stack.

IP-layer support for QoS: The two most prominent proposals of QoS services are the Integrated Services (IntServ) and Differentiated Services (DiffServ). IntServ [64] uses the resource reservation protocol (RSVP) [229] to provide the QoS by reserving resources explicitly at all routers along an end-to-end path, and hence all routers must keep states related to services, leading to prohibitive scalability and complexity issues. In contrast, the DiffServ [62] aggregates flows into pre-configured classes based on packet header fields, and hence is more scalable. However, since DiffServ treats packets in the same class identically, it is difficult to provide QoS with strong semantics to individual flows. Another widely used technique is the Multiprotocol Label Switching (MPLS) [176], which reduces the routing complexity and table lookups by packet labeling techniques. Other in-network schemes also seek to provide richer services by adding more functionality to IP layer, such as Multicast [93]. With rapid growth of Internet traffic and applications, these early solutions have become increasingly unfit to cope with the need for QoS with stronger semantics and more scalable implementation over an increasingly ossified infrastructure. These trends have inspired many new in-network solutions over the past decade, including novel QoS service architecture (e.g., Core-Stateless Fair Queueing [195]) and clean-slate solutions (e.g., Active Networks [205], Network OS [105], and Content-Centric Networks [117]). While these efforts have enduring impact on the ensuing research, they impose significant deployment cost to revamp the existing ISP infrastructure. As a result, the Internet today still only provides best-effort service to most applications.

Router-assist congestion control: While congestion control is an end-to-end functionality, many proposals have shown that congestion control may benefit from explicit router assistance, such as providing explicit or implicit feedback on network congestions (e.g., ECN [96], XCP [129], RCP [202], VCP [219]), and Active Queue management (AQM) schemes (e.g., RED [97], AVQ [136], and CoDel [158]). In XCP and RCP, each router along an end-to-end path marks special bits on the packet header to help the senders determine the end-to-end available bandwidth. AQM schemes in contrast aims to prevent persistent queue buildup in routers,

by marking packets with ECN or dropping them before the queue is full, such as in [97]. Similar to IP-layer support for QoS, these methods also add significantly complexity to in-network devices and thus induce significant deployment cost. Moreover, most proposals require making changes on all ISPs along the end-to-end paths, creating more barriers to deployment. A notable exception is in data centers, where router-assist congestion control mechanisms are widely used, because network devices, and end hosts are under the same administrative domain, and are upgraded on a regular basis.

SDN-based approach: SDN is a new network architecture to greatly simplify network management [94] by separating the data plane from the control plane that determines the routing states [152], thus making the data plane "programmable". The SDN-based solutions have many open-source offerings (e.g., [59, 95]), and offer a viable path to the deployment of new routing protocols, as routing table may be updated in near real-time without interrupting any ongoing traffic [131], allowing routing to programmable on a per-flow basis. Moreover, recent research on high-speed SDN-enabled routers has suggested that such programmability can be realized on a per-packet level to implement AQM schemes in a scalable manner [187, 188]. The logically centralized nature of SDN also offers an opportunity that individual ISPs can coordinate with other ISPs (e.g., [181]) to provide end-to-end optimization and even application QoE optimization (e.g., [58]). In today's federated Internet architecture, optimizing end-to-end quality requires multiple ISPs to coordinate, but even if each ISP is willing to adopt SDN paradigm for reducing management costs, it remains unclear whether they have the incentives to coordinate to optimize end-to-end application quality.

2.3.2 Endpoint Solutions

In contrast, the endpoint-based approach relies on endpoints to adapt to changes in network conditions and resource availability.

Overlay routing: Overlay networking has been proposed as an alternative to adding functionality to the network core. It has been applied in a variety of contexts, such as virtual private networks (VPNs) and multicast [55, 79, 164]. Of interest to us here is work focused on overlay routing with a view to improving routing QoS and robustness [47, 180]. This work showed that network performance such as delay, packet loss, and reliability, could be improved by using an overlay path that traverses well-chosen waypoints. Despite this promise, overlay routing for performance gains has not seen much adoption in practice, for several reasons including the last-mile performance bottlenecks when client nodes are used as as peers, and the policy issues involved in turning stub networks (e.g., university campus networks) into de facto transit networks. Perhaps most importantly, these early systems operate on a relatively small scale (e.g., RON [47] had tens of machines), and hence they still lack enough information to maintain an up-to-date view of the dynamic network conditions. It is also unclear whether these systems can scale up to support today's applications running on millions of clients.

Congestion control: Over the past decades, TCP congestion control has received intense research in the literature; from the early efforts towards a general-purpose scheme (e.g., [65, 116, 192]) to later work on specializing congestion control in different operational environments (e.g., data center [45], high bandwidth-delay product networks [107], and satellite connections [67])

and to the latest efforts towards a unifying scheme by using machine-generated code [217] and modeling networks as a blackbox [89]. Current congestion control schemes suffer from a key limitation that they only *react* based on the locally observed information; e.g., without assistance of routers, they will only react after congestion has caused packet loss or latency inflation. A notably exception is congestion manager [54, 193], which aggregates information of multiple TCP connections to predict network performance for a new connection, but these schemes involve changing the kernel or setting up management server, and thus did not see wide adoption. Moreover, state-of-the-art congestion control schemes may mismatch with the goal of applications [103], and even cause bad interaction with control loops in the application layer [114, 121], indicating that application-layer adaptation might be in a better position to meet the need of applications.

Application-level adaptation: To cope with the dynamic network conditions, most applications run custom client-side adaptation logics in the application layer, rather than relying on transport-level or in-network solutions for two reasons: (1) clients is in the best position to detect and respond to QoE problems; and (2) recent work suggests the need for cross-CDN optimizations [147] and flexible web object scheduling [66], which implies the need for keeping minimal state in the network or servers. In video streaming, most commercial products today perform proprietary client-side bitrate adaptation (e.g., [2, 3, 22, 171]) over HTTP-based adaptive streaming protocol [190]. Many studies have identified problems in existing client-adaptation algorithms (e.g., [42, 84]), bad interactions with TCP control loops (e.g., [103, 114]), and techniques to improve bitrate adaptation (e.g., [43, 121, 143]). Other efforts have demonstrated inefficiencies in existing CDN and server selection strategies [37, 145, 147, 209]. In Internet telephony, there has been work to improve client-side rate adaptation through multi-path wireless [130], network performance profiling [85, 218], and better supernode selection [57, 133]). A shared feature of these schemes is that the adaptation is driven by local information observed by a single application session. A notably exception is SPAND [193], which proposed to improve endpoint adaptation by sharing passive measurement from multiple endpoints at the application layer. This dissertation is in part inspired by SPAND and revisits its ideas in the light of the advances in large-scale data analytics and pervasive client-side instrumentations in today's Internet applications.

Network performance prediction: The key issue of endpoint adaptation is that it can only rely on limited, locally observed information to *react* to changes in network conditions. To overcome this limitation, many studies have attempted to predicting network performance, such as latency and available bandwidth, by exploiting stability and stationarity of network performance (e.g., [113]), constancy of various network metrics [53, 230], and longitudinal patterns of cellular performance (e.g., [159]). Researchers have explored three approaches to network performance prediction: (1) to use packet-level probes to estimate end-to-end performance (e.g., [112, 119, 169, 196]), (2) to build an "Internet performance map" based on active probes from a selective set of "vantage points" (e.g., [83, 148, 173, 193]), and (3) to leverage the history of the same client-server pair (e.g., [111, 120, 154, 201, 211]). While this work has shown substantial benefit of accurate performance predictions, it has not seen wide adoption for practical reasons: the packet-level probing requires per-packet information which is invisible to applications, the Internet-map approach needs a dedicated monitoring infrastructure, and the history-based approach requires enough history be accumulated before performance prediction is fea-

sible, but many application sessions, such as web, consist of a few short-lived flows sent from different servers.

2.3.3 Other Related Work

Finally, we discuss other directions closely related to this dissertation.

QoE metrics: A key aspect in video and VoIP delivery is the need to optimize user-perceived QoE. While there is evidence that video viewers are sensitive to frequent bitrate switches (e.g., [82]), sudden changes in bitrate (e.g., [155]), and buffering (e.g., [88]), it is somewhat surprisingly challenging to design a good QoE metric (e.g., [191]), and this is still an active area of research. While recent work also suggests ISPs are inferring user experience from network-layer measurements and QoS metrics (e.g., [41, 52, 108, 149, 184]), we argue that application providers are in a better position than these infrastructure providers to measure and adapt to QoE. The goal of this dissertation is not to offer new QoE models, but to improve the metrics that are known to have high impact on user experience.

Video measurements: There are many recent measurement studies on understanding video content popularity and access patterns (e.g., [70][166]), flash crowds during highly popular events (e.g., [223]), and their implications for CDN and caching designs. While these efforts help motive the techniques proposed in this dissertation, our goal again is to understand the structure of quality problems and further develop solutions to actually improve QoE.

While both in-network solutions and endpoint solutions have to compromise on either visibility to QoE or visibility to network conditions, the solutions proposed in this dissertation strikes a better balance by taking advantage of the access of millions of endpoints with a data-driven approach.

2.4 Prior Work on Data-Driven Optimization in Networking

Another direction closely related to this dissertation is the application of the data-driven paradigm to improving networked systems. We introduce a taxonomy of how networked systems can benefit from the data-driven paradigm. We also use a concrete case study to show how data-driven ideas can be applied to improve TCP congestion control in different ways.

While most systems already use data-driven techniques (e.g., TCP congestion control schemes react against congestion signals such as packet loss and round-trip time), we argue that they could be substantially improved by borrowing systematic frameworks from data science literature and further exploring more data-driven opportunities by leveraging more available data. In particular, we observe two aspects of network systems along which the data-driven paradigm can help: *better setting of parameters* within the existing adaptation logic, and *better run-time decision making logic* to replace the existing one.

Since the way in which networking problems benefit from data-driven techniques bears much resemblance to other systems areas, this section will focus on networking problem, but also draw related work from a broad set of systems areas.

2.4.1 Type I: Better Settings of Parameters

The conventional wisdom has been that in most of network protocols, the adaptation should use *hand-picked* parameters, (e.g., TCP parameters such as initial congestion window size and AIMD parameters). Recent research, however, has shown that performance can be improved by *automatically* tweaking these parameters to achieve better performance in different operating conditions (e.g., better parameters of TCP congestion control [92, 217], AQM logic [144, 185]), configurations of routing algorithms (e.g., [186], and workload modeling in wireless networks [46]). Recent research has also shown similar data-driven opportunities in other areas including CPU cache replacement policy [118], database consistency management [203], and rational database system management [210]. While these efforts are in entirely different areas, they have largely exploited the same inefficacy in traditional control logic that the parameters are statically configured based on assumptions that are ill-matched with the dynamic workload in runtime. Such problem can be fixed by dynamically training these parameters with recent measurements (e.g., [118]) or synthetically simulated data (e.g., [217]). The problem of learning the best configurations for a complex system is conceptually similar to the hyper-parameter selection problem in machine learning [61].

Case study: Tunning congestion control parameters: We use TCP congestion control as a case study to show how a protocol can be improved by tuning the key parameters. Prior work on Remy [217] showed that a machine-designed TCP that uses offline simulation driven by prior knowledge of the network context (e.g., topology, degree of multiplexing) can learn the best values of some key constants and achieve significant performance improvement over today's manually picked values; e.g., a simulated 15 Mbps fixed-rate link with eight senders contending and an RTT of 150 ms, Remy-generated control logic achieves 40+% throughput speed up and 20+% delay reduction over many specially engineered TCP variants.

2.4.2 Type II: Better Run-Time Decisions

While finding better parameters of the existing control logics can improve performance, it is arguably incomplete, since it still uses handcrafted control logics based on analytical model that models the relationship between the feedback signals (e.g., RTT) with internal states of the underlying system (e.g., router queue occupancy). While this approach served us well for decades, it is fundamentally inefficient to characterize today's application delivery systems, which have grown too complex to model analytically. Rather than merely tuning the parameters for the existing control logic, recent research has made a case for a more direct approach in which the logic models the network as a blackbox and use measurement data to drive the decision making. In other word, rather than inferring the underlying network states and finding best decisions analytically, it builds a model that directly associate quality feedback with decisions (configurations). Research has shown that this more direct data-driven approach can improve the performance of routing [181], TCP throughput [89], server selection in web performance [146], as well as bitrate adaptation in video streaming [150]. Besides networking problems, this approach also found its application in improving resource scheduling in cloud services [44]. The techniques commonly used in these solutions are largely based on exploration-exploitation strategies (multi-armed techniques [215]), or Bayesian optimization [163]), in which feedback of network performance is

used to explore the decision spaces. Compared to the first type, the main advantage of the second type is that it could converge to a better decision faster by making minimal assumption about the underlying systems. A primary drawback, however, is that it has less interpretability as the underlying system is modeled as a blackbox.

Case study: Blackbox-based congestion control: Again, we use TCP congestion control as a case study to show how a protocol can be transformed to a simpler logic driven completely by measurement data, while making less assumptions on the underlying network system. PCC [89] attempts to simplified TCP congestion control algorithm by using locally observed performance to directly drive the setting of congestion window size. Specifically, it continuously maintains a model between observed performance (e.g., throughput and latency) and cwnd. This approach greatly simplifies TCP congestion control, while at the same time, significantly improves TCP performance; e.g., $10\times$ higher throughput of TCP CUBIC on global commercial Internet.

2.5 Summary

In the first part of this chapter, we have used empirical studies to show that QoE problems are pervasive in Internet video and Internet telephony. Then we have introduced related background on today's Internet video and Internet telephony services.

In the second part of this chapter, we have introduced a taxonomy to highlight the inherent trade-offs in prior work to improve Internet application quality. In-network solutions have more direct and accurate visibility to network conditions, while endpoint solutions can measure metrics more directly related to user-perceived QoE, and are usually more deployable. As we will see, our solution strikes a better balance between the visibility to user-perceived QoE and the visibility to network conditions through taking a more data-driven approach to endpoint solutions.

In the last section, we have discussed two approaches to using the data-driven paradigm to improve network systems: improving setting of parameters, and improving real-time decision-making logic. While our solution follows the second type, it unleashes more data-driven benefit by taking advantage of more data of real-time measurements collected from millions of endpoints.

Chapter 3

Overview

The main contribution of this dissertation is *a suite of solutions to make data-driven QoE optimization practical* – one can substantially improve QoE by maintaining a global view of up-to-date network conditions based on the QoE information collected from many endpoints. Our solutions achieve this through novel algorithm designs and system implementation that integrate machine learning techniques with domain-specific insights.

This chapter is organized as follows. We begin with an overview of the envisioned architecture of data-driven QoE optimization, called *Data-Driven Networking* or *DDN* (Section 3.1). Then we discuss the algorithmic and architectural challenges of DDN in Section 3.2, motivate the unifying insights behind our solutions in Section 3.3, and finally describe the key ideas of our solutions in Section 3.4.

3.1 Formalizing DDN

We begin with a conceptual overview of the Data-Driven Networking (DDN) paradigm (Section 3.1.1), and then put the DDN approach in perspective of prior work (Section 3.1.2). We end with some illustrative examples of how different applications can benefit from DDN (Section 3.1.3).

3.1.1 Conceptual Architecture

DDN is a new paradigm for designing the adaptation logic of end-to-end protocols (such as adaptive video streaming protocols). Unlike prior endpoint approaches, DDN-based control loop is driven by real-time *multi-session* (not single-session) view of *in-situ quality* [100] measurement (not active measurements or indirect metrics), and *automatically tuned actuation algorithms* based on data-driven insights (with little to no manual tuning).

A DDN-enabled protocol has two additional components: (a) the *client-side instrumentation* code which runs inside client-side application to measure client-perceived quality of each session and applies decisions made by DDN; and (b) the *DDN controller* which runs two loosely coupled steps:

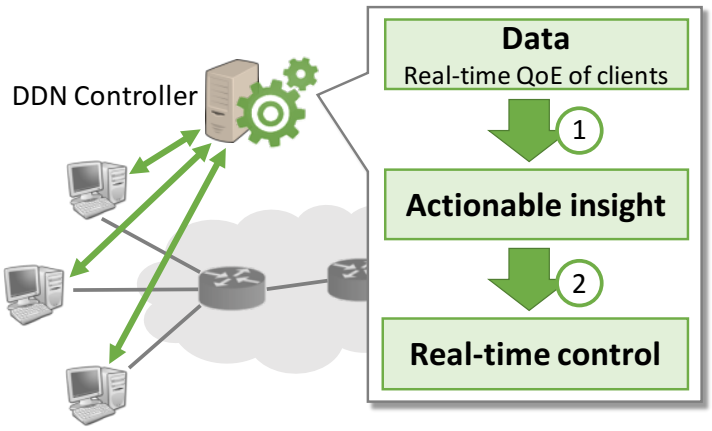


Figure 3.1: Overview of the DDN controller

1. Aggregate quality measurement from client-side instrumentation into a global view of up-to-date network conditions and some actionable insights.
2. Make control decisions based on the actionable insights, and send them to client-side instrumentation for execution.

DDN is aligned with several favorable technology trends, and can be readily deployed in the existing distribution infrastructures of Internet applications. (1) Many application providers today have widely deployed client-side instrumentations that can collect real-time in-situ QoE data en masse from clients (e.g., [5, 88, 125, 135]). (2) Logically centralized control platforms are commonly deployed by many application providers (e.g., content providers [101], web services [146]) and CDN providers (e.g., [74, 156]). (3) The emergence of large-scale data analytics platforms and cloud infrastructure provides the ability to extract insights efficiently from large corpses of data (e.g., [28]) and streams of updates (e.g., [227]).

3.1.2 Contrast to Prior Work

The design choices of DDN bear distinctive features compared to both prior work on quality optimization and other applications of data-driven techniques in networking.

Compared to prior work on quality optimization: We use the taxonomy described in Section 2.3 to contrast the DDN approach to QoE optimization with prior work. Remember that there are two classes of prior solutions: in-network solutions that change in-network devices, and endpoint solutions that rely on endpoint adaptations.

- Unlike prior endpoint solutions which use only single-endpoint information, decision making in DDN is driven by QoE information of multiple endpoints. By expanding the spatial scope of input information from the QoE of one single endpoint to that of multiple endpoints, DDN addressing the endpoint approach's lacking of visibility to network conditions, while retaining its ethos that the decision is driven by user-perceived QoE at the application level. DDN is in spirit similar to seminal work from a decade agao (e.g., SPAND [193]), and our contribution lies in providing end-to-end solutions that combine

the recent advances in large-scale data analytics and domain-specific insights to make DDN practical for today's applications.

- Unlike in-network solutions which rely on indirect signals on quality (e.g., acks or bandwidth), or active probes from a handful of vantage points (e.g., iPlane [148]), DDN relies on in-situ QoE measurement to drive the adaptation; that is, what to be sensed matches what to be optimized. While in-situ quality data may compromise on the fidelity of individual measurement, they are far more efficient than alternatives in obtaining a panoramic and representative view of client-perceived quality from the growingly diverse platforms [100]. Relying solely on in-situ quality data also serves pragmatic purposes as many application providers today already have a vested interest in measuring user-perceived quality for various reasons [26].

Compared to prior work on data-driven optimization in networking: In the taxonomy of Section 2.4, DDN belongs to the type (Section 2.4.2) in which decisions are driven by directly modeling their impact on the metric of interest (i.e., QoE). The key distinction of DDN is that the decisions are driven by real-time data from many different application sessions, rather than a single session as in prior work. This difference has two profound implications. (1) The input data of DDN is much larger both in scale and scope, allowing DDN to learn a more accurately model of the network conditions and make more informed decisions. (2) The input data of DDN is collected from concurrent and history sessions that have different session-level features (client-side, network-level, and server-side), so the DDN decision logic must take into account the potentially complex relationship between these session-level features and QoE.

3.1.3 Illustrative Examples of DDN Benefits

Several early applications of DDN from prior work have shown tremendous promise of this new paradigm.

CDN/bitrate selection for video: The first example shows how a global view of video quality can optimize CDN and bitrate selection for individual video sessions. Video players today have the flexibility of streaming content from one of multiple CDNs and bitrates. However, with only information on a single session, the current protocols always start with a default CDN and fixed (and conservative) bitrate, and gradually converge to a better bitrate and CDN by local trial-and-error strategies. Given both performance of CDNs and client-side bandwidth have a substantial spatial diversity and temporal variability [147], there is a remarkable room for improvement by dynamically mapping a session to the optimal CDN and bitrate with no trial-and-errors. To exploit this opportunity, one can imagine a DDN controller that maps a video session to the CDN and bitrate that has the best quality on similar sessions (e.g., those in the same AS and watching the same video content).

Relay selection for Internet telephony: The second example shows how VoIP quality can be improved by a DDN controller that selects relay servers judiciously. VoIP applications (e.g., Hangout and Skype) use relay servers for NAT traversal, where the selection of relay servers has traditionally been agnostic to real-time network conditions. But recent work has shown a substantial room for improvement on call quality by selecting optimal relay servers for each call [109]. To exploit this opportunity, one can imagine a DDN controller that select near-optimal

relay servers for individual Skype calls by identifying which relay has the best quality for similar calls (e.g., those between the same source and destination ASes on the same date).

Online service cluster selection: The third example shows how the quality of online services (e.g., search engines) can be improved by a centralized control platform, which selects optimal proxies by consolidating quality data of multiple applications and profiles of the infrastructure. Recent work [146] takes the stance of a company who has the visibility and controllability over multiple applications as well as key infrastructure building blocks. By measuring end-to-end quality from clients and dynamically modeling the workload of network paths and servers, it can select proxies that reduce mean latency by 60% and carry 2× more traffic, compared with a baseline that finds proxies by Anycast.

File sharing: Finally, file sharing applications (e.g., Dropbox) have the flexibility to allow each client to fetch files [90] from a chosen server or data center. By using data-driven approaches to predict the throughput between a client and a server [193, 199, 230], we could potentially improve the QoE for these applications.

3.2 Challenges of DDN

Despite its promise, DDN has fundamental challenges that have to be addressed before we can unleash its full potential. The next three sections present our roadmap (depicted in Figure 3.2) towards making DDN practical. We start with describing the high-level algorithmic and architectural challenges, and their manifestations we have seen in different applications.

3.2.1 Need for Expressive Models

The algorithmic objective of DDN is to build a model that maps each session in the session-level feature space to the optimal decision in the decision space. At a high level, the challenges to build such a model stem from the complex relationships between session-level features, decisions, and QoE. To address the challenge, we need an *expressive model* to express this complex relationship using the available measurement data. We have seen two manifestations of the challenge of an expressive model.

- *High-dimensional relationship between session-level features and QoE:* The first illustration is the need to handle the complex relationship, both spatially and temporally, between video QoE and session-level features (Chapter 5). This complex relationship has made it challenging to build an accurate video QoE prediction system, which could help to significantly improve video QoE. In particular, we observe a combinational effect where video QoE is affect by a specific combination of feature values, but does not appear to be correlated with any individual feature. We also observe that QoE of different sessions may be affected with different feature combinations. In addition to these spatial patterns, these QoE-determining factors (as well as QoE itself) may change over time on timescales of several minutes. Therefore, an accurate QoE prediction model must be expressive enough to capture all these spatial and temporal complexities.

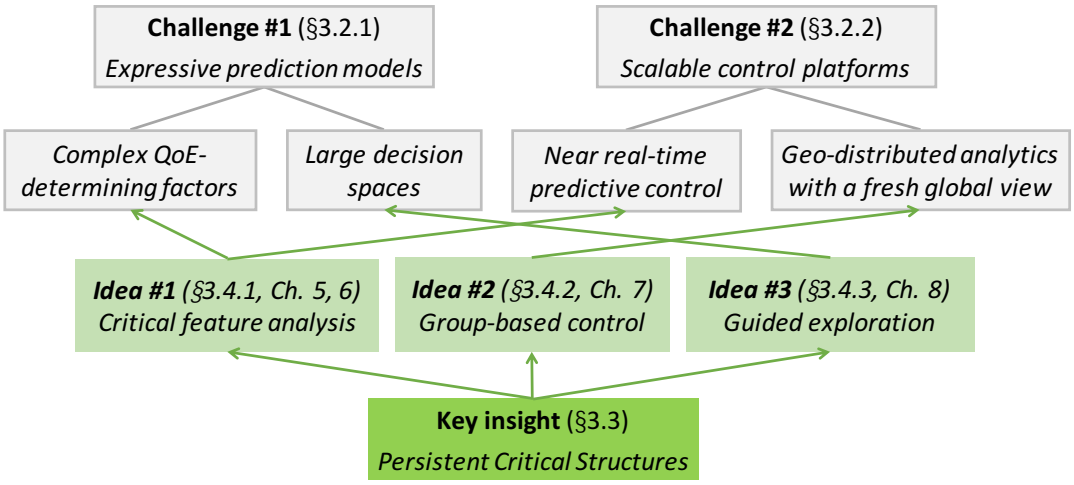


Figure 3.2: The technical roadmap of this dissertation towards making DDN practical. We present three ideas to address the four manifestations of the high-level challenges of expressive prediction models and scalable control platforms. The key enabling insight behinds our ideas is the persistent critical structures of QoE-determining factors.

- *Large decision spaces:* The second illustration is the need to handle large decision spaces, especially in Internet telephony, where one needs to find a good relay path for each VoIP call in a set of hundreds of relay points. In addition, the performance of these relay paths could change on timescales of minutes. Simply using geo-distance to reduce the decision spaces is suboptimal because low geo-distance to end users do not necessarily mean low latency and low packet loss rate which have weak if any correlation with the geo-distance. Moreover, the best choice of relays depends on locations of both caller and callee. All these suggest a need to reduce the size of the mapping between the client space and the decision space.

3.2.2 Need for Scalable Platforms

The system design of DDN should meet the following requirements: the DDN controller has to make control decisions in near real time based on fresh data from many other geo-distributed clients, and serve the decisions to clients within low response time. The key architectural challenge is how to strike a balance between three seemingly conflicting objectives: (1) *data freshness*, (2) *responsiveness to geo-distributed clients*, and (3) *global view*. We have seen two manifestations of the challenge of a scalable platform.

- *Global view vs. data freshness:* A practical issue of running the DDN controller in the existing control platforms of application providers is that it is not clear how to maintain a fresh, global view of measurement data from all sessions. These control platforms typically consist of multiple geo-distributed frontend clusters and a centralized backend cluster. Each session uploads its quality measurement to a nearby frontend, which then updates the backend every tens of minutes to hours. While this design makes much sense for real-time

per-session quality monitoring and offline analytics, none of which requires fresh, global data of all sessions, it is ill-suited to run the DDN control logic in either the geo-distributed frontends (without global view) or the centralized backend (without fresh data).

- *Near real-time predictive analytics:* Even if we can gather real-time data to the same data center, it is still very challenging to run real-time analytics to predict the optimal decision in near real time, e.g., on a timescale of tens of seconds. As mentioned in Section 3.2.1, we need a large amount of data to update a high-dimensional model between session-level feature space and video QoE in near real-time (on timescales of tens of seconds). Our evaluation based on standard large-scale analytics platforms show that given the sheer volume of measurement data, it would take tens of minutes, a magnitude longer than needed, to update the model.

3.3 Key Insight: Persistent Critical Structures of QoE-Determining Factors

Our solutions to address these challenges integrate standard ML algorithms and systems with a key domain specific insight that Internet applications have *persistent critical structures that help identify network sessions with similar QoE-determining factors, and that such structure tends to be persistent on timescales of at least tens of minutes*. We now give the formal definition of persistent critical structures and how they intuitively help address DDN's challenges.

Formal definition: Let us first formally describe DDN as follows. In essence, DDN is as a decision-making function $F : 2^{\mathbb{S}} \times 2^{\mathbb{D}} \times \mathbb{S} \times \mathbb{R} \mapsto \mathbb{D}$, which takes as input a set of historical sessions $S \in 2^{\mathbb{S}}$ whose QoE is already measured, a set of available decisions $D \in 2^{\mathbb{D}}$, a new session $s \in \mathbb{S}$, and s's timestamp $t \in \mathbb{R}$, and outputs a decision $d \in \mathbb{D}$ for session s. Now, a *structure* is formally defined as a function $P : 2^{\mathbb{S}} \times 2^{\mathbb{D}} \times \mathbb{S} \times \mathbb{R} \mapsto 2^{\mathbb{S}} \times 2^{\mathbb{D}}$, which takes as input a set of historical sessions $S \in 2^{\mathbb{S}}$, a set of decisions $D \in 2^{\mathbb{D}}$, and a session $s \in \mathbb{S}$, and the timestamp $t \in \mathbb{R}$, and outputs a pair of subset of history sessions $S' \subset S$ and a subset of decisions $D' \subset D$.

Key properties: Persistent critical structures are a type of structures that have two following properties:

- *Criticality:* These structures identify (often small) subsets of history sessions and decisions which are more critical than others history sessions or decisions in determining QoE; i.e., $F(S,D,s,t) = F(S',D',s,t)$, where $(S',D') = P(S,D,s,t)$. This essentially means the DDN control logic can make the same decisions by only looking at the subset of relevant history sessions and decisions.
- *Persistence:* These structures tend to persist on timescales of tens of minutes. This means in a time window Δ of tens of minutes, the function P is likely to find the same relevant history sessions and decisions, i.e., $P(S,D,s,t) = P(S,D,s,t+\Delta)$.

Illustrative examples: These persistent critical structures of QoE-determining factors can manifest themselves in many forms. For instance, if the QoE of a video session depends on the server load (i.e., some decision-specific properties) and client-side ASN (i.e., some session-level features), and such dependency lasts for tens of minutes, then it would be possible to identify the

best decision for the session by looking at history sessions having in the same AS (instead of all history sessions), and only consider server with low load (instead of all possible decisions).

Intuitively explanation of persistent critical structures: The intuitive explanation of these persistent critical structures is that in networked systems and application delivery systems, performance bottlenecks are often persistent, and the persistent critical structures can be viewed as "manifestations" of these bottlenecks in the space of session-level features and decision-specific properties–a session's QoE only depend on history sessions and decisions experiencing the same bottleneck. Such persistent bottlenecks can be found in many prior studies in the context video streaming [126], web service [146], end-to-end network performance [230]. They can also be viewed as a generalization of the persistent network congestions, which can be mathematically explained using queueing theory (Chapter 6 of [132]). In Chapter 4, we will show evidence of these persistent structures through an empirical structural analysis on QoE problems based on real datasets.

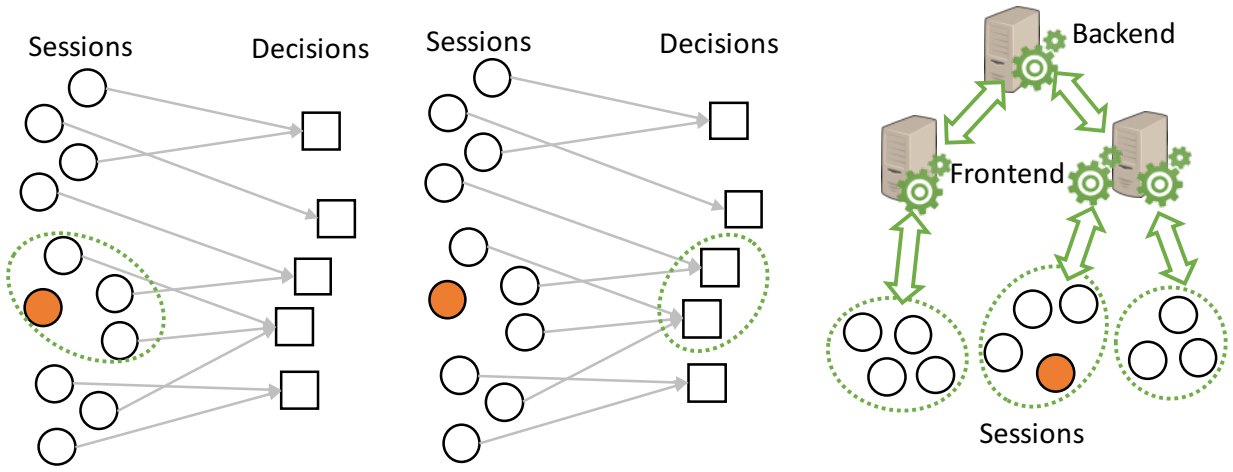
3.3.1 How Intuitively Persistent Critical Structures Address the Challenges?

Next, we intuitively illustrate how the persistent critical structures help to address challenges of DDN (Figure 3.3). In the next four sections, we will see more concrete ideas that are based on the insight of persistent critical structures, and how they address challenges in the context of Internet video and Internet telephony.

Reducing session-level feature spaces: One implication of the criticality of these domain-specific structures is that the complex relationship between session-level features and QoE (an manifestation of the challenge of expressive models in Section 3.2.1) can be expressed by low dimensional models that can be maintained with limited available data. To see this idea in action, let us consider the example of Figure 3.3a, in which we want to make the decision for a new session (the orange circle) based on the QoE measured by the history sessions (the white circles). It is suboptimal to look at only the sessions that match values on all features (e.g., IP prefix, location, device, content, network path, etc) with the new session, because we will end up with too few matches, and therefore the decisions will not be reliable. Nor is it scalable to use all history sessions as the input to make the decision in real time. The advantage of persistent critical structures is that each session's QoE only depends on by a few critical features (rather than all features). Therefore, we can find sufficient amount of similar sessions by matching along the most relevant features. At a high level, this idea resembles the ML techniques that leverage the "locality" in data to tackle curse of dimensionality [61].

Reducing large decision spaces: Another implication of the structures' criticality property is that the large decision spaces (another manifestation of the challenge of expressive models in Section 3.2.1) can be reduced to a subset of most promising decisions which can be explore efficiently by concurrent application sessions who share the same characteristics. Figure 3.3b illustrates an example of this idea, where instead of exploring all decisions, it would be more efficient to focus on a subset of decisions that are most likely to be optimal.

Decomposing the decision-making process: In the context of network applications, the persistent critical structures often correlate with network locality (e.g., clients in the same IP prefix). This observation allows us to strike a balance between global view and data freshness (Sec-



(a) Reducing session-level feature spaces.

(b) Reducing large decision spaces.

(c) Decomposing the decision-making process.

Figure 3.3: Illustrations of how persistent critical structures help to address challenges of DDN. (Each application session (depicted as a circle) on the left is mapped to one of the available decisions (depicted as boxes) on the right.)

tion 3.2.2) with an implementation that is amenable to the infrastructure of geo-distribution frontend clusters. At a high level, the idea (illustrated in Figure 3.3c) is to decompose all sessions into *groups* of similar sessions who share the persistent critical structure as well as the network locality. Since the sessions in the same group share network locality, their QoE measurement will be sent to the same nearby frontend cluster. To make decisions for the sessions in a group, we can use a logic that runs in the same frontend cluster where their fresh data is collected, and that uses only the information of these similar sessions to make decisions. In this way, the decisions are effectively equivalent to using a fresh global view, because they are made with fresh data of the most relevant sessions.

Learning the persistent critical structures from data: While the criticality of the persistent critical structures serves as the key to making DDN practical, it is unclear how to obtain these structures in the first place. The key to learning these structures lies in their persistence. Since these structures tend to persist on long timescales, we can learn these structures from massive data in time windows of long timescales and use an offline process that is separate from real-time decision making. Figure 3.3c illustrates the idea: we can run an algorithm to discover the persistent critical structures in the backend cluster where the measurement data of all sessions are collected. And although the data received by the backend cluster is slightly stale, it is still sufficient to learn the slow-changing persistent critical structures.

3.4 Making DDN Practical by Persistent Critical Structures

The insight of persistent critical structures enables three key ideas (Figure 3.2) to address DDN's challenges in the context of Internet video streaming and Internet telephony.

3.4.1 Critical Features Analysis

Prior work has shown that video QoE can be improved by a prediction system that accurately predicts the QoE of a video session, if it uses a certain CDN and bitrate. The challenge is that this prediction system must be (a) expressive enough to capture complex relations between video quality and observed session features, and (b) capable of updating quality predictions in near real time. We have tried several off-the-shelf machine learning techniques, such as random forests and SVM, but found they did not produce expected QoE improvements, because the long-term historical data is too coarse-grained for these algorithms to capture the dynamics of video quality, while the short-term historical data is not sufficient for the algorithms to learn complex relations between video quality and observed session features.

Our solution leverages an instantiation of persistent critical structures in video streaming, called *persistent critical features: each video session has a small set of critical features that ultimately determines its video quality, and these critical features change much more slowly than video quality*. Let us consider a concrete example of such persistent critical features from [126]. In a real-world incident, video sessions of Comcast users in Baltimore who watched videos from Level3 CDN experienced high failure rate (VSF) for several hours. The reason turned out to be the overloaded local cluster serving Comcast users in that area, which can be characterized by three critical features: CDN ("Level3"), ASN ("Comcast") and City ("Baltimore"), and the correlation between the combination of these feature values and high VSF persist for the whole duration of this incident, even though the QoE has fluctuated a lot during this period.

The insight of persistent critical features has inspired a prediction model that captures complex QoE-determining factors and is amenable to scalable implementation. Given a video session under prediction, the model identifies many similar sessions from a short-term history by matching only on its critical features, thus capturing complex QoE determining factors while avoiding curse of dimensionality. The persistence of these critical features also naturally enables decoupled implementation: we can learn these critical features from long-term historical data and update the models by short-term historical data in near real time to capturing quality fluctuation.

3.4.2 Group-Based Control

While the predictive decision-making algorithm described above shows promising QoE improvement, it faces has two fundamental limitations. (a) Casting the data-driven QoE optimization as a prediction problem suffers from the many known biases such as incomplete visibility. (b) The prediction algorithm is not a geo-distributed one, so it requires a fresh and global view be maintained in one cluster. Having a fresh and global view physically in the same cluster is, however, impractical because in many control platforms, measurement data are first collected in several geo-distributed frontend clusters each having a partial view of nearby clients, and then periodically archived in a backend cluster to form a global though slightly staled view. A baseline approach is to run control logic in a single backend cluster with global data from frontend clusters, but this approach leads to non-trivial staleness of the global data and suboptimal decisions.

To overcome these limitations, we re-cast the data-driven QoE optimization as a real-time exploration and exploitation process, and build a practical system to run it among all clients at scale. Our key insight is inspired by the criticality of persistent critical structures – *the clients that*

exhibit similar QoE behavior will have similar network-level features (e.g., same IP prefix), and thus their fresh data will likely be collected by the same frontend cluster. We see manifestations of this insight in many settings. For instance, video sessions with similar QoE from the same CDN/server tend to match on client IP prefix [126, 199]. Similarly, VoIP calls between the same ASes are likely to share the best relays [125], and clients from same /24 IP prefix will have similar web load time from the same edge proxy [146].

This insight inspires the notion of *group-based control*, which enables the real-time global exploration-exploitation process by decomposing the process into subprocesses, each controlling a group of clients with similar context by network locality and other key session-level features and running in the frontend cluster that has these clients' fresh data. Since sessions within a group share network locality (e.g., in the same locations and IP prefixes), they are likely to be mapped to the same frontend cluster. By running the per-group exploration-exploitation logic in this frontend cluster, we can update decisions with fresh data from other sessions i n the group received by this frontend cluster.

3.4.3 Guided Exploration

To make data-driven QoE optimization practical in Internet telephony, we have to address an additional challenge that it has a large decision space of relay choices, so there are usually not enough VoIP calls to reliably estimate the dynamic network performance between each AS pair.

The key insight to address this challenge is another illustration of persistent critical structures – *the stability of promising relay choices: for each pair of caller AS and callee AS, there is a small and stable subset of relays that almost always contains the best relay*. This insight has two implications: (1) because this subset of relays is stable, it can be learned from history; and (2) because this subset has only a few relays (less than five), it can be explored efficiently even with limited data. Inspired by this insight, we develop a relay selection system that achieved close-to-optimal quality using the concept of *guided exploration*. The idea is to learn a small set of promising relays for each AS pair based on long-term (e.g., daily) historical data, and explore these relays using most calls in real time.

3.5 Summary

In this section, we first discussed the advantages of DDN over prior approaches. As an application-level endpoint approach, DDN enjoys the advantage of direct access to user-perceived QoE, and at the same time, compensates the limited insight to network conditions by consolidating real-time measurement from many endpoints, thus achieving the best world of both endpoint solutions and in-network solutions.

We have also presented an overview of our solutions to address DDN's technical challenges– the need for expressive models and scalable systems. Our key insight is that there are persistent critical structures in the relationship between session-level features, decisions, and QoE. This insight has led to three concrete ideas, which we will present in the next chapters: (1) critical feature analysis to enable an expressive QoE prediction model, (2) group-based control to enable

real-time exploration-exploitation process at scale, and (3) guided exploration to handle the large decision spaces, particularly in Internet telephony.

Chapter 4

Structural Analysis of QoE Problems

We have seen that DDN is a promising alternative to overcoming the fundamental limitations in prior work of QoE optimization, and the key insight to make DDN practical is the *persistent critical structures in the QoE-determining factors*. In this chapter, we present empirical evidence of these persistent critical structures by using a large-scale structural analysis on the video and VoIP QoE. In particular, we shed light on the *temporal persistence* and *spatial criticality* of these structures of QoE-determining factors in the wild.

- Spatial patterns: Are the QoE problems uniformly spread through the space of feature combinations or are there specific combinations that have a higher concentration?
- Temporal patterns: Is each QoE problem a transient for a specific ISP, CDN, or provider (or combination of these) or are these problems persistent over long periods?

We also show that there are substantial correlations among the QoE problems of different metrics, suggesting that it is possible to improve multiple QoE metrics simultaneously.

This chapter is organized as follows. Section 4.1 presents the structural analysis on video QoE, Section 4.2 presents a similar analysis on VoIP QoE, and Section 4.3 summarizes the chapter with key observations.

4.1 Internet Video

In this section, we begin by describing our dataset and the methodology of analyzing spatial and temporal patterns of video QoE problems (Section 4.1.1), then present our results in Section 4.1.3, 4.1.2, and 4.1.4, and finally summarizes the observations in Section 4.1.5.

4.1.1 Methodology

Dataset: We use the same dataset as described in Section 2.1.1. The dataset consists of client-side measurements of video quality from over 300 million sessions of 379 distinct content providers spanning diverse genres, both live and video-on-demand content, different content delivery platforms, different types of bitrate adaptation algorithms, and device/browser platforms.

Session-level features: The basic unit in our dataset is a video session. A session represents

a user viewing a video on one of our affiliates' sites for some duration of time. Each session is associated with a set of *seven features*:

1. *ASN:* The Autonomous System Number (ASN) that the client IP belongs to. Note that a single ISP (e.g., Comcast) may own different ASNs both for management and business reasons. We focus on the ASN as it is more fine-grained than the ISP granularity. We observe in aggregate 15K unique ASNs spanning multiple countries.
2. *CDN:* In total, we observe 19 unique CDNs spanning popular CDN providers as well as several in-house and ISP-run CDNs. (Some providers use proprietary CDN switching logic; in this case we pick the segment of the session with the CDN used for the longest duration.)
3. *Content provider (Site):* This is the specific provider from which the client requested some content. We have 379 content providers that span different genres of content. We use the terms site and content provide interchangeably.
4. *VoD or Live:* Video content falls in one of two categories: video-on-demand (VoD) or Live. We use a binary indicator to see if the particular content was a Live event or a VoD video.
5. *Player type:* We see diverse players such as Flash, Silverlight, and HTML5.
6. *Browser:* We see diverse client browsers including Chrome, Firefox, MSIE, and Safari.
7. *Connection type:* Finally, we have the type of access network connection such as mobile/fixed wireless, DSL, fiber-to-home. These annotations come from third party services [24].

Identifying problem sessions: Our focus is on understanding quality problems as they appear in the wild. To this end, we identify problem sessions w.r.t. each of the quality metrics. Note that a given session may appear as a problem session on a subset of metrics; i.e., it might have a low join time but may have a high buffering ratio or vice versa. We consider the metrics separately to avoid implicitly assuming that the metrics or failures are correlated.

- For join failures, we use a binary indicator if the session failed or not. For the remaining metrics, we choose specific thresholds based on domain-specific knowledge and observations in prior work[1]. Our specific thresholds and rationale are follows.
- For buffering ratio, we identify a problem session if the value is greater than 5%; this is based on the observation that beyond this value there is a sharp decrease in amount of video viewed [88].
- For bitrate, we mark a problem session if the average bitrate is less than 700kbps; this value roughly corresponds to the recommended "360p" setting on video providers. We use a fixed threshold of bitrate in this work for simplicity, but we do acknowledge that bitrate settings are content-dependent (e.g., some contents do not provide high resolution streams).
- Third, we mark all sessions with a join time greater than 10 seconds; this represents a conservative upper bound on the tolerance of users [63, 135].

Identifying problem clusters: We begin by dividing our dataset into discrete one hour epochs.[2]

[1] We do acknowledge that there is no ideal choice of threshold and it is likely that these thresholds will evolve as user expectations and network conditions improve. As such, the choice of thresholds is illustrative of the structure of video quality problems that occur today. The methodology and qualitative observations we present are not tied to the specific thresholds. We have confirmed that the results are qualitatively similar for other choices of these thresholds as well.

[2] One hour is the finest granularity of the dataset and thus we cannot analyze effects at smaller timescales.

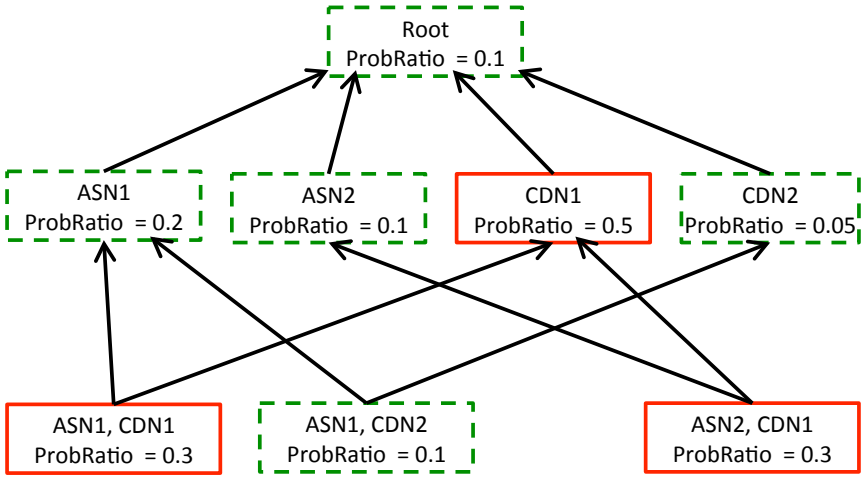


Figure 4.1: Representing the relationship between clusters using a DAG. Red boxes represent the problem clusters.

As a first step to analyze the structure, we *cluster*[3] together sessions that share one or more client/session features within the same epoch. For instance, the cluster "ASN=*ASN1*" describes all sessions where the user belongs to *ASN1* and the cluster "ASN=*ASN1*, CDN=*CDN1*", describes all sessions where the user belongs to *ASN1* and the session was assigned to a server from *CDN1*. In order for our observations to be reliable, we want to focus on clusters that are deemed to be statistically significant sources of problem sessions. To this end, we define the problem ratio of a cluster as the ratio $\frac{\text{\# of problem sessions}}{\text{\# of sessions}}$. Then, we cull out the clusters whose problem ratio is significantly higher than the global average problem ratio. We also remove all clusters that have a small number of sessions in aggregate; i.e., problems observed within a small cluster may not be statistically significant. Combining these two steps, we define a problem cluster as a cluster that has a problem ratio $\geq 1.5\times$ the global problem ratio,[4] and the number of sessions in this cluster is ≥ 1000. In the rest of this paper, we start from the problem clusters as our basis and subsequently refine the analysis.

While the grouping of problem sessions into problem clusters provides some insights into the structure of problems, there is still one key missing aspect. Specifically, we may have different granularities of problem clusters that may be intrinsically related to the same underlying root cause. Thus, our next step is to refine these problem clusters to identify such potential causal structures across the problem sessions.

The set of all clusters can be viewed as a *hierarchical* structure across the space of client/session features with natural parent-child relationships. We can visualize these parent-child relationships as a DAG as shown in Figure 4.1. A cluster *C1* is a parent of cluster *C2*, if the set of features defining the cluster *C1* is a strict subset of that of *C2*. For instance, the cluster "*ASN1*" is a parent of the more specific clusters "*ASN1*, *CDN1*" and " *ASN1*, *CDN2*".

[3]The term "cluster" represents a group of sessions that share common features, and it is indeed different from traditional clustering algorithms where a cluster can be a group of any data points.

[4]This value roughly represents two standard deviations away from the mean of the per-cluster problem ratio distribution.

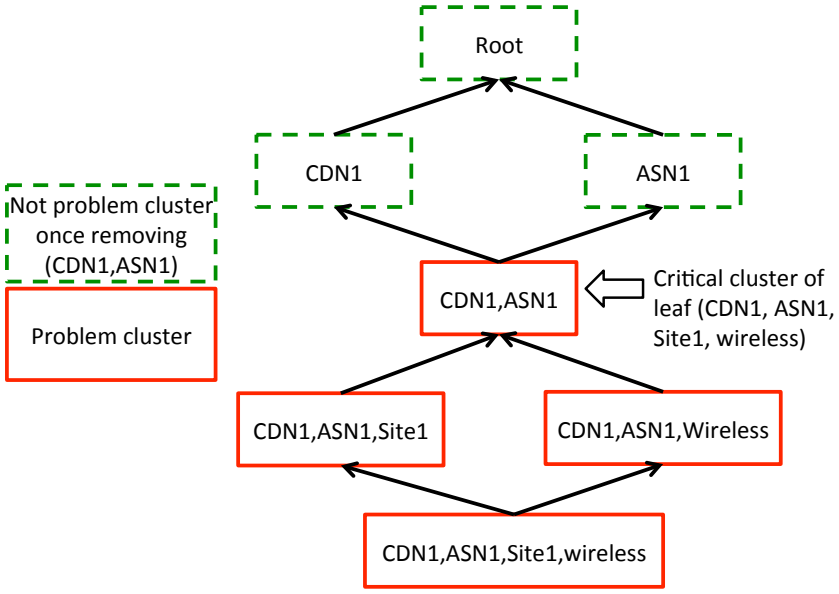


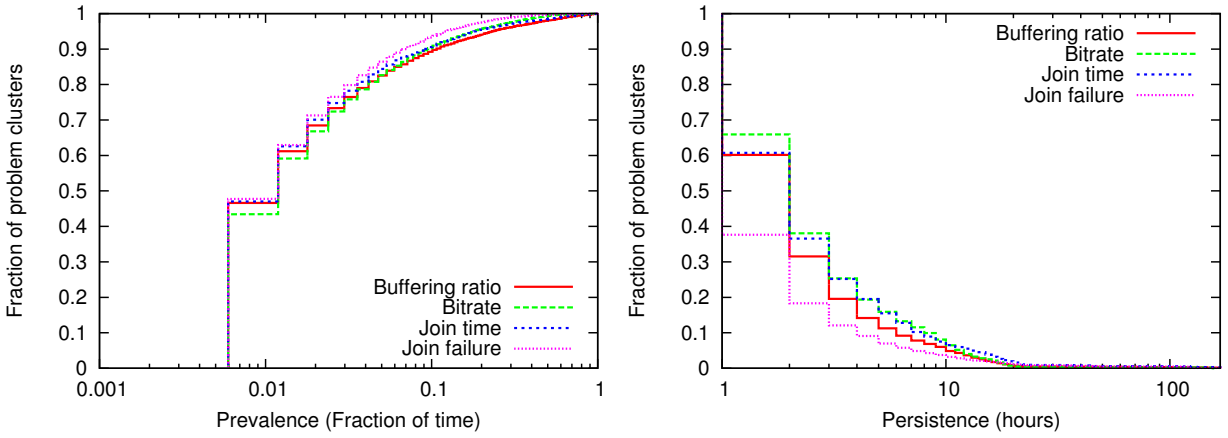
Figure 4.2: An illustration of the phase transition idea for identifying a critical cluster.

Identifying critical clusters: Our goal is to identify a small number of *critical clusters* that can potentially explain the occurrences of different problem clusters. In our example in Figure 4.1, intuitively we should pick the "*CDN1*" cluster rather than pick "*ASN1*, *CDN1*" and "*ASN2*, *CDN2*" clusters separately. Given that we do not have ground truth, critical clusters can serve as starting points for further investigation. An intuitive criterion for identifying a critical cluster is analogous to the notion of the minimum description length (or Occam's razor) from the machine learning literature. Conceptually, we should pick the most compact description to explain an observation. Building on the above intuition, we can identify a critical cluster as consisting of the minimal set of features that when combined together can lead to significantly high problem ratio in its cluster (e.g., a problem cluster) and removing even one feature from this set will reduce the problem ratio. To this end, we identify critical clusters using a *phase transition* algorithm as follows. For each session, we construct all logical paths in the DAG from the root to the leaf. Then, for each of these paths, we identify the point closest to the root along this path such that every cluster that is a descendant is a problem cluster and once removing it every cluster that is an ancestor is not a problem cluster. We use Figure 4.2 to explain the intuition. In this figure, "*CDN1*, *ASN1*" is the critical cluster—every cluster that is a child of this combination is a problem cluster and if we remove the sessions in this combination, the parents "*CDN1*" and "*ASN1*" cease to be problem clusters. That is, this combination of features represents a key "transition point" in this hierarchy between problem clusters and non-problem clusters.

4.1.2 Temporal Patterns

We begin by analyzing the temporal *prevalence* and *persistence* of the problem clusters.

Prevalence of problem cluster: We define the *prevalence* of a problem cluster as the fraction of the total number of epochs in which this cluster appears as a problem cluster. Figure 4.3a shows



(a) Distribution of the prevalence of problem clusters

(b) Inverse CDF of the median persistence of problem clusters

Figure 4.3: Distributions of the prevalence and persistence of problem clusters. We find a natural skewed distribution with a few clusters having high prevalence. Many problem clusters last multiple hours and that a non-trivial number of problem clusters last for tens of hours.

the distribution of the prevalence of the problem clusters for the different quality metrics. We see a consistent pattern across all quality metrics that around 10% of the clusters have a prevalence greater than 8% across all metrics. In other words, many of these problem clusters are repeated observations that are recurrent problem events.

Prevalence of problem cluster: We define the *persistence* of a problem cluster in terms of the length of the consecutive occurrences of this cluster as a problem cluster. To this end, we coalesce consecutive occurrences of the cluster into a single logical event that lasts for multiple hours. For each problem cluster, we consider the distribution of the length of these "streaks" and report the median value. Figure 4.3b shows the distribution of the median persistence. For three of the metrics, more than 60% of the problem clusters have a median duration that last more than 2 hours.

4.1.3 Spatial Patterns

The previous results showed that there are a non-trivial number of persistent/prevalent problem clusters that last for several hours. As we discussed earlier, multiple problem clusters may be implicitly related by a single root cause as we saw in Figure 4.1. To this end, we focus next on the critical clusters using the algorithm described in Section 4.1.1. Recall that every critical clusters is also a problem cluster; i.e., it has a sufficiently high problem ratio and it has a significant number of sessions. The motivation to focus on a few critical cluster rather than all problem clusters is the observation (as shown shortly) that a small fraction of problem clusters cover most of the problem sessions.

Critical cluster analysis: Figure 4.4 shows the number of problem clusters relative to the number of critical clusters in the case of the Join Time metric. We see that number of critical clusters is almost 50$\times$ lower than the number of problem clusters suggesting that there are indeed

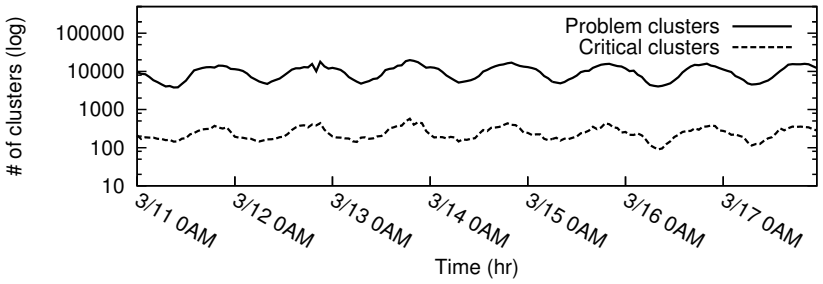


Figure 4.4: The number of critical clusters is significantly smaller than the number of problem clusters. The timeseries shown here is for the join time; we see similar results for the other quality metrics too.

Metric	Mean problem clusters	Mean critical clusters	Mean problem cluster coverage	Mean critical cluster coverage
BufRatio	10433	286 (2%)	0.8	0.66 (82%)
JoinTime	9953	247 (2%)	0.86	0.83 (96%)
JoinFailure	9620	302 (3%)	0.87	0.84 (96%)
Bitrate	9437	287 (3%)	0.57	0.44 (77%)

Table 4.1: Reduction via focusing only on critical clusters and the effective coverage of the critical clusters.

a small number of events that might have "caused" most problems. One natural question is whether the critical clusters *cover* most of the problem sessions. Table 4.1 summarizes the mean coverage and reduction of the critical clusters for the four quality metrics and in all cases, we see that the number of critical clusters is only 2-3% of the number of problem clusters (i.e., 50× fewer), but they manage to cover 44–84% of the problem sessions. As a point of reference, we also show the coverage of the problem clusters; i.e., not all sessions are part of a problem cluster as they may be part of small clusters or clusters with very small problem ratio. We see that the critical clusters cover almost all problem sessions that are part of some problem cluster; i.e., many of the coverage gaps are really due to problem sessions that belong to a statistically insignificant cluster (i.e., either with too few sessions or with too few problem sessions).

Next, we analyze the structure of the critical clusters for the different quality metrics. First, we analyze the types of client/session feature combinations that appear frequently in the critical clusters. Then, we analyze if the different metrics are correlated in the critical clusters. Finally, we highlight some interesting observations and some hypothesis to explain the most prevalent critical clusters.

Understanding most prevalent critical clusters: In order to illustrate the causes for the problem, we consider the critical clusters with a prevalence higher than 60% for the different quality metrics. For clarity of presentation, we only consider the critical clusters whose features fall in one of the following categories: ASN, CDN, Site, and ConnectionType as our previous breakdown shows these as the most dominant features. We present this analysis with two disclaimers. First, due to the sensitive nature of this data, we do not present the names of the actual providers,

	ASN	CDN	Site	ConnType
BufRatio	Asian ISPs	In-house, single bitrate	Single bitrate	Mobile wireless
JoinTime	Chinese ISPs accessing CDNs in China, but player loads modules from US CDN	In-house CDNs of UGC providers	High bitrates	
JoinFailure		Same set as buffering ratio	Same single global CDN, maybe low priority providers	
Bitrate	Wireless provider		UGC Sites	

Table 4.2: Analysis of the most prevalent critical clusters. A empty cell implies that we found no interesting cluster in this combination.

but focus on their characteristics. Second, this involves a fair amount of manual analysis and domain knowledge. As such, we intend this result to be illustrative (and somewhat speculative) rather than attempt to be conclusive. This said, we still believe that the high-level insights are still useful to inform future video delivery architectures.

Table 4.2 presents some of the anecdotal examples we observed. The empty cells simply indicate that there were no critical clusters in this category with a prevalence higher than 60%. We see a few interesting patterns here. In terms of buffering ratio, we see that the top ASNs are typically in Asia, and the content providers that had issues typically only had a single bitrate of content. The CDNs with buffering/join time problems are also typically "in-house" CDNs run by the Site itself; i.e., not a third-party CDN like Akamai or Limelight. We also see that wireless connections and wireless ISPs appear in the buffering and bitrate cells respectively, which is somewhat expected.

One interesting artifact we uncovered in the case of join time was that these were mostly ASNs in China accessing content from Chinese CDNs but there were third-party player modules loaded from US providers that led to higher join times. Another curious observation is that all the Sites with significant join failures tended to use the same global CDN. However, the CDN in aggregate does not have a significant presence in terms of failures, except in the case of these Sites.[5] We speculate that these, presumably low-end, providers may have lower priority service and could have potentially benefited from using multiple CDNs.

4.1.4 Cross-Metric Correlations

Next, we would like to know how much the critical clusters of different quality metrics correlate with each other. In other words, a different set of CDNs or Sites may be responsible for problems across buffering ratio and join time. To analyze this, we compute the *Jaccard similarity* index between the top-100 in terms of the total number of problem sessions covered critical clusters

[5]These Sites used a single CDN; recall that our critical cluster algorithm will prefer more compact descriptions and thus features these problems to the Site rather than the single Site-CDN combination.

BufRatio vs. Bitrate	BufRatio vs. JoinTime	BufRatio vs. JoinFailure	Bitrate vs. JoinTime	Bitrate vs. JoinFailure	JoinTime vs. JoinFailure
0.07	0.23	0.13	0.08	0.01	0.09

Table 4.3: Average Jaccard similarity index between the top 100 critical clusters for the different metrics. We see that most metrics are relatively uncorrelated, possibly because the critical features are very different.

for the different metrics. (The Jaccard similarity measure for two sets A and B is $\frac{|A \cap B|}{|A \cup B|}$.) We find that the overlap between the different metrics is only around 23% in the best case (buffering ratio and join time) and in the worst case is only around 1% (between bitrate and join failure). We manually analyzed the specific clusters and we found that the actual set of Site, CDN, and ASN critical clusters are indeed very different.

4.1.5 Key Observations

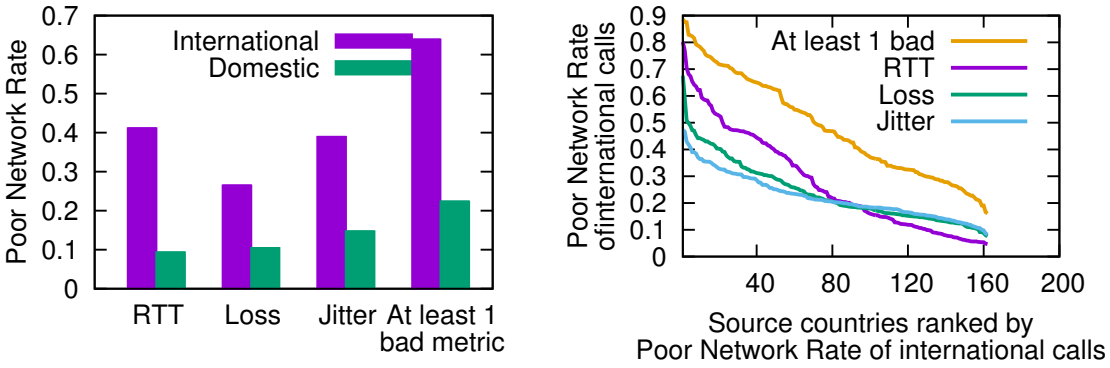
Our key observations from the analysis of problem clusters and critical clusters are:

- There is a distinct skewed distribution in the prevalence; around 8-12% of the problem clusters appear more than 10% of the time.
- There is also a skewed distribution in the persistence; more than 60% of problem clusters have a median duration greater than 2 hours.
- We find that a small number of critical clusters (2-3% of the number of problem clusters) can account for 44-84% of all problem sessions.
- While the set of feature combinations in the critical clusters that cover the most number of problem sessions is very similar across the quality metrics (i.e., Site, CDN, ASN), the actual values of these features is very different (with a max overlap of 23%).
- We see a few expected patterns such as Asian and wireless ISPs appearing as most prevalent critical clusters. We see some unexpected patterns that can be easily alleviated (e.g., the player modules loaded remotely for Chinese users) and Sites that could benefit from standard strategies such as using more fine-grained bitrates or using multiple CDNs.

4.2 Internet Telephony

We have seen in Section 2.1.2 that user experience is sensitive to poor network performance and that a significant fraction of calls suffer from poor performance when using default routing.

In this section, we use production data from a large VoIP service provide Skype (same dataset described in Section 2.1.2) to understand the QoE problems in Internet telephony based on a similar spatial and temporal analysis used in the last section. We begin by describing the dataset, QoE metrics, and the methodology of analyzing spatial and temporal patterns of VoIP QoE problems (Section 4.2.1), and present our results in Section 4.2.2, 4.2.3, and 4.2.4.



(a) International vs. domestic

(b) Countries of one side of a call

Figure 4.5: International vs. Domestic Calls.

4.2.1 Methodology

Identifying bad QoE: First, we define bad VoIP QoE by a similar threshold-based method as we defined problem sessions in video. We define the *poor network rate* (PNR) of a network metric for a set of calls as the fraction of calls whose performance on the metric is worse than the chosen thresholds: RTT $\geq$ 320ms, loss rate $\geq$ 1.2%, jitter $\geq$ 12ms. Recall from Figure 2.3 that these thresholds correspond to the user-specified poor call rate (PCR) of 0.3. These values are in line with literature from industry and standards bodies that recommend one-way end-to-end delay of no more than 150 ms and a packet loss rate of no more than 1% for good call quality [7, 19].

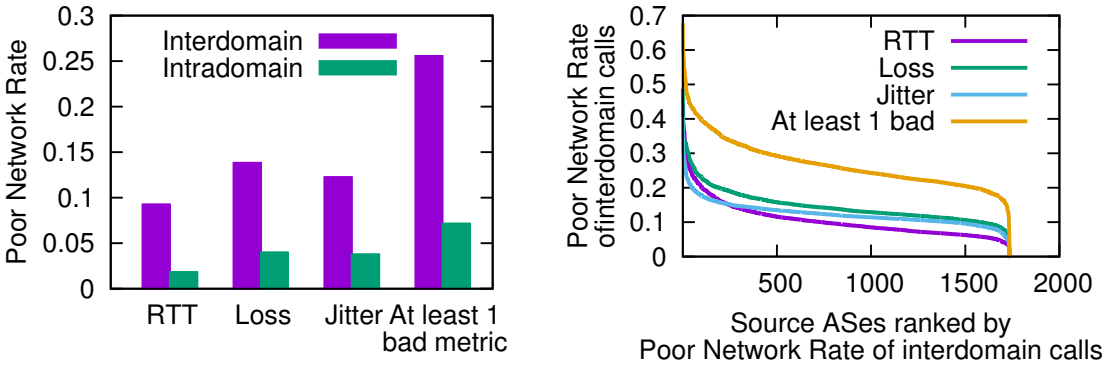
Clustering calls: Similarly to video streaming, we would like to understand whether the calls with bad network performance concentrate spatially (e.g., are they mostly International calls?) and persist over time. To this end, we group calls in the dataset based on different spatial features (e.g., geo-locations and IP prefixes of the caller and callee) as well as time-stamp (the date in which the call was made).

4.2.2 Spatial Patterns

International vs. domestic calls: On all three network metrics, we see that international calls (between users in different countries) have a higher PNR, i.e., they are more likely to suffer from bad network performance than domestic calls. Figure 4.5 shows a $2-3\times$ higher PNR on international calls than on domestic calls. The figures also show the fraction of calls with at least one metric being poor (the last pair of bars), where the gap between international and domestic calls is even larger. Though conclusively diagnosing the root cause of bad performance on international calls is hard and beyond the scope of this work, the higher PNR for international calls points to the WAN path as the culprit.[6]

To understand this further, Figure 4.5b zooms into the international calls and classifies them

[6]One aspect is that users tend to use VoIP regardless of its performance for international calls, unlike domestic calls.



(a) Inter-domain vs. intra-domain

(b) Source AS

Figure 4.6: Inter-domain vs. intra-domain calls.

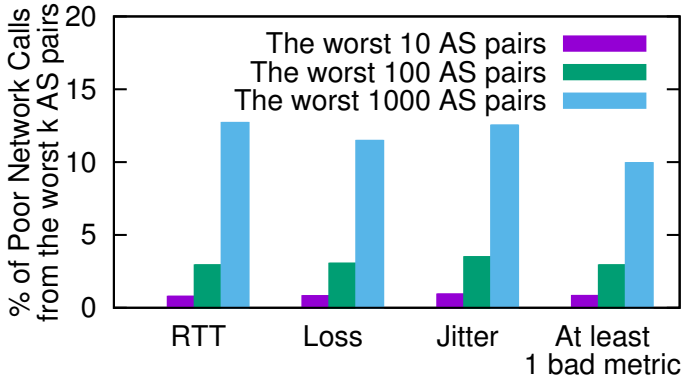
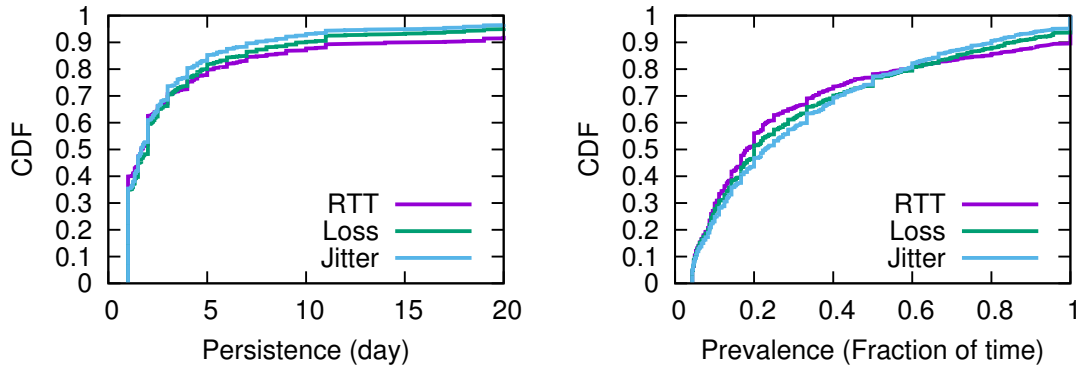


Figure 4.7: The percentage of calls over poor network conditions that come from the worst n AS pairs; AS-pairs are ranked in descending order of their contribution to total amount of calls with poor performance.

by the country of the callers (source). We see that there is a skewed distribution, with certain countries having a PNR as high as on the individual metrics. The PNR of international calls across the remaining countries drops gradually but half of them still see a non-negligible PNR of $25\% - 50\%$. This suggests that poor network performance is quite widespread, highlighting the suitability of a *globally* deployed overlay network that provides high performance inter-connection between overlay nodes.

Inter-AS vs. intra-AS calls: Similar to international calls, calls across ASes are $2-3\times$ more likely to experience poor network performance than those within the same AS domain. This, again, points to the need for enabling alternatives to default routing to improve WAN performance.

Not just a few problematic source-destination pairs: Contrary to our expectation, a few source-destination pairs alone do *not* account for a big chunk of the PNR. Figure 4.7 shows the fraction of calls that suffer from poor network performance from the worst AS pairs, ranked in



(a) Persistence (b) Prevalence

Figure 4.8: Temporal patterns of poor network performance. Figure 4.8a and 4.8b show the distribution of the persistence and prevalence of AS pairs having high PNR.

order of their contribution to the overall PNR. Even the worst 1000 AS pairs together only count for less than 15% of the overall PNR. This means that localized solutions that fix a few bad ASes or AS pairs, e.g., informing the AS administrators or the clients directly regarding their ISPs, are not sufficient.

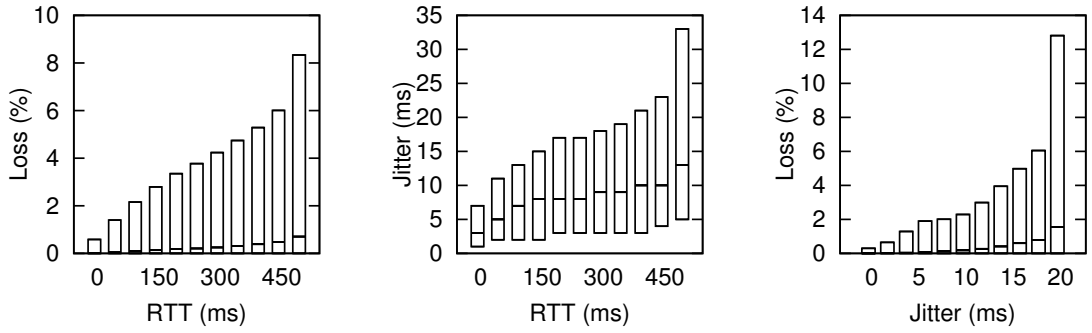
While the above analysis was at the granularity of ASes, we also tested at other, finer granularities (e.g., $/24$ and $/20$ prefixes of the caller and callee IP addresses) and found similar results (of not just a few culprits). In fact, for the pairs with sufficient data density at the $/24$ granularity, we found that performance distributions of the network metrics were similar to those at the granularity of ASes.

4.2.3 Temporal Patterns

We now analyze temporal patterns of poor network performance. We perform this analysis by grouping the performance of AS pairs into 24-hour time windows.[7] We conservatively label an AS pair as having *high PNR* for a specific metric (on a given day) if its PNR on that day is at least 50% higher than the overall PNR of all calls on that day.

Figure 4.8a and 4.8b show the distribution of *persistence* and *prevalence* of high PNR AS-pairs. The *persistence* of an AS pair is the median number of consecutive days when it has high PNR. The *prevalence* of an AS pair is the fraction of time it has high PNR. The figures show a highly skewed distribution with $10\% - 20\%$ AS pairs always having high PNR, while $60\% - 70\%$ AS pairs have poor performance for less than 30% of time and lasting no longer than one day at a stretch. This observation suggests that instead of statically configuring the system to improve performance for only the (relatively few) most prevalent and persistent AS pairs, we need to dynamically decide if a call should use default Internet routing or be relayed.

[7]Different grouping granularities yielded similar observations.



(a) RTT vs. loss rate (b) RTT vs. jitter (c) Jitter vs. loss rate

Figure 4.9: Pair-wise correlation between performance metrics. The Y-axis shows the distribution (10^{th}, 50^{th}, 90^{th} percentiles) of one metric as a function the other metric over the same set of calls.

4.2.4 Cross-Metric Correlations

As there could be dependencies between network metrics, improving one metric may increase PNR of another metric. Figure 4.9 shows the three pair-wise correlations. While the plot is based on an aggregation of data across all calls and paths, the substantial spread suggests at least the possibility that improving one performance metric *could* lead to a worsening of the other metrics. Therefore, we also focus on reducing PNR of three metrics collectively, i.e., minimizing how often *at least one* of the metrics is poor.

4.3 Summary

In this chapter, we have used empirical studies based on real large-scale datasets of video and VoIP QoE to shed light on the structures of QoE problems in the wild. Our findings suggest that there are persistent critical structures in the factors that determine video and VoIP QoE. Our key observations can be summarized as follows:

- *QoE depends on critical spatial structures.* Most quality problems can be attributed to a relatively small number of feature value combinations. We observe that in both Internet video and Internet telephony, bad quality can be attributed to smaller number of session-level features. For instance, we see that critical clusters of bad video quality only account for 2-3% of all problem clusters. Note that VoIP quality problems are relatively more spatially spread out than video quality problems, because VoIP quality depends on both sides of a call, while the video quality more often is determined by the client-side performance.
- *Many quality problems tend to persist on timescales of tens of minutes to hours.* We observe that a substantial fraction of problems last for multiple hours (even days). For instance, 60% of the problem clusters have a median duration that last more than 2 hours. At the same time, we note that both video and VoIP quality problems have highly skewed distributions in the prevalence and persistence, where some problems are still transit.

- *Quality problems of different metrics are correlated.* Finally, we observed that the quality problems of different metrics are likely to be correlated, suggesting that, instead of having to trading one metric for another, it is possible to improve multiple metrics simultaneously.

Chapter 5

Predictive QoE Optimization By Critical Feature Analysis

In this chapter, we present the first illustration of how DDN paradigm improves video QoE by formulating DDN as a prediction problem. Prior studies have shown that video quality can be substantially improved by optimally selecting the best CDN and bitrate for each video session, and the key to realize this potential is to build an video quality prediction system that can accurately predict the quality of a video session, if it were to use any combination of CDN and bitrate. However, building such a prediction system is challenging on two fronts. First, the relationships between video quality and observed session features can be quite complex. Second, video quality changes dynamically. Thus, we need a prediction model that is (a) expressive enough to capture these complex relationships, and (b) capable of updating quality predictions in near real-time. Unfortunately, several seemingly natural solutions (e.g., simple machine learning approaches and simple network models) fail on one or more fronts.

To address these challenges, we present *Critical Feature Analytics (CFA)*, which is inspired by the persistent critical structures of the QoE-determining factors. In particular, video quality is typically determined by a small subset of critical features whose criticality persists over several tens of minutes. This enables a scalable and accurate workflow where we automatically learn critical features for different sessions on coarse-grained timescales, while updating quality predictions in near real-time. Using a combination of real-world pilot deployment and trace-driven analysis, we demonstrate that CFA leads to significant improvements in video quality; e.g., 32% less buffering time and 12% higher bitrate than a random decision maker.

This chapter is organized as follows. Section 5.1 provides some background on the promise of video QoE prediction, and identifies key challenges in building an accurate video QoE prediction system. Then Section 5.2, Section 5.3, and Section 5.4 outlines the key design ideas behind CFA, the detailed design, and implementation of CFA, respectively. Section 5.5 presents real-world and trace-driven evaluation that demonstrates substantial quality improvement by CFA. Section 5.6 uses critical features learned by CFA to make interesting observations about video quality. Finally, Section 5.7 discusses some open issues in CFA, Section 5.8 discusses the related work, and Section 5.9 concludes the section.

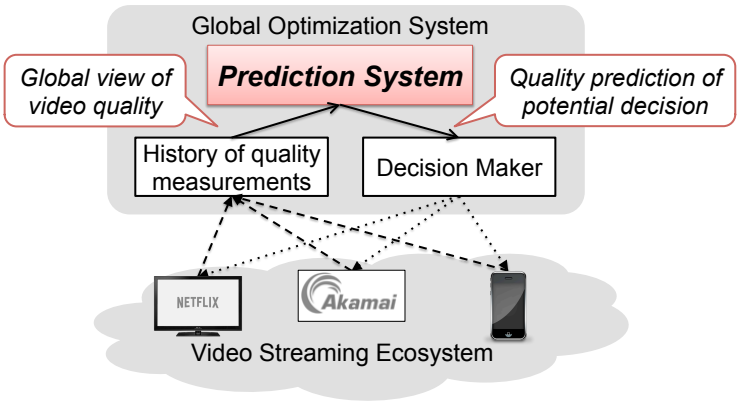


Figure 5.1: Overview of a global optimization system and the crucial role of a prediction system.

5.1 Background

This section begins with some background on video quality prediction. Then, we articulate two key challenges faced by any video quality prediction system: (1) The factors affecting video quality are complex, so we need expressive models; (2) Quality changes rapidly, so models must be updated in near real-time by recent quality measurements. We also argue why existing solutions do not address these challenges.

5.1.1 Data-Driven Quality Prediction

Prior work has made the case for a quality optimization system (Figure 5.1) that uses a *prediction oracle* to suggest the best parameter settings (e.g., bitrate, CDN) to optimize quality (e.g., [50, 101, 145, 147, 156]). Seen in a broader context, this predictive approach can be applied beyond Internet video (e.g., [41, 78, 81, 177, 193]).

In the context of video streaming, most video service providers today allow a video client (player) to switch CDN and bitrate among a set of available choices [101, 145, 147]. These switches have little overhead and can be performed at the beginning of and during a video playback [190]. Our goal then is to choose the best CDN and bitrate for a client by *accurately predicting the video quality of each hypothetical choice of CDN and bitrate*. In theory, if we can accurately predict the quality of each potential decision, then we can identify the optimal decision.

To this end, we envision a prediction system that uses a *global view* of quality measurements to make predictions for a specific video session. It learns a *prediction function* for each quality metric $Pred : 2^{\mathbb{S}} \times \mathbb{S} \mapsto \mathbb{R}$, which takes as input a given set of historical sessions $S \in 2^{\mathbb{S}}$ whose quality is already measured, and a new session $s \in \mathbb{S}$, and outputs a quality prediction $p \in \mathbb{R}$ for s.

Each quality measurement summarizes the quality of a video session for some duration of time (in our case, one minute). It is associated with values of four *quality metrics* (as defined in Section 2.1.1) and a set of *features* (summarized in Table 5.1). By feature, we refer to the type of attribute (e.g., *CDN*), rather than value of these attributes (e.g., $CDN = Akamai$) In general,

Features	Description
ASN	Autonomous System to which client IP belongs.
City	City where the client is located.
ConnectionType	Type of access network; e.g., mobile/fixed wireless, DSL, fiber-to-home.
Player	e.g., Flash, iOS, Silverlight, HTML5.
Site	Content provider of requested video contents.
LiveOrVoD	Binary indicator of live vs. VoD content.
ContentName	Name of the requested video object.
CDN	CDN a session started with.
Bitrate	Bitrate value the session started at.

Table 5.1: Quality metrics and session features associated with each session. *CDN* and *Bitrate* refer to initial CDN/bitrate values as we focus on initial selections.

the set of features depends on the degree of instrumentation and what information is visible to a specific provider. For instance, a CDN may know the location of servers, whereas a third-party optimizer [9] may only have information at the CDN granularity. Our focus is not to determine the best set of features that should be recorded for each session, but rather engineer a prediction system that can take an arbitrary set of features as inputs and extract the relationships between these features and video quality. In practice, the above set of features can already provide accurate predictions that help improve quality.

Our dataset consists of 6.6 million quality measurements collected from 2 million clients using 3 large public CDNs distributed across 168 countries and 152 ISPs.

Next, we show real examples of the complex factors that impact video quality, and the limitations of strawman solutions in capturing these relationships.

5.1.2 Challenge 1: Complex QoE-Determining Factors

High-dimensional relationship between video quality and session features: Video quality could be impacted by *combinations* of multiple components in the network. Such high-dimensional effects make it harder to learn the relationships between video quality and features, in contrast to simpler settings where features affect quality independently (e.g., assumed by Naive Bayes).

In a real-world incident, video sessions of Comcast users in Baltimore who watched videos from Level3 CDN experienced high failure rate (VSF) due to congested edge servers, shown by the blue line in Figure 5.2. The figure also shows the VSF of sessions sharing the same values on one or two features with the affected sessions; e.g., all Comcast sessions across different cities and CDNs. In the figure, the high VSF of the affected sessions cannot be clearly identified if we look at the sessions that match on only one or two features. Only when three features of *CDN* ("Level3"), *ASN* ("Comcast") and *City* ("Baltimore") are specified (i.e., blue line), can we detect the high VSF and predict the quality of affected sessions accurately.

In practice, we find that such high-dimensional effects are the common case, rather than an

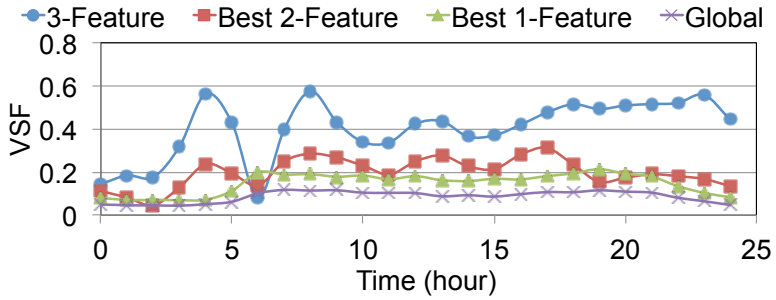


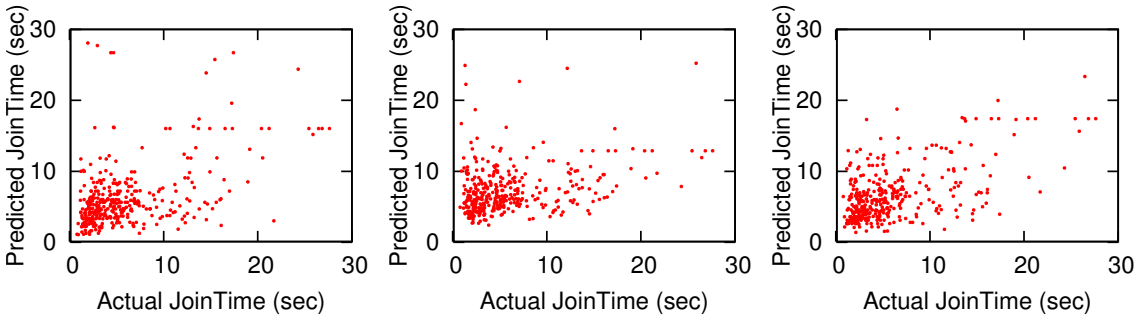
Figure 5.2: The high VSF is only evident when three factors (CDN, ISP and geo-location) are combined.

anomalous corner case. For instance, more than 65% of distinct CDN-ISP-City values have VSF that is at least 50% higher or lower than the VSF of sessions matching only one or two features (not shown). In other words, their quality is affected by a combined effect of at least three features.

Limitation of existing solutions: It might be tempting to develop simple predictors; e.g., based on the last-hop connection by using average quality of history sessions with the same *ConnectionType* value. However, they do not take into account the combined impact of features on video quality. Conventional machine learning techniques like Naive Bayes also suffer from the same limitation. In Figures 5.3a and 5.3b, we plot the actual JoinTime and the prediction made by the last-hop predictor and Naive Bayes (from `Weka` [30]) for 300 randomly sampled sessions. The figures also show the mean relative error ($\frac{|predicted-actual|}{actual}$). For each session, the prediction algorithms train models using historical sessions within a 10-minute interval prior to the session under prediction. It shows that the prediction error of both solutions is significant and two-sided (i.e., not fixable by normalization).

Highly diverse structures of factors. The factors that affect video quality vary across different sessions. This means the prediction algorithm should be expressive enough to predict quality for different sessions using different prediction models. For instance, the fact that many fiber-to-the-home (e.g., FiOS) users have high bitrates and people on cellular connections have lower bitrates is largely due to the speed of their last-mile connection. In contrast, some video clients may experience video loading failures due to unavailability of specific content on some CDNs. Chapter 4 has shown that many heterogeneous factors are correlated with video quality issues. In Section 5.2, we show that 15% of video sessions are impacted by more than 30 different combinations of features and give real examples of different factors that affect quality.

Limitation of existing solutions: To see why existing solutions are not sufficient, let us consider the k-nearest neighbor (k-NN) algorithm. It does not handle diverse relationships between quality and features, because the similarity between sessions is based on the same function of features independent of the specific session under prediction. In Figure 5.3c, we plot the actual values of JoinTime and the prediction made by k-NN with the same setup as Figure 5.3a(b). Similar to Naive Bayes and the last-hop predictor, k-NN has substantial prediction error.



(a) Last hop (0.76) (b) Naive Bayes (0.61) (c) k-NN (0.63)

Figure 5.3: Prediction error of some existing solutions is substantial (mean of relative error in parentheses).

5.1.3 Challenge 2: Fresh Updates

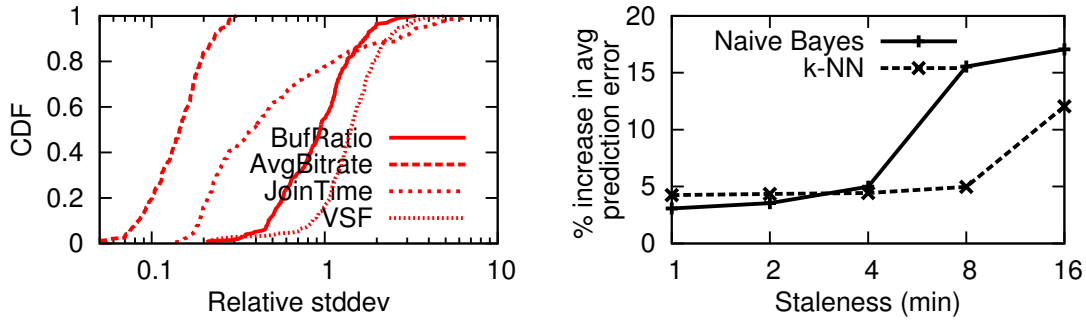
Video quality has significant temporal variability. In Figure 5.4a, for each quality metric and combination of specific CDN, city and ASN, we compute the mean quality of sessions in each 10-minute interval, and then plot the CDF of the relative standard deviation ($\frac{stddev}{mean}$) of the quality across different intervals. In all four quality metrics of interest, we see significant temporal variability; e.g., for 60% of CDN-city-ASN combinations, the relative standard deviation of JoinTime across different 10-minute intervals is more than 30%. Such quality variability has also been confirmed in other studies (e.g., [147]).

The implication of such temporal variability is that the prediction system must update models in near real-time. In Figure 5.4b, we use the same setup as Figure 5.3, except that the time window used to train prediction models is several minutes prior to the session under prediction. The figure shows the impact of such staleness on the prediction error for JoinTime. For both algorithms, prediction error increases dramatically if the staleness exceeds 10 minutes. As we will see later, this negative impact of staleness on accuracy is not specific to these prediction algorithms (Section5.5.3).

Limitation of existing solutions: The requirement to use the most recent measurements makes it infeasible to use computationally expensive models. For instance, it takes at least one hour to train an SVM-based prediction model from 15K quality measurements in a 10-minute interval for one video site, so the quality predictions will be based on information from more than one hour ago.

5.2 Overview of CFA Ideas

This section presents the domain-specific insights we use to help address the expressiveness challenge (Section 5.1.2). The first insight is that sessions matching on all features have similar video quality. However, this approach suffers from the curse of dimensionality. Fortunately, we can leverage a second insight that each video session has a subset of *critical features* that ultimately determine its video quality. We conclude this section by highlighting two outstanding



(a) Temporal variability

(b) Impact of staleness on accuracy

Figure 5.4: Due to significant temporal variability of video quality (left), prediction error increases dramatically with stale data (right).

Input: Session under prediction s, Previous sessions S
Output: Predicted quality p
/* S':identical sessions matching on all features with s in recent history(Δ) */
1 $S' \leftarrow SimilarSessionSet(s, S, AllFeatures, \Delta)$;
/* Summarize the quality (e.g.,median) of the identical sessions in S'. */
2 $p \leftarrow Est(S')$;
3 **return** p;

Algorithm 1: ***Baseline prediction that finds sessions matching on all features and uses their observed quality as the basis for prediction.***

issues in translating these insights into a practical prediction system.

5.2.1 Baseline Prediction Algorithm

Our first insight is that sessions that have identical feature values will naturally have similar (if not identical) quality. For instance, we expect that all Verizon FiOS users viewing a specific HBO video using Level3 CDN in Pittsburgh at Fri 9 am should have similar quality (modulo very user-specific effects such as local Wi-Fi interference inside the home). We can summarize the intuition as follows:

Insight 1: ***At a given time, video sessions having same value on every feature have similar video quality.***

Inspired by Insight 1, we can consider a baseline algorithm (Algorithm 1). We predict a session's quality based on "identical sessions", i.e., those from recent history that match values on *all* features with the session under prediction. Ideally, given infinite data, this algorithm is accurate, because it can capture all possible combinations of factors affecting video quality.

However, this algorithm is unreliable as it suffers from the classical curse of dimensional-

Quality issue	Set of critical features
Issue on one player of Vevo	$\{Player, Site\}$
ESPN flipping between CDNs	$\{CDN, Site, ContentName\}$
Bad Level3 servers for Comcast users in Maryland	$\{CDN, City, ASN\}$

Table 5.2: Real-world examples of critical features confirmed by analysts at a large video optimization vendor.

ity [168]. Specifically, given the number of combinations of feature values (ASN, device, content providers, CDN, just to name a few), it is hard to find enough identical sessions needed to make a robust prediction. In our dataset, more than 78% of sessions have no identical session (i.e., matching on all features) within the last 5 minutes.

5.2.2 Critical Features

In practice, we expect that some features are more likely to "explain" the observed quality of a specific video session than others. For instance, if a specific peering point between Comcast and Netflix in New York is congested, then we expect most of these users will suffer poor quality, regardless of the speed of their local connection.

Insight 2: ***Each video session has a subset of* critical features *that ultimately determines its video quality.***

We already saw some real examples in Section 5.1.2: in the example of high dimensionality, the critical features of the sessions affected by the congested Level3 edge servers are $\{ASN, CDN, City\}$; in the examples of diversity, the critical features are $\{ConnectionType\}$ and $\{CDN, ContentName\}$. Table 5.2 gives more real examples of critical features that we have observed in operational settings and confirmed with domain experts.

A natural implication of this insight is that it can help us tackle the curse of dimensionality. Unlike Algorithm 1, which fails to find a sufficient number of sessions, we can estimate quality more reliably by aggregating observations across a larger amount of "similar sessions" that only need to match on these *critical features*. Thus, critical features can provide expressiveness while avoiding curse of dimensionality.

Algorithm 2 presents a logical view of this idea:

1. **Critical feature learning (line 1):** First, find the critical features of each session s, denoted as $CriticalFeatures(s)$.
2. **Quality estimation (line 2, 3):** Then, find similar sessions that match values with s on critical features $CriticalFeatures(s)$ within a recent history of length Δ (by default, 5 minutes). Finally, return some suitable estimate of the quality of these similar sessions; e.g., the median[1] (for BufRatio, AvgBitrate, JoinTime) or the mean (for VSF).

[1] We use median because it is more robust to outliers.

Input: Session under prediction s, Previous sessions S
Output: Predicted quality p
/* CF_s:Set of critical features of s */
1 $CF_s \leftarrow CriticalFeatures(s)$;
/* S':Similar sessions matching values on critical features CF_s with s. */
2 $S' \leftarrow SimilarSessionSet(s, S, CF_s, \Delta)$;
/* Summarize the quality of the similar sessions in S'. */
3 $p \leftarrow Est(S')$;
4 **return** p;

Algorithm 2: ***CFA prediction algorithm, where prediction is based on similar sessions matching on critical features.***

A practical benefit of Algorithm 2 is that it is interpretable [212], unlike some machine learning algorithms (e.g., PCA or SVM). This allows domain experts to combine their knowledge with CFA and diagnose prediction errors or resolve incidents, as we explore in Section 5.6.2.

At this time, it is useful to clarify what critical features are and what they are not. In essence, critical features provide the explanatory power of how a prediction is made. However, critical features are not a minimal set of factors that determine the quality (i.e., root cause). That is, they can include both features that reflect the root cause as well as additional features. For example, if all HBO sessions use Level3, their critical features may include both *CDN* and *Site*, even if *CDN* is redundant, since including it does not alter predictions. The primary objective of CFA is accurate prediction; root cause diagnosis may be an added benefit.

5.3 Design of CFA

In this section, we present the detailed design of CFA and discuss how we address the two practical challenges mentioned in the previous section: learning critical features and reducing update delay.

The key to addressing these challenges is our third and final domain-specific insight:

Insight 3: ***Critical features tend to*** **persist** ***on long timescales of tens of minutes.***

This insight is derived from prior measurement studies. For instance, our measurement study in Chapter 4 on shedding light on video quality issues in the wild showed that the factors that lead to poor video quality persist for hours, and sometimes even days. Another recent study from the C3 system suggests that the best CDN tends to be relatively stable on the timescales of few tens of minutes [101]. We independently confirm this observation in Section 5.5.3 that using slightly stale critical features (e.g., 30-60 minutes ago) achieves similar prediction accuracy as using the most up-to-date critical features. Though this insight holds for most cases, it is still possible (e.g., on mobile devices) that critical features persist on a relatively shorter timescale (e.g., due to the nature of mobility).

Note that the persistence of critical features does not mean that quality values are equally

Notations	Domains	Definition
$s, S, \mathbb{S}$		A session, a set of sessions, set of all sessions
$q(s)$	$\mathbb{S} \mapsto \mathbb{R}$	Quality of s
$QualityDist(S)$	$2^{\mathbb{S}} \mapsto 2^{\mathbb{R}}$	$\{q(s) \mid s \in S\}$
$f, F, \mathbb{F}$		A feature, a set of features, set of all features
$CriticalFeatures(s)$	$\mathbb{S} \mapsto 2^{\mathbb{F}}$	Critical features of s
$\mathbb{V}$		Set of all feature values
$FV(f,s)$	$\mathbb{F} \times \mathbb{S} \mapsto \mathbb{V}$	Value on feature f of s
$FSV(F,s)$	$2^{\mathbb{F}} \times \mathbb{S} \mapsto 2^{\mathbb{V}}$	Set of values on features in F of s
$SimilarSessionSet$ (s,S,F,Δ)	$\mathbb{F} \times 2^{\mathbb{F}} \times \mathbb{S} \times \mathbb{R}^+ \mapsto 2^{\mathbb{F}}$	$\{s' \mid s' \in S, t(s) - \Delta < t(s') < t(s), FSV(F,s') = FSV(F,s)\}$

Table 5.3: Notations used in learning of critical features.

persistent. In fact, persistence of critical features is on a timescale an order of magnitude longer than the persistence of quality. That is, even if quality fluctuates rapidly, the critical features that determine the quality do not change as often.

As we will see below, this persistence enables (a) automatic learning of critical features from history, and (b) a scalable workflow that provides up-to-date estimates.

5.3.1 Learning Critical Features

Recall that the first challenge is obtaining the critical features for each session. The persistence of critical features has a natural corollary that we can use to automatically learn them:

Corollary 3.1: ***Persistence implies that critical features of a session are learnable from history.***

Specifically, we can simply look back over the history and identify the subset of features F such that the quality distribution of sessions matching on F is most similar to that of sessions matching on *all* features. For instance, suppose we have three features $\langle ContentName, ASN, CDN \rangle$ and it turns out that sessions with $ASN = Comcast, CDN = Level3$ consistently have high buffering over the last few hours due to some internal congestion at the corresponding exchange point. Then, if we look back over the last few hours, the data from history will naturally reveal that the distribution of the quality of sessions with the feature values $\langle ContentName = Foo, ASN = Comcast, CDN = Level3 \rangle$ will be similar to $\langle ContentName = *, ASN = Comcast, CDN = Level3 \rangle$, but very different from, say, the quality of sessions in $\langle ContentName = *, ASN = *, CDN = Level3 \rangle$, or $\langle ContentName = *, ASN = Comcast, CDN = * \rangle$. Thus, we can use a data-driven approach to learn that ASN, CDN are the critical features for sessions matching $\langle ContentName = Foo, ASN = Comcast, CDN = Level3 \rangle$.

Algorithm 3 formalizes this intuition for learning critical features. Table 5.3 summarizes the notation used in Algorithm 3. For each subset of features F (line 3), we compute the similarity between the quality distribution (D_F) of sessions matching on F and the quality distribution (D_{finest}) of sessions matching on all features (line 7). Then, we find the F that yields the maximum similarity (line 8-10), under one additional constraint that $SimilarSessionSet(s,S,F,\Delta)$

Input: Session under prediction s, Previous sessions S
Output: Critical features for s

```
/* Initialization                                                    */
```
1 $MaxSimilarity \leftarrow -\infty, CriticalFeatures \leftarrow NULL$;
```
/* D_finest : Quality distribution of sessions matching on F in Δ_learn.
   */
```
2 $D_{finest} \leftarrow QualityDist(SimilarSessionSet(s,S,\mathbb{F},\Delta_{learn}))$;
3 **for** $F \subseteq 2^{\mathbb{F}}$ **do**
```
    /* Exclude F without enough similar sessions for prediction.
       */
```
4 **if** $|SimilarSessionSet(s,S,F,\Delta)| < n$ **then**
5 **continue;**
```
    /* D_F : Quality distribution of sessions matching on F in Δ_learn.
       */
```
6 $D_F \leftarrow QualityDist(SimilarSessionSet(s,S,F,\Delta_{learn}))$;
```
    /* Get similarity of D_finest & D_F.                             */
```
7 $Similarity \leftarrow Similarity(D_F, D_{finest})$;
8 **if** $Similarity > MaxSimilarity$ **then**
9 $MaxSimilarity \leftarrow Similarity$;
10 $CriticalFeatures \leftarrow F$;
11 **return** *CriticalFeature*;

Algorithm 3: ***Learning of critical features.***

should include enough (by default, at least 10) sessions to get reliable quality estimation (line 4-5). This check ensures that the algorithm will not simply return the set of all features.

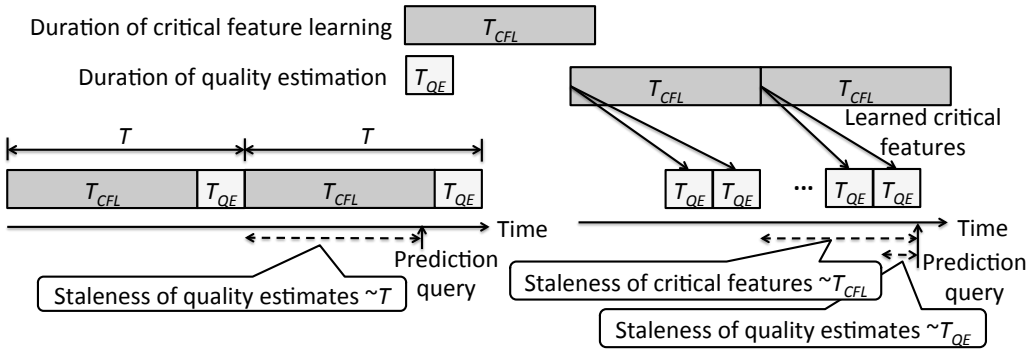
As an approximation of the duration in which critical features persist, we use $\Delta_{learn} = 60min$. Note that Δ_{learn} is an order of magnitude larger than the time window Δ used in quality estimation, because critical features persist on a much longer timescale than quality values. We use (the negative of) Jensen-Shannon divergence between D_1 and D_2 to quantify their similarity $Similarity(D_1, D_2)$.

Although Algorithm 3 can handle most cases, there are corner cases where $SimilarSessionSet(s,S,\mathbb{F},\Delta_{learn})$ does not have enough sessions (i.e., more than n) to compute $Similarity(D_F, D_{finest})$ reliably. In these cases, we replace D_{finest} by the set of n sessions that share most features with s over the time window of Δ_{learn}. Formally, we use $\{s' | s'$ matches k_s features with $s\}$, where $k_s = \text{argmin}_k(|\{s' | s' \text{ matches } k \text{ features with } s\}| \geq n\})$.

5.3.2 Using Fresh Updates

Next, we focus on reducing the update delay between when a quality measurement is received and used for prediction.

Naively running critical feature learning and quality estimation of Algorithm 2 can be time-consuming, causing the predictions to rely on stale data. In Figure 5.5(a), T_{CFL} and T_{QE} are the duration of critical feature learning and the duration of quality estimation, respectively. The staleness of quality estimation (depicted in Figure 5.5) to respond to a prediction query can



(a) Naive workflow (b) CFA workflow

Figure 5.5: To reduce update delay, we run critical feature learning and quality estimation at different timescales by leveraging persistence of critical features.

be as large as the total time of two steps (i.e., $T_{CFL} + T_{QE}$), which typically is tens of minutes (Section 5.5.3). Also, simply using more parallel resources is not sufficient. The time to learn critical features using Algorithm 3 grows linearly with the number of sessions under prediction, the number of history sessions, and the number of possible feature combinations. Thus, the complexity of learning critical features T_{CFL} is exponential in the number of features. Given the current set of features, T_{CFL} is on the scale of tens of minutes.

To reduce update delay, we again leverage the persistence of critical features:

Corollary 3.2: ***Persistence implies that critical features can be cached and reused over tens of minutes.***

Building on Corollary 3.2, we decouple the critical feature learning and quality estimation steps, and run them at separate timescales. On the timescale of tens of minutes, we update the results of critical feature learning. Then, on a faster timescale of tens of seconds, we update quality estimation using fresh data and the most recently learned critical features.

This decoupling minimizes the impact of staleness on prediction accuracy. Learning critical features on the timescale of tens of minutes is sufficiently fast as they persist on the same timescale. Meanwhile, quality estimation can be updated every tens of seconds and makes predictions based on quality updates with sufficiently low staleness. Thus, the staleness of quality estimation T_{QE} of the decoupled workflow (Figure 5.5(b)) is a magnitude lower than $T_{QE} + T_{CFL}$ of the naive workflow (Figure 5.5(a)). In Section 5.5.3, we show that this workflow can retain the freshness of critical features and quality estimates.

In addition, CFA has a natural property that two sessions sharing all feature values and occurring close in time will map to the same critical features. Thus, instead of running the steps per-session, we can reduce the computation to the granularity of *finest partitions*, i.e., distinct values of all features.

5.3.3 Putting It Together

Building on these insights, we create the following practical *three-stage* workflow of CFA.

- **Stage I: Critical feature learning** (line 1 of Algorithm 2) runs offline, say, every tens of minutes to an hour. The output of this stage is a key-value table called *critical feature function* that maps all observed finest partitions to their critical features.
- **Stage II: Quality estimation** (line 2,3 of Algorithm 2) runs every tens of seconds for all observed finest partitions based on the most recent critical features learned in the first stage. This outputs another key-value table called *quality function* that maps a finest partition to the quality estimation, by aggregating the most recent sessions with the corresponding critical features.
- **Stage III: Real-time query/response.** Finally, we provide real-time query/response on the arrival of each client, operating at the millisecond timescale, by simply looking up the most recent precomputed value function from the previous stage. These operations are simple and can be done very fast.

Finally, instead of forcing all finest partition-level computations to run in every batch, we can do triggered recomputations of critical feature learning only when the observed prediction errors are high.

5.4 Implementation and Deployment

This section presents our implementation of CFA and highlights engineering solutions to address practical challenges in operational settings (e.g., avoiding bulk data loading and speeding up development iterations).

5.4.1 Implementation of CFA Workflow

CFA's three stages are implemented in two different locations: a centralized backend cluster and geographically distributed frontend clusters as depicted in Figure 5.6.

Centralized backend: The critical feature learning and quality estimation stages are implemented in a backend cluster as periodic jobs. By default, critical feature learning runs every 30 minutes, and quality estimation runs every minute. A centralized backend is a natural choice because we need a global view of all quality measurements. The quality function, once updated by the estimation step, is disseminated to distributed frontend clusters using Kafka [134].

Note that we can further reduce learning time using simple parallelization strategies. Specifically, the critical features of different finest partitions can be learned independently. Similarly in Algorithm 3, the similarity of quality distributions can be computed in parallel. To exploit this data-level parallelism, we implement them as Spark jobs [28].

Distributed frontend: Real-time query/response and decision makers of CDN/bitrate are co-located in distributed frontend clusters that are closer to clients than the backend. Each frontend cluster receives the quality function from the backend and caches it locally for fast prediction. This reduces the latency of making decisions for clients.

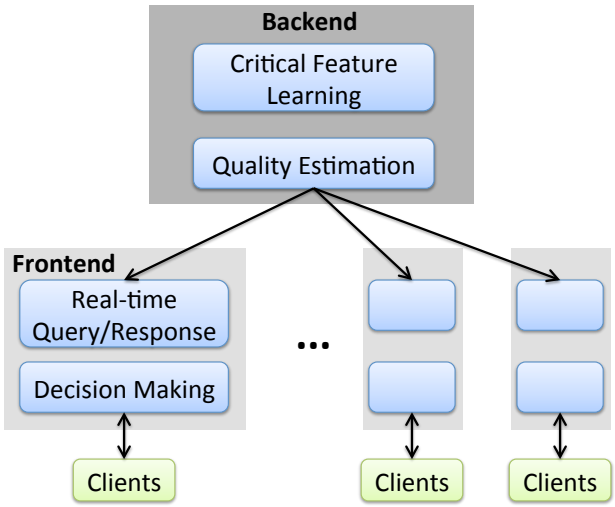


Figure 5.6: Implementation overview of CFA. The three stages of CFA workflow are implemented in a backend cluster and distribute frontend clusters.

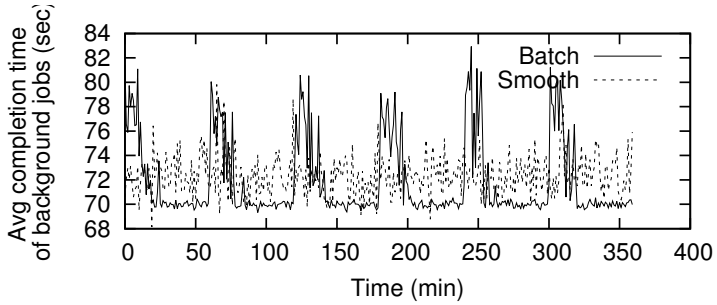
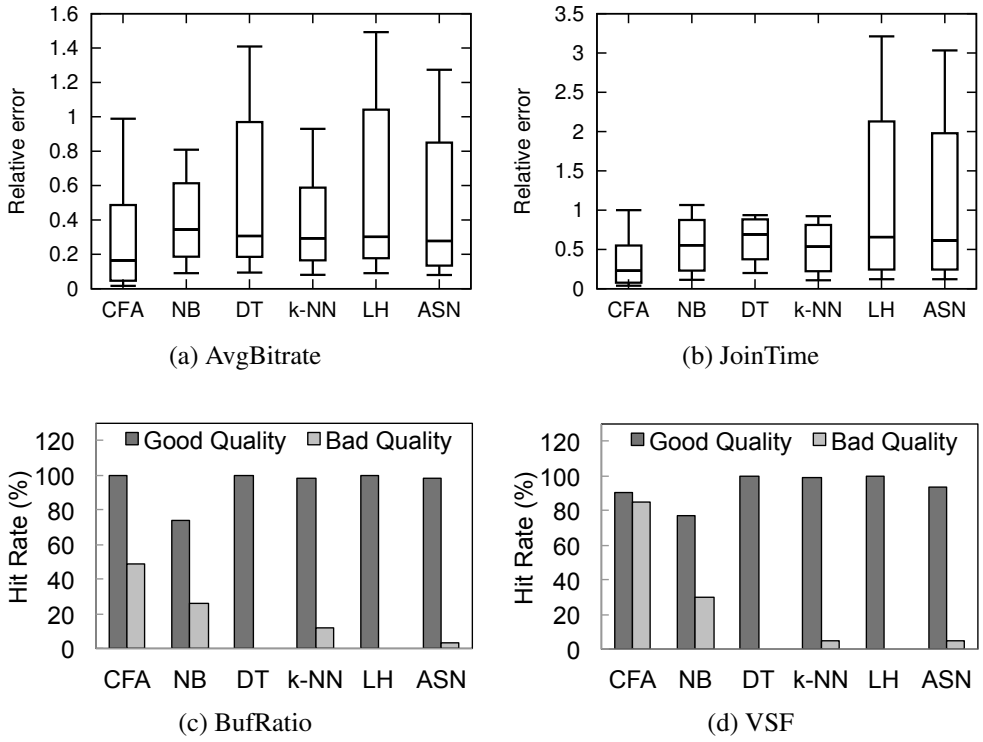


Figure 5.7: Streaming data loading has smoother impact on completion delay than batch data loading.

5.4.2 Challenges in an Operational Setting

Mitigating impact of bulk data loading: The backend cluster is shared and runs other delay-sensitive jobs; e.g., analytics queries from production teams. Since the critical feature learning runs periodically and loads a large amount of data (≈30 GB), it creates spikes in the delays of other jobs (Figure 5.7). To address this concern, we engineered a simple heuristic to evenly spread the data retrieval where we load a small piece of data every few minutes. As Figure 5.7 shows, this reduces the spikes caused by bulk data loading in batch mode. Note that this does not affect critical feature learning.

Iterative algorithm refinement: Some parameters (e.g., learning window size Δ_{learn}) of CFA require iterative tuning in a production environment. However, one practical challenge is that the frontend-facing part of the backend can only be updated once every couple of weeks due to code release cycles. Thus, rolling out new prediction algorithms may take several days and is a practical concern. Fortunately, the decoupling between critical feature learning and quality estimation (Section 5.3.2) means that changes to critical feature learning are confined to the



(a) AvgBitrate (b) JoinTime

(c) BufRatio (d) VSF

Figure 5.8: Distributions of relative prediction error ($\{5, 10, 50, 90, 95\}$%iles) on AvgBitrate and JoinTime and hit rates on BufRatio and VSF. They show that CFA outperforms other algorithms.

backend cluster. This enables us to rapidly refine and customize the CFA algorithm.

5.5 Evaluation

In this section, we show that:

- CFA predicts video quality with 30% less error than competing machine learning algorithms (Section 5.5.1).
- Using CFA-based prediction, we can improve video quality significantly; e.g., 32% less BufRatio, 12% higher AvgBitrate in a pilot deployment (Section 5.5.2).
- CFA is responsive to client queries and makes predictions based on the most recent critical features and quality measurements (Section 5.5.3).

5.5.1 Prediction Accuracy

We compare CFA with five alternative algorithms: three simple ML algorithms, Naive Bayes (NB), Decision Tree (DT), k-Nearest Neighbor (k-NN)[2], and two heuristics which predict a session's quality by the average quality of other sessions from the same ASN (ASN) or matching

[2]NB, DT, and k-NN are mplemented using a popular ML library `weka`[30].

	CFA	Baseline	Improvement
QoE	155.43	138.27	12.4%
BufRatio	0.0123	0.0182	32%
AvgBitrate	3200	2849	12.31%

Table 5.4: Random A/B testing results of CFA vs. baseline in real-world deployment.

the last-mile connection type (LH). All algorithms use the same set of features listed in Table 5.1.

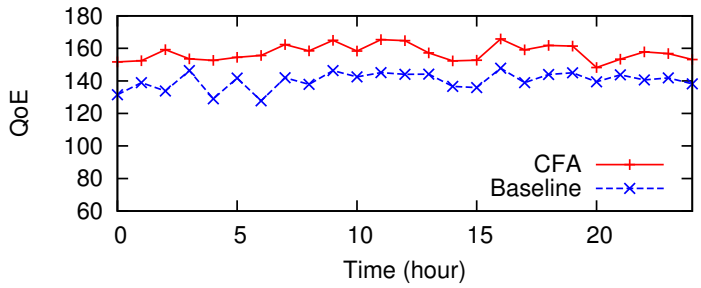
Ideally, we want to evaluate how accurately an algorithm can predict the quality of a given client on every choice of CDN and bitrate. However, this is infeasible since each video client is assigned to only one CDN and bitrate at any time. Thus, we can only evaluate the prediction accuracy over the observed CDN-bitrate choices, and we use the quality measured on these choices as the ground truth. That said, this approach is still useful for doing a relative comparison across different algorithms.

For AvgBitrate and JoinTime, we report *relative error*: $\frac{|p-q|}{q}$, where the q is the ground truth and p is the prediction. For BufRatio and JoinTime, which have more "step function" like effects [88], we report a slightly different measure called *hit rate*: how likely a session with good quality (i.e., BufRatio $< 5\%$, VSF=0) or bad quality is correctly identified. Figure 5.8 shows that for AvgBitrate and JoinTime, CFA has the lowest $\{5, 10, 50, 90\}$%th percentiles of prediction error and lower 95%th percentiles than most algorithms. In particular, median error of CFA is about 30% lower than the best competing algorithm. In terms of BufRatio and VSF, CFA significantly outperforms other algorithms in the hit rate of bad quality sessions. The reason for hit rate of bad quality to be lower than that of good quality is that bad quality sessions are almost always less than good quality, which makes them hard to predict. Note that accurately identifying sessions that have bad quality is crucial as they have the most room for improvement.

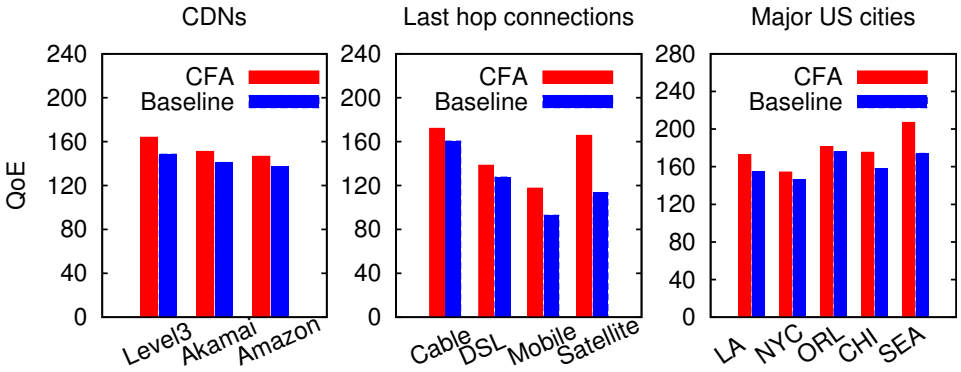
5.5.2 Quality Improvement

Pilot deployment: As a pilot deployment, we integrated CFA in a production system that provides a global video optimization service [101]. We deployed CFA on one major content provider and used it to optimize 150,000 sessions each day. We ran an A/B test (where each algorithm was used on a random subset of clients) to evaluate the improvement of CFA over a baseline random decision maker, which many video optimization services use by default (modulo business arrangement like price) [38].

Table 5.4 compares CFA with the baseline random decision maker in terms of the mean BufRatio, AvgBitrate and a simple QoE model ($QoE = -370 * BufRatio + AvgBitrate/20$), which was suggested by [88, 147]. Over all sessions in the A/B testing, CFA shows an improvement in both BufRatio (32% reduction) and AvgBitrate (12.3% increase) compared to the baseline. This shows that CFA is able to simultaneously optimize multiple (possibly conflicting) metrics. To put these numbers in context, our conversation with domain experts confirmed that these improvements are significant for content providers and can potentially translate into substantial benefits in engagement and revenues [10]. CFA's superior performance and that CFA is more automated than the custom algorithm indicate that domain experts were willing to invest



(a) CFA vs. baseline by time



(b) CFA vs. baseline by spatial partitions

Figure 5.9: Results of real-world deployment. CFA outperforms the baseline random decision maker (over time and across different large cities, connection t ypes and CDNs).

time running longer pilot. Figure 5.9 provides more comparison and shows that CFA consistently outperforms the baseline over time and across different major cities in the US, connection types and CDNs.

Trace-driven simulation: We complement this real-world deployment with a trace-driven simulation to simultaneously compare more algorithms over more quality metrics. However, one key challenge is that it is hard to estimate the quality of a decision that was not used by a specific client in the trace.

To address this problem, we use the *counterfactual methodology* from prior work in online recommendation systems [140, 142]. Suppose we have quality measurements from a set of clients, where client c is assigned to a decision $d_{rand}(c)$ of CDN and bitrate at random. Now, we have a new hypothetical algorithm that maps client c to $d_{alg}(c)$. Then, we can evaluate the average quality of clients assigned to each decision d, $\{c|d_{alg}(c) = d\}$, by the average quality of $\{c|d_{alg}(c) = d, d_{rand}(c) = d\}$. Finally, the average quality of the new algorithm is the weighted sum of average quality of all decisions, where the weight of each decision is the fraction of sessions assigned to it. This can be proved to be an unbiased (offline) estimate of d_{alg}'s (online) performance [49].[3] For instance, if out of 1000 clients assigned to use Akamai and 500Kbps,

[3]One known limitation of this analysis is that it assumes the new assignments do not affect each decision's overall performance. For instance, if we assign all sessions to one CDN, they may overload the CDN and so this CDN's quality in the random assignments is no longer useful. Since this work only focuses on controlling traffic at

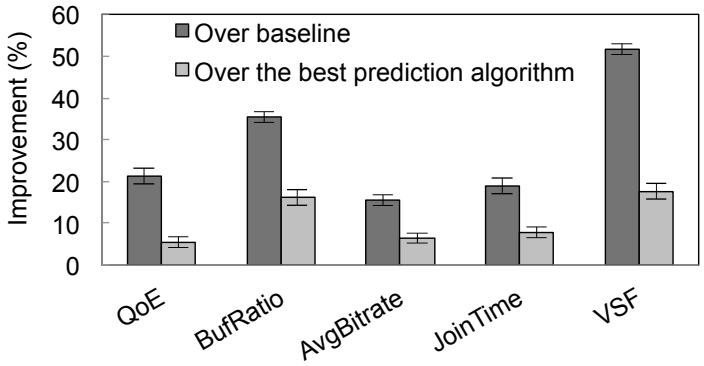


Figure 5.10: Comparison of quality improvement between CFA and strawmen.

Stage	Run time (mean / median)	Required freshness
Critical feature learning	30.1/29.5 min	30-60 min
Quality estimation	30.7/28.5 sec	1-5 min
Query response	0.66/0.62 ms	1 ms

Table 5.5: Each stage of CFA is refreshed to meet the required freshness of its results.

200 clients are assigned to this decision in the random assignment, then we can use the average quality of these 200 sessions as an unbiased estimate of the average quality of these 1000 sessions. Fortunately, our dataset includes a (randomly chosen) portion of clients with randomized decision assignments (i.e., CDN and bitrate). Thus, we only report improvements for these clients.

Figure 5.10 uses this counterfactual methodology and compares CFA with the best alternative from Section 5.5.1 for each quality metric and the baseline random decision maker (e.g., the best alternative of AvgBitrate is k-NN). For each quality metric and prediction algorithm, the decision maker selects the CDN and bitrate that has the best predicted quality for each client. For instance, the improvement of CFA over the baseline on VSF is 52% – this means the number of sessions with start failures is 52% less than when the baseline algorithm is used. The figures show that CFA outperforms the baseline algorithm by 15%-52%. They also show that CFA outperforms the best prediction algorithms by 5%-17%.

5.5.3 Timeliness of Prediction

Our implementation of CFA should (1) retain freshness to minimize the impact of staleness on prediction accuracy, and (2) be responsive to each prediction query.

We begin by showing how fast each stage described in Section 5.3.2 needs to be refreshed. Figure 5.11 shows the impact of staleness of critical features and quality values on the prediction accuracy of CFA. First, critical features learned 30-60 minutes before prediction can still achieve

a small scale relative to the total load on the CDN (and our experiments are in fact performed at such a scale), this methodology is still unbiased.

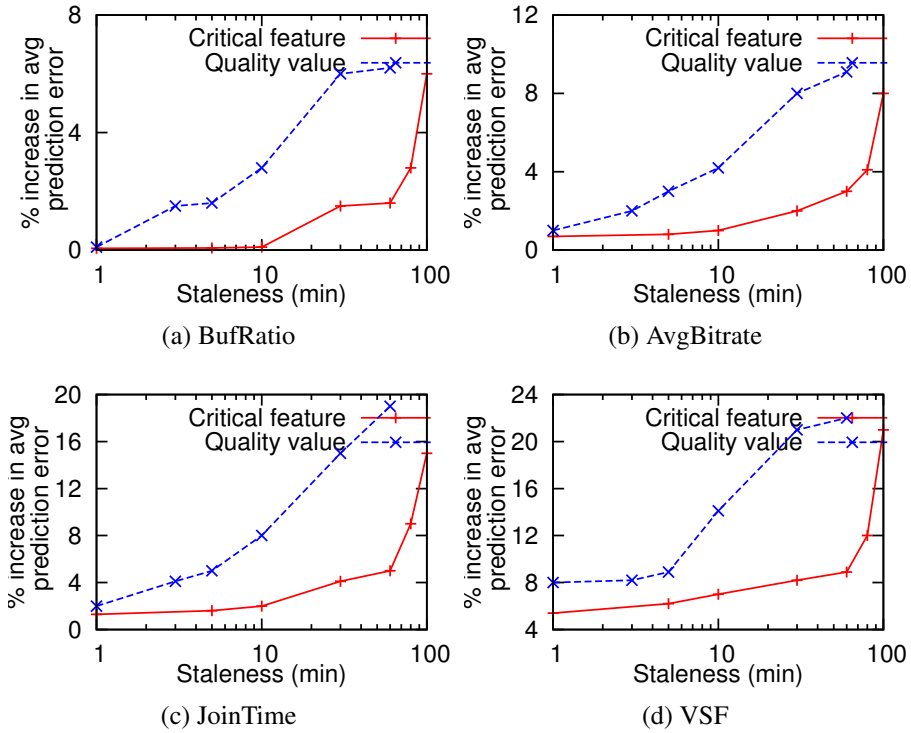


Figure 5.11: Latency of critical features and quality values (x-axis) on increase in accuracy (y-axis).

similar accuracy as those learned 1 minute before prediction. In contrast, quality estimation cannot be more than 10 minutes prior to when prediction is made (which corroborates the results of Figure 5.4b). Thus, critical feature learning needs to be refreshed every 30-60 minutes and quality estimation should be refreshed at least every several minutes. Finally, prediction queries need to be responded to within several milliseconds [101] (ignoring network delay between clients and servers).

Next, we benchmark the time to run each logical stage described in Section 5.3.2. Real-time query/response runs in 4 geographically distributed data centers. Critical feature learning and quality estimation run on two clusters of 32 cores. Table 5.5 shows the time for running each stage and the timescale required to ensure freshness. It confirms that the implementation of CFA is sufficient to ensure the freshness of results in each stage.

5.6 Insights from Critical Features

In addition to the predictive power, CFA also offers insights into the "structure" of video quality in the wild. In this section, we focus on two questions: (1) What types of critical features are most common? (2) What factors have significant impact on video quality?

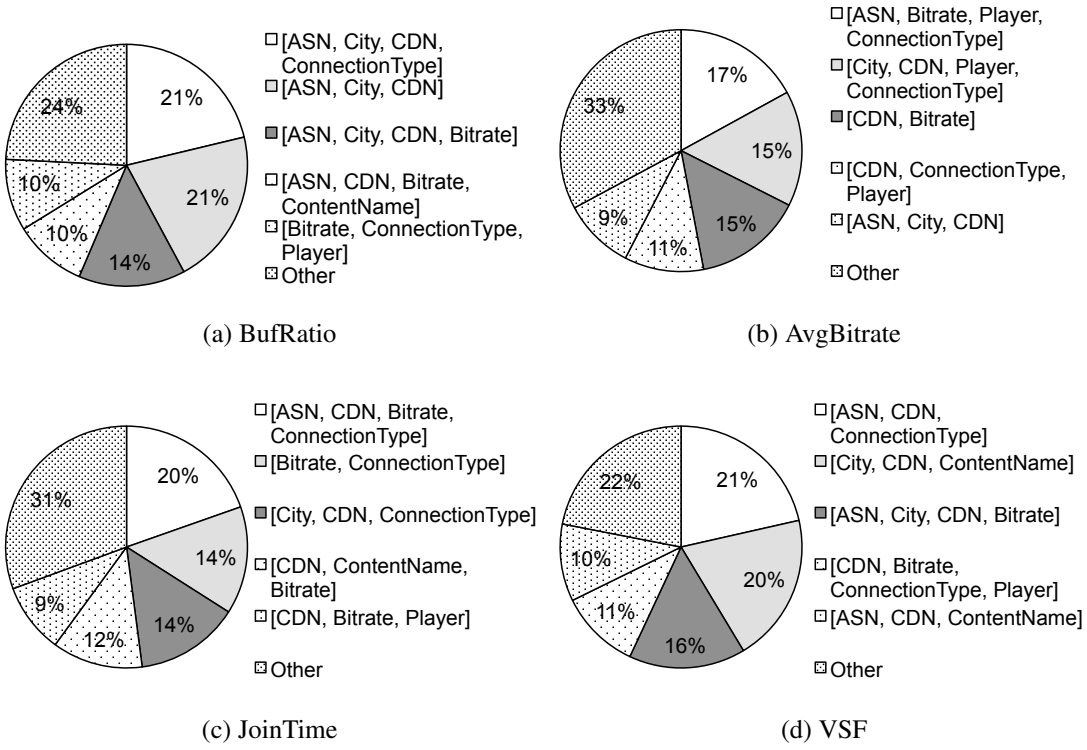


Figure 5.12: Analyzing the types of critical features: This shows a breakdown of the total number of sessions assigned to a specific type of critical features.

5.6.1 Types of Critical Features

Popular types of critical features: Figure 5.12 shows a breakdown of the fraction of sessions that are assigned to a specific type of critical feature set. We show this for different quality metrics. (Since we focus on a specific VoD provider, we do not consider the *Site* or *LiveOrVoD* for this analysis.) Across all quality metrics, the most popular critical features are *CDN*, *ASN* and *ConnectionType*, which means video quality is greatly impacted by network conditions at the server (*CDN*), transit network (*ASN*), and last-mile connection (*ConnectionType*).

We also see interesting patterns unique to individual metrics. *City* is among the top critical features of BufRatio. This is perhaps because network congestion usually depends on the volume of concurrent viewers in a specific region. *Bitrate* (initial bitrate) has a larger impact on AvgBitrate than on other metrics, since the videos in the dataset are mostly short content (2-5 minutes) and AvgBitrate is correlated with initial bitrate. Finally, *ContentName* has a relatively large impact on failures (VSF) but not other metrics, because VSF is sometimes due to the requested content not being ready.

Distribution of types of critical features: While the quality of about 50% of sessions is impacted by 3-4 popular types of critical features, 15% of sessions are impacted by a diverse set of more than 30 types of critical feature (not shown). This corroborates the need for expressive prediction models that handle the diverse factors affecting quality (Section 5.1.2).

	City	*ASN*	*Player*	*ConnectionType*
BufRatio	Some major east-coast cities			Satellite, Mobile, Cable
AvgBitrate		Cellular carriers	Players with different encodings	
JoinTime		Cellular carrier		Satellite, DSL
VSF		Small ISPs		Satellite, Mobile

Table 5.6: Analysis of the most prevalent values of critical features. A empty cell implies that we found no interesting values in this combination.

5.6.2 Values of Critical Features

Next, we focus on the most prevalent *feature values* (e.g., a specific ASN or player). To this end, we define *prevalence* of a feature value by the fraction of video sessions matching this feature value that have this feature as one of their critical features; e.g., the fraction of video sessions from Boston that have *City* as one of their critical features. If a feature value has a large prevalence, then the quality of many sessions that have this feature value can be explained by this feature.

We present the values of critical features with a prevalence higher than 50% for each quality metric and only consider a subset of the features (*ASN*, *City*, *ContentName*, *ConnectionType*) that appear prominently in Figure 5.12. We present this analysis with two caveats. First, due to proprietary concerns, we do not present the names of the entities, but focus on their characteristics. Second, we cannot confirm some of our hypothesis as it involves other providers; as such, we intend this result to be illustrative rather than conclusive.

Table 5.6 presents some anecdotal examples we observed. In terms of BufRatio, we see some of the major east coast cities (e.g., Boston, Baltimore) are more likely to be critical feature values than other smaller cities. We also see both poor (Satellite, Mobile) and broadband (Cable) connection types have high prevalence on BufRatio and JoinTime. This is because poor quality sessions are bottlenecked by poor connections, while some good quality sessions are explained by their broadband connections. "Player" has a relatively large prevalence on AvgBitrate, because the content provider uses different bitrate levels for different players (Flash or iOS). Finally, in terms of VSF, some small ISPs have large prevalence. We speculate that this is because their peering relationships with major CDNs are not provisioned, so their video sessions have relatively high failure rates.

5.7 Discussion

Relationship to existing ML techniques: CFA is a domain-specific prediction system that outperforms some canonical ML algorithms (Section 5.5.1). We put CFA in the context of three types of ML algorithms.

- *Multi-armed bandit* algorithms [215] find the decision with the highest reward (i.e., best CDN and bitrate) from multiple choices. They assume each decision has a fixed distribution

of rewards, but the video quality of a CDN also depends on client-side features. In contrast, contextual multi-armed bandit algorithms [189] assume the best decision depends on contextual information, but they require appropriate modeling between the context and decision space, to which critical features provide one viable approach.

- *The feature selection* problem [106] seems similar to critical feature learning, but with a key difference: critical features vary across video sessions. Thus, techniques looking for features that are most important for all sessions are not directly applicable.
- *Advanced ML* algorithms today can handle highly complex models [139, 182] efficiently, so in theory the critical features could be automatically identified, albeit in an implicit manner. CFA uses existing ML models (specifically, the "variable kernel conditional density estimation" method [206]) and may be less accurate than advanced ML techniques, but CFA can predict with more recent data since it tolerates stale update on the critical features. Furthermore, CFA is less opaque since it is based on domain-specific insights about critical features (Section 5.2).

Prediction and selection bias: One concern of making predictions and decisions based on quality measurements from clients is that they may be biased by previous decisions. For instance, if we move all video clients to the current optimal decision, we would not be able to predict quality on other decisions (i.e., a classical exploration-exploitation tradeoff). A simple solution to address the issue is to make random selection on a small portion (e.g., 10%) of clients.

Finer grain selection: Currently, CFA selects the resources at the CDN granularity. This means CFA cannot do much if the CDN redirects the client based on its location and the servers the CDN redirects the client to are congested. However, if the client were able to specify the server to stream from, we could avoid the overloaded servers and improve the quality.

Leveraging network and CDN information: CFA makes predictions and decisions based on client side information only. While clients provide accurate information regarding QoE, this information is not always optimal when making predictions and decisions. Prediction can be much more accurate if CFA were to leverage finer-grained information from other entities in the ecosystem, including servers, caches and network path.

Critical vs. minimal features: In general, critical features are not intended to be the *minimal* set of features that determines the quality. This makes critical features slightly less useful for root cause diagnosis. An interesting direction for future work is to incorporate notions of description length (MDL) into the learning process [175].

5.8 Related Work

Internet video optimization: There is a large literature on measuring video quality in the wild (e.g., content popularity [166, 223], quality issues [122] and server selection [197, 209]) and techniques to improve user experience (e.g., bitrate adaptation algorithms [115, 121, 225], CDN optimization and federation [50, 145, 156, 165] and cross-provider cooperation [99, 123, 226]). Our work builds on insight from the prior work (e.g., critical features of Section 5.2.2 are inspired by quality "bottleneck" in [122]). That said, the global optimization system in our work is an enhancement of these approaches as it uses the accurate prediction of CFA to make predictive

decisions. While a case for similar vision is made in [147], our work gives a systematic and practical algorithmic design.

Global coordination platform: Decision making based on a global view is similar to other logically centralized control systems (e.g., [101, 147, 200, 202]). They examined the architectural issues of decoupling control plane from data plane, including scalability (e.g., [87, 208]), fault tolerance (e.g., [161, 222]) and use of big data systems (e.g., [28, 101]). In contrast, our work offers concrete algorithmic techniques over such control platform [101] for video quality optimization.

Large-scale data analytics in system design: Many studies have applied data-driven and "Big Data" techniques to system problems, such as performance monitoring and diagnosis (e.g., [78, 177, 194]), revenue debugging (e.g., [60]), TCP throughput prediction (e.g., [111, 154]), and tuning TCP parameters (e.g., [179, 193]). Recent studies also try to operate these techniques at scale [81]. While CFA shares the data-driven approach, we exploit video-specific insights to achieve scalable and accurate prediction based on a global view of quality measurements, and we are not aware of prior publications on real-time predictive analytics at scale.

Relationship of CFA to ML techniques: CFA is a domain-specific prediction system that outperforms some canonical ML algorithms. Due to space limitations, we only highlight some salient points. In essence, CFA is an instance of a "variable kernel conditional density estimation" method [206]. It addresses the curse of dimensionality by contracting parts of the feature space that are not critical for prediction or in which there is too little available data.

QoE models: Prior work has shown correlations between various video quality metrics and user engagement (e.g., users are sensitive to BufRatio [88]), and built various QoE model (e.g., [41, 51, 135, 191]. Our work focuses on improving QoE by predicting individual quality metrics, and can be combined with these QoE models.

5.9 Summary

This chapter presents the first illustration of how DDN can be used to improves video QoE. In particular, we have formulated data-driven QoE optimization as a prediction problem, which as shown by prior work could lead to improved QoE. However, prior efforts failed to provide a prescriptive solution that (a) is expressive enough to tackle the complex feature-quality relationships observed in the wild and (b) can provide near real-time quality estimates. To this end, we developed CFA, a solution based on domain-specific insight of critical features (an illustration of the persistent critical structures) that video quality is typically determined by a subset of critical features which tend to be persistent. CFA leverages these insights to engineer an accurate algorithm that outperforms off-the-shelf machine learning approaches and lends itself to a scalable implementation that retains model freshness. Using real deployments and trace-driven analyses, we showed that CFA achieves up to 30% improvement in prediction accuracy and 12-32% improvement in QoE over alternative approaches. CFA leverages these insights to engineer an accurate algorithm that outperforms off-the-shelf machine learning approaches and lends itself to a scalable implementation that retains model freshness. Using real deployments and trace-driven analyses, we showed that CFA achieves up to 30% improvement in prediction accuracy and 12-32% improvement in QoE over alternative approaches.

Chapter 6

Cross-Session Throughput Prediction for Initial Video Bitrate Selection

In the previous chapter, we have demonstrated that if we observe the QoE of many video sessions, we can improve QoE by predicting QoE of any given video session based on the QoE of similar sessions. However, it is not always feasible for anyone to measure QoE directly, as it requires access to the internal states of the client-side applications. In the meantime, there are public datasets on throughput measurements, e.g., FCC Measuring BroadBand America Platform [14], which can be used as input to the QoE prediction system. To exploit this opportunity, we remove the assumption that QoE can be observed from video sessions from the CFA formulation, and pose the question that whether it is feasible to prediction end-to-end throughput of a video session based on the throughput observed from other HTTP sessions in third-party datasets.

The new objective of throughput prediction is particularly relevant to improving the video QoE of many content provide. Most video players today start streaming videos in a low bitrate and then gradually increase the bitrate using a local logic, which typically takes several seconds to tens of seconds before it reaches a high and sustainable bitrate (if it ever does before the session ends). In contrast, if the player can accurately predict throughput before a video session starts, it will be able to start streaming in the highest-yet-sustainable bitrate (Section 6.1).

To accurately predict end-to-end throughput, this chapter argues for a *cross-session* prediction approach inspired by DDN, where throughput measured on sessions of different servers and clients is used to predict the throughput of a new session. We observe substantial similarity among the throughput of similar sessions, but it is challenging to transform such similarity to accurate throughput prediction due to complex relations between session-level features and throughput. We develop an accurate throughput predictor called *DDA*, which combines the nearest neighbor model with the domain-specific insight that throughput only depends a subset of session-level features (Section 6.2). DDA is another illustration of persistent critical structures. Using two throughput measurement datasets, we show that DDA predicts throughput more accurately than simple predictors and conventional machine learning algorithms; e.g., DDA's 80%ile prediction error of DDA is $\geq 50\%$ lower than other algorithms. We also show that this improved accuracy enables video players to select a higher sustainable initial bitrate; e.g., compared to initial bitrate without prediction, DDA leads to $4\times$ higher average bitrate (Section 6.3).

Bitrate selection	Examples	Limitations	How throughput prediction helps
Fixed bitrate	NFL, Lynda	Too low bitrates	Higher bitrate with less re-buffering and lower start-up time
Adaptative bitrate	ESPN, Vevo, Netflix	Initial chunks are used to probe throughput	

Table 6.1: Limitations of today's video players and how they benefit from throughput prediction. www.lynda.com uses fixed bitrate of 520Kbps (360p) by default. Netflix (www.netflix.com/WiMovie/70136810?tr kid=439131) takes roughly 25 seconds to adapt from the initial bitrate (560Kbps) to the highest sustainable bitrate (3Mbps).

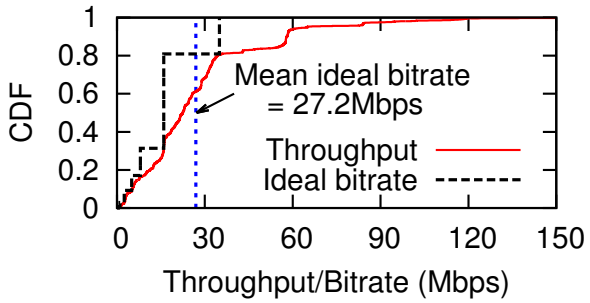


Figure 6.1: Distribution of throughput in the FCC dataset

6.1 Background

6.1.1 Today's Suboptimal Initial Bitrate Selection

A video player should ideally pick the highest initial bitrate that is sustainable (i.e., below the throughput), in order to ensure desired user experience of video streaming. Existing approaches to initial bitrate selection, however, are inefficient. Table 6.1 shows measured anecdotal evidence of such inefficiencies from several commercial providers. Fixed-bitrate players that use the same bitrate for the whole video session often intentionally use low bitrate to prevent mid-stream rebuffering (e.g., NFL, Lynda). Even if bitrate can be adapted midstream (e.g., [22, 121, 190]) the player often conservatively starts with a low bitrate and takes a significant time to reach the optimal bitrate (e.g., Netflix). Furthermore, for short video clips such adaptation may not reach the desired bitrate before the video finishes (e.g., Vevo music clips).

6.1.2 Dataset

We use two datasets of HTTP throughput measurement to evaluate DDA's performance: (i) a primary dataset collected by FCC's Measuring Broadband American Platform [14] in September 2013, and (ii) a supplementary dataset collected by a major VoD provider in China.

FCC dataset: This dataset consists of 9.9 million sessions and is collected from 6204 clients in US spanning 17 ISPs. In each test, a client set up an HTTP connection with one of the web servers for a fixed duration of 30 seconds and attempted to download as much of the payload as possible. It also recorded average throughput at 5 second intervals during the test. The test used

Feature	Description	# of unique values
ClientID	Unique ID associated to a client	6204
ISP	ISP of client (e.g., AT&T)	17
State	The US state where the client is located	52
Technology	The connection technology (e.g., DSL)	5
Target	The server-side identification	30
Downlink	Advertised download speed of the last connection (e.g., 15MB/s)	36
Uplink	Advertised upload speed of the last connection (e.g., 5MB/s)	25

Table 6.2: Basic statistics of the FCC dataset.

three concurrent TCP connections to ensure the line was saturated. Reader may refer to [15] for more details on the methodology.

Figure 6.1 shows the throughput distribution of all sessions. It also shows the distribution of ideal bitrate (i.e., highest bitrate chosen from {0.016, 0.4, 1.0, 2.5, 5.0, 8.0, 16.0, 35.0}Mbps[1] below the throughput). With perfect throughput prediction, we should be able to achieve average bitrate of 26.9Mbps with no session suffering from re-buffering. Compared to the fixed initial bitrate (e.g., 2.5Mbps) used today, this suggests a large room of improvement.

The clients represent a wide spatial coverage of ISPs, geo-locations, and connection technology (see Table 6.2). Although the number of targets are relatively small, the setting is very close to what real-world application providers face – the clients are widely distributed while the servers are relatively fewer. In addition, its measurement frequency (i.e., each client fetching content from each server once every hour) provides a unique opportunity to test the prediction algorithms' sensitivity to different measurement frequency. For instance, to emulate the effect of reduced data, we take one (the first) 5-second throughput sample from each test, and then randomly drop (e.g., 90% of) the available measurements to simulate a dataset where each client accesses a server less frequently (e.g., in average once every 10 hours).

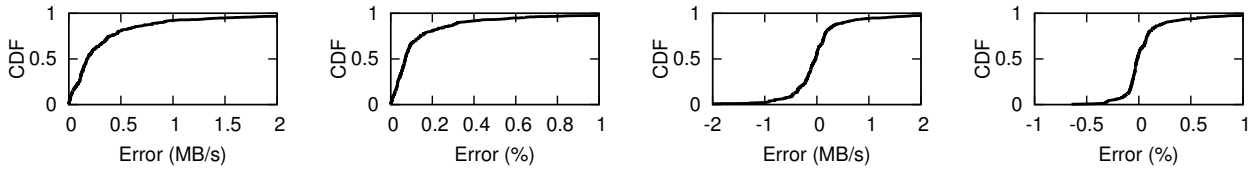
Supplementary VoD dataset: As a supplementary dataset, we use throughput dataset of 0.8 millions VoD sessions, collected by a major video content provider in China. Each video session has the average throughput and a set of features, that are different from the FCC dataset, including content name, user geolocation, user ID and server IP. This provides an opportunity to test the sensitivity of the algorithms to different sets of available features.

6.1.3 Limitations of Simple Predictors

This section starts by showing that simple predictors fail to yield desirable prediction accuracy, and then shows fundamental challenges of cross-session throughput prediction.

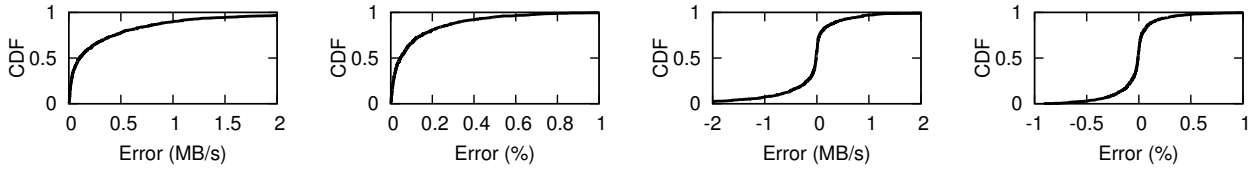
- First, we consider the **last-mile predictor**, which uses sessions with the same *downlink* feature (see definition in Table 6.2) to predict a new session's throughput. This is consistent

[1]The bitrates are recommended upload encoding by YouTube [34].



(a) Non-normalized absolute (b) Normalized absolute (c) Non-normalized signed (d) Normalized signed

Figure 6.2: Prediction error of the last-mile predictor



(a) Non-normalized absolute (b) Normalized absolute (c) Non-normalized signed (d) Normalized signed

Figure 6.3: Prediction error of last-sample predictor

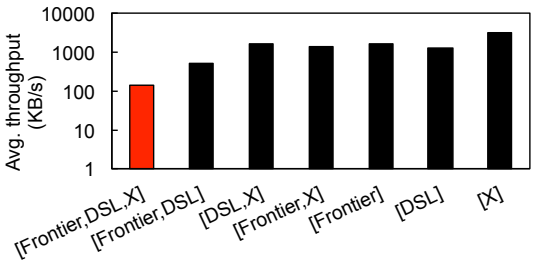
to the conventional belief that last-mile connection is usually the bottleneck. However, Figure 6.2a and 6.2b show substantial prediction error[2], especially on the tail where at least 20% of sessions have more than 20% error (Figure 6.2b). To put it into perspective, if a player chooses bitrate based on throughput prediction that is 20% higher or lower than the actual, the video session will experience mid-stream re-buffering or under-utilize the connection. Finally, Figure 6.2c and 6.2d show that the prediction error is two-sided, suggesting that simply adding or multiplying the prediction by a constant factor will not fix the high prediction error.

- Second, we consider the **last-sample predictor**, which uses the throughput of the last session of the same client-target pair to predict the throughput of a future session. However, the last-sample predictor is not reliable as the last sample is too sparse and noisy to offer reliable and accurate prediction. Figure 6.3 shows that, similar to the last-mile predictor, (i) the prediction error, especially on the tail, is not desirable – more than 25% of sessions have more than 20% normalized prediction error, and (ii) the prediction error is two-sided, suggesting the prediction error cannot be fixed by simply adding or multiplying the prediction with a constant factor.

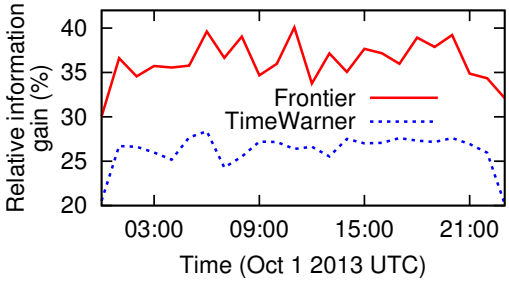
Challenges: The fundamental challenge to produce accurate prediction is the complex underlying interactions between session features and their throughput. In particular, there are two manifestations of such high complexity.

First, the simple predictors are both based on single feature (e.g., downlink or time), while combinations of multiple features often have a much greater impact on throughput than individual features. This can be intuitively explained as the throughput is often *simultaneously* affected

[2]Given throughput prediction p and actual throughput q, we define four types of prediction error: non-normalized absolute prediction error: $|p-q|$, normalized absolute prediction error: $\frac{|p-q|}{q}$, non-normalized signed prediction error: $p-q$, normalized signed prediction error: $\frac{p-q}{q}$.



(a) The average throughput of sessions matching all and a subset of three features: *ISP* = Frontier, *Technology* = DSL and *Target* = samknows1.lax9.level3.net (X). Time: 18:00-00:00 UTC, Oct 7, 2013



(b) The relative information gain of *Target* in two ISPs over time.

Figure 6.4: Two manifestations of the high complex interaction between session features and the throughput.

by multiple factors (e.g., the last-mile connection, server load, backbone network congestion, etc), and that means sessions sharing individual features may not have similar throughput. Figure 6.4a gives an example of the effect of feature combinations. It shows the average throughput of sessions of ISP Frontier using DSL fetching target samknows1.lax9.level3.net, and average throughput of sessions having same values on one or two of the three features. The average throughput when all three features are specified is at least 50% lower than any of other cases. Thus, to capture such effect, the prediction algorithm must be expressive to combine multiple features.

Second, the simple predictors both use same feature to all sessions, but the impact of same features on different sessions could be different. For instance, throughput is more sensitive to last-mile connection when it is unstable (e.g., Satellite), and it depends more to ISP during peak hours when the network tends to be the bottlenecks. Figure 6.4b shows a real-world example. Relative information gain[3] is often used to quantify how useful a feature is used for prediction. The figure shows the relative information gain of feature *Target* on the throughput of sessions in two ISPs over time. It shows that the impact of the same feature varies across sessions in different hours and in different ISPs.

We will see in Section 6.3 that due to the complex underlying interactions between features and throughput, it is non-trivial for conventional machine learning algorithms (e.g., decision tree, naive bayes) to yield high accuracy.

6.2 Design of DDA

In this section, we present the DDA approach that yields accurate throughput prediction (Section 6.3). We start with an intuitive description of DDA before formally describing the algorithm.

[3] $RIG(Y|X) = 1 - H(Y|X)/H(Y)$, where $H(Y)$ and $H(Y|X)$ are the entropy of Y and the average conditional entropy of Y [18].

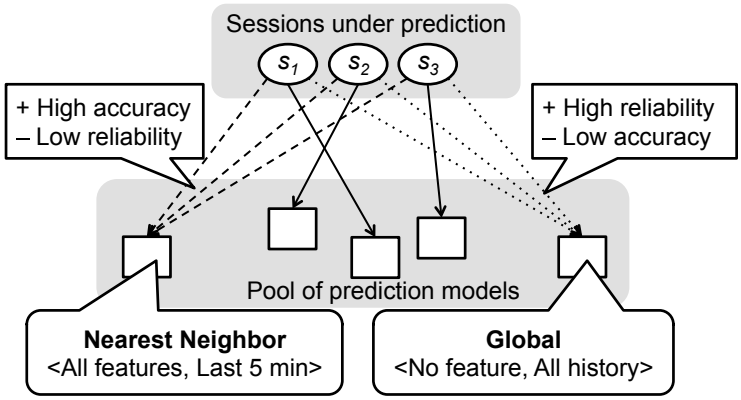


Figure 6.5: Mapping between sessions under prediction and prediction models.

6.2.1 Insight of DDA

At a high level, DDA finds for any session s a *prediction model* – a pair of features and time range, which is used to aggregate history sessions that match the specific features with s and happened in the specific range.

To motivate how DDA maps a session to a prediction model, let us consider two strawmen of session-model mapping shown in Figure 6.5. The first strawman maps each session s to the "Nearest Neighbor" prediction model (dash arrows), which aggregates only history sessions matching all features with s and happening in very short time (e.g., 5 minute) before s. Theoretically, "Nearest Neighbor" model should be highly accurate as it represents sessions that are the most similar to s, but history sessions meeting this requirement are too sparse to provide reliable prediction. Alternatively, one can map any s to the "Global" prediction model (dot arrows), which aggregates all history sessions regardless of their features or happening time. While "Global" model is highly reliable as it has substantial samples in history, the accuracy is low because it does not capture the effect of feature combination introduced in the last section.

Ideally, we would like achieve both high accuracy and high reliability. To this end, DDA (shown by solid arrows in Figure 6.5) differs from the above strawmen in two important aspects. First, DDA finds for a given session a prediction model between the Nearest Neighbor and Global prediction models, so that it strikes a balance between being closer to Nearest Neighbor for accuracy and being closer to Global for reliability. The resulting prediction model should be expressive (e.g., have more features) and yet have enough samples to offer a reliable prediction. Second, instead of mapping all sessions to the same prediction model, DDA maps different sessions to different prediction models, which allows DDA to address inherent heterogeneity that the same feature has different impact on different sessions.

6.2.2 Algorithm

Overall workflow: DDA uses two steps to predict the throughput of a new session s.

1. First, DDA learns a prediction model M_s^* based on history data. A prediction model is a pair of feature combination and time range.

2. Second, DDA estimates s's throughput by the median throughput of sessions in $Agg(M_s^*, s)$ that match s on the feaures of M_s^* and are in the time range of M_s^*. I.e., DDA's prediction is $Pred(s) = Median(Agg(M_s^*, s))$.

Learning of prediction model: First, DDA learns a prediction model M_s^* based on history data from a pool of all possible prediction models, i.e., pairs of all feature combinations (i.e., 2^n subsets of n features in Table 6.2) and possible time windows. Specifically, the possible time windows include time windows of certain history length (i.e., last 10 minutes to last 10 hours) and those of same time of day/week (i.e., same hour of day in the last 1-7 days or same hour of week in the last 1-3 weeks).

The objective of M_s^* is to minimize the prediction error, $Err(Pred(s), s_w) = \frac{|Pred(s) - s_w|}{s_w}$, where s_w is the actual throughput of s. That is,

$$M_s^* = \underset{M}{\operatorname{argmin}} Err(Median(Agg(M, s)), s_w) \tag{6.1}$$

Rather than solving Eq 6.1 analytically, DDA takes a *data-driven* approach and finds the best prediction model over a set of history sessions $Est(s)$ (defined shortly). Formally, the process can be written as following:

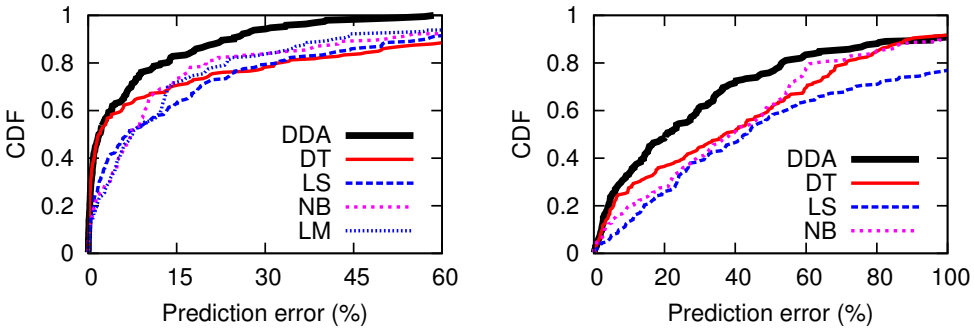
$$M_s^* = \underset{M}{\operatorname{argmin}} \frac{1}{|Est(s)|} \sum_{s' \in Est(s)} Err(Median(Agg(M, s')), s'_w) \tag{6.2}$$

$Est(s)$ should include sessions that are likely to share the best prediction model with s. In DDA, $Est(s)$ consists of sessions that match features *Target*, *ISP*, *Technology* and *Downlink* with s and happened within 4 hours before s.

Estimating throughput: Second, DDA estimates s's throughput by the learned prediction model M_s^*. To make the prediction $Pred(s)$ reliable, DDA ensures that $Pred(s)$ is based on a substential amount of sessions in $Agg(M_s^*, s)$. Therefore, if M_s^* yields $Agg(M_s^*, s)$ with less than 20 sessions, DDA will remove that model from the pool and learn the prediction model as in the first step again. We have also found that for some pairs of client and server, DDA's prediction error is one-sided. For instance, the throughput of a particular client-server pair is 1Mbps, while the best prediction model always predicts 2Mbps (i.e., a one-sided 100% error). We compensate this error by changing $Median(S)$ to $Median(S, k)$ which reports the median of throughput in S times a factor k. To train a proper value of k, DDA first uses Eq 6.2 to learn M_s^* by assuming $k = 1$, and then, DDA trains the best factor k_s^* for s as follows:

$$k_s^* = \underset{k}{\operatorname{argmin}} \frac{1}{|Est(s)|} \sum_{s' \in Est(s)} Err(Median(Agg(M_s^*, s'), k), s'_w)$$

where k is chosen from 0 to 5. Finally, the prediction made by DDA will be $Median(Agg(M_s^*, s), k_s^*)$.



(a) FCC (b) VoD China

Figure 6.6: CDF of prediction error.

6.3 Evaluation

This section evaluates the prediction accuracy of DDA (Section 6.3.1) and how much DDA improves video bitrate (Section 6.3.2). Overall, our findings show the following:

1. DDA can predict more accurately than other predictors.
2. With higher accuracy, DDA can select better bitrate.

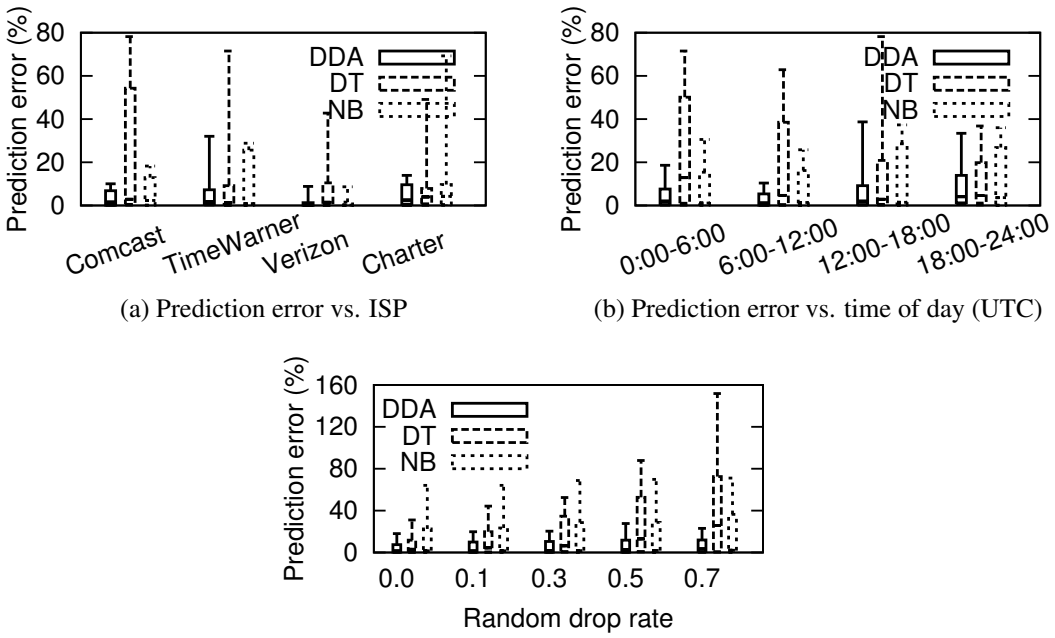
6.3.1 Prediction Accuracy

Methodology: As points of comparison, we use implementations of Decision Tree (DT) and Naive Bayes (NB) with default configurations in `weka`, a popular ML tool [30]. For a fair comparison, all algorithms use the same set of features. We also compare them with last-mile predictor (LM)[4] and last-sample predictor (LS), introduced in Section 6.1. We update the model of other algorithms in a same way as DDA: for each session under prediction, we use all available history data before it as the train data. Each session's timestamp is grouped into 10-minute intervals and used as discrete time feature. By default, we use absolute normalized error (Section 6.2) as the metric of prediction error, and the results are based on the FCC dataset, unless specified otherwise.

Distribution of prediction error: Figure 6.6 shows the distribution of prediction error of DDA and other algorithms. DDA outperforms all algorithms, especially on the tail of prediction error. For the FCC dataset (Figure 6.6a), 80%ile prediction error of DDA is 50% to 80% lower than that of other algorithms, and DDA has less than 20% sessions with more than 10% prediction error, while all other algorithms have at least 30% session with more than 10% error. While the VoD dataset in general has higher prediction error than the FCC dataset (due to the lack of some features such as last-connection and longitudinal information), DDA still outperforms other algorithms, showing that DDA is robust to the available features.

Dissecting prediction accuracy of DDA: To evaluate the prediction accuracy in more details, we first partition the prediction error by four most popular ISPs (6.7a) and by different time

[4]LM is not applicable to the VoD dataset as it has no feature related to last-mile connection.



(a) Prediction error vs. ISP

(b) Prediction error vs. time of day (UTC)

(c) Prediction error vs. random drop rate

Figure 6.7: Dissecting DDA prediction error. The boxes show the 10-20-50-80-90 percentile.

of day (6.7b). Although the ranking of algorithms varies across different partitions, DDA consistently outperforms other two algorithms (DT, NB), especially in the tail of 90%ile. Finally, Figure 6.7c evaluates DDA's sensitivity to measurement frequency by comparing the distribution of prediction error of three algorithms under different random drop rates (Section 6.1.2). It shows that DDA is more robust to measurement frequency than the other algorithms.

6.3.2 Improvement of Bitrate Selection

Methodology: To evaluate bitrate selected based on some prediction algorithm, we consider a simple bitrate selection algorithm (while a more complex algorithm is possible, it is not the focus of this paper): given a session of which the prediction algorithm predicts the throughput by w, the bitrate selection algorithm simply picks highest bitrate from $\{0.016, 0.4, 1.0, 2.5, 5.0, 8.0, 16.0, 35.0\}$Mbps [34] and below αw, where α represents the safety margin (e.g., higher α means higher bitrate at the risk of exceeding the throughput). We use two metrics to evaluate the performance: (1) AvgBitrate – average value of picked bitrate, and (2) GoodRatio – percentage of sessions with no re-buffering (i.e., picked bitrate is lower than the throughput). Therefore, one bitrate selection algorithm is better than another if it has both higher AvgBitrate and higher GoodRatio. As points of reference, "Global" bitrate selection algorithm picks the same bitrate for any session, which represents how today's players select starting bitrate. As a optimal reference point, "Ideal" bitrate selection algorithm picks the bitrate identical to the throughput for any session (Section 6.1.2).

Overall improvement: Table 6.3 compares DDA-based bitrate selection and the "Global". In

	FCC		VoD China	
	AvgBitrate	**GoodRatio**	**AvgBitrate**	**GoodRatio**
Global	2.5Mbps	88.2%	2.5Mbps	77.5%
DDA	13.3Mbps	99.5%	2.7Mbps	88.2%
Ideal	27.2Mbps	100%	3.5Mbps	100%

Table 6.3: Comparing DDA and "Global" in AvgBitrate and GoodRatio.

	Mean/median prediction error	**AvgBitrate**	**GoodRatio**
DDA	9.0%/2.3%	13.3Mbps	99.5%
DT	23.1%/3.4%	13.0Mbps	91.0%
LS	28.7%/9.8%	12.3Mbps	90.6%
NB	91.4%/17.1%	12.2Mbps	71.8%

Table 6.4: Higher accuracy means better bitrate selection.

both algorithms, we use $\alpha = 0.8$ for the FCC dataset, and $\alpha = 0.6$ for the VoD dataset. In both datasets, DDA leads to higher AvgBitrate and GoodRatio, and DDA is much closer to "Ideal" than "Global". Note that the VoD dataset still has a substantial room of improvement due to the relatively low prediction accuracy (Figure 6.6b).

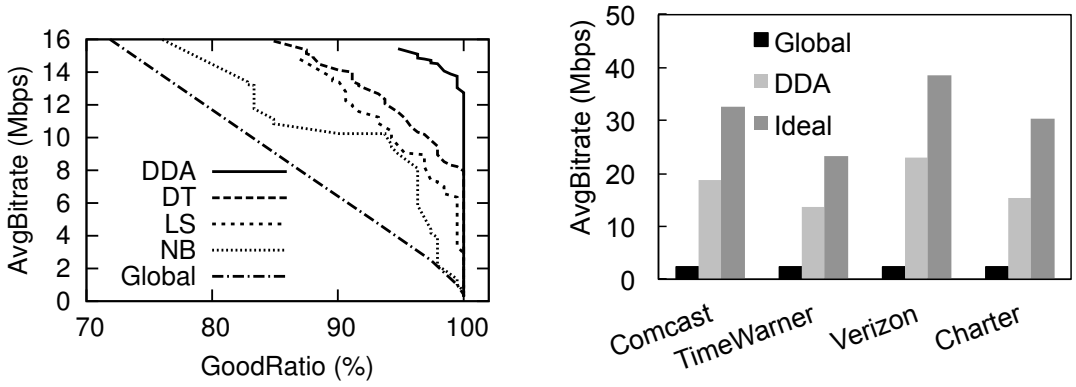
Bitrate selection vs. prediction accuracy: Next, we examine the intuition that higher prediction accuracy leads to higher performance of bitrate selection. Table 6.4 shows the bitrate selection performance as a function of median prediction error. We consider four prediction algorithms (DDA, DT, LS, NB). For a fair comparison, the bitrate selection algorithm always uses $\alpha = 0.8$. As prediction error increases, the performance of bitrate selection degrades in terms of both lower AvgBitrate and lower GoodRatio.

Understanding bitrate improvement: There is a natural tradeoff between AvgBitrate and GoodRatio (e.g., higher α means higher AvgBitrate at the cost of lower GoodRatio). Figure 6.8a shows such tradeoff of various bitrate selection algorithms by adjusting the value α. It is shown that DDA-based bitrate selection strikes a better tradeoff of higher AvgBitrate and higher GoodRatio (i.e., more towards the top-right corner of the figure).

Finally, we would like to test the robustness of DDA-based bitrate selection in different regions. Figure 6.8b compares the AvgBitrate of DDA with "Global" and "Ideal" in four popular ISPs. DDA uses the maximum α on the tradeoff curve in Figure 6.8a that ensures at least 95% GoodRatio, while "Global" only has GoodRatio of 88.2%. Across all ISPs, DDA consistently outperforms "Global" and achieve at least 60% of the "Ideal".

6.4 Related Work

At a high-level, our work is related to prior work in measuring Internet path properties, bandwidth measurements, and video-specific bitrate selection. With respect to prior measurement work, our key contribution is showing a practical data-driven approach for throughput prediction. In terms



(a) AvgBitrate-GoodRatio tradeoff

(b) Performance by ISP

Figure 6.8: In-depth analysis of bitrate selection

of video, our predictive approach offers a more systematic bitrate selection mechanism.

Measuring path properties: Studies on path properties have shown prevalence and persistence of network bottlenecks (e.g., [113]), constancy of various network metrics [230], longitudinal patterns of cellular performance (e.g., [159]), and spatial similarity of network performance (e.g., [53]). While DDA is inspired by these insights, it addresses a key gap because these efforts fall short of providing a prescriptive algorithm for throughout prediction.

Bandwidth measurement: Unlike prior "path mapping" efforts (e.g., [83, 148, 173, 193]), DDA uses a data-driven model based on available session features (e.g., ISP, device). Specifically, video measurements are taken within a constraint sandbox environment (e.g., browser) that do not offer interface for path information (e.g., traceroute). Other approaches use packet-level probing to estimate the end-to-end performance metrics (e.g., [112, 119, 169, 196]). Unlike DDA, these require additional measurement and often need full client/server-side control which is often infeasible in the wild. A third class of approaches leverages the history of the same client-server pair (e.g., [111, 120, 154, 201, 211]). However, they are less reliable when the available history of the same client and server is sparse.

Bitrate selection: Choosing high and sustainable bitrate is critical to video quality of experience [51]. Existing methods (e.g., [121, 153]) require either history measurement between the same client and server or the player to probe the server to predict the throughput. In contrast, DDA is able to predict throughput before a session starts. Other approaches include switching bitrate midstream (e.g., [115, 207, 224]) but do not focus on the initial bitrate problem which is the focus of DDA.

6.5 Summary

Inspired by the DDN paradigm, this chapter has presented DDA to demonstrate the feasibility of using throughput measurement on many HTTP sessions to accurately predict throughput and has applied DDA to initial video bitrate selection. DDA uses another illustration of the persistent critical structures that HTTP throughput depends on a subset of session-level features,

which persist on long timescales. Evaluation based on two real-world datasets shows (i) DDA predicts throughput more accurately than simple predictors and conventional machine learning algorithms, and (ii) with more accurate throughput prediction, a player can choose a higher-yet-sustainable bitrate (e.g., compared to initial bitrate without prediction, DDA leads to $4\times$ higher average bitrate with less sessions using bitrate exceeding the throughput).

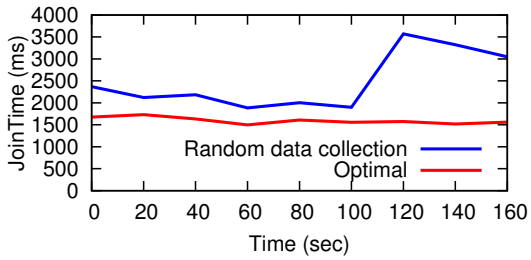
Chapter 7

Improving QoE via Exploration and Exploitation at Scale

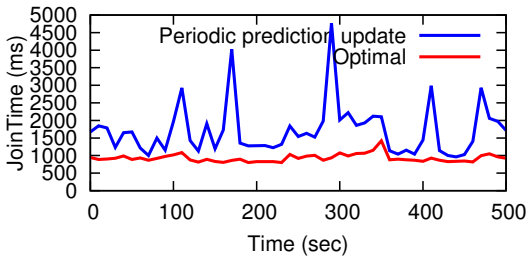
The previous two chapters have formulated the DDN process as a prediction problem, where we use observed QoE of recent sessions to build a QoE prediction system to inform the optimal selection of key configurations (e.g., server, bitrate, relay). While such prediction-based formulation has shown promising QoE improvements, it is necessarily incomplete as it: (1) suffers from many known biases (e.g., incomplete visibility) and (2) cannot respond to sudden changes (e.g., load changes). Drawing on a parallel from machine learning, we argue that data-driven QoE optimization should instead be cast as a real-time exploration and exploitation (E2) process rather than as a prediction problem. However, applying E2 in network applications, introduces architectural (e.g., how to update decisions in real time with fresh data) and algorithmic (e.g., capturing complex interactions between session features vs. QoE) challenges.

In this chapter, we present *Pytheas*, a control platform which addresses these challenges using a group-based E2 mechanism. Inspired by the insight of persistent critical structures, we observe that application sessions sharing the same features (e.g., IP prefix, location) can be grouped so that we can run E2 algorithms at a per-group granularity. This naturally captures the complex interactions and is amenable to realtime control with fresh measurements. Using an end-to-end implementation and a proof-of-concept deployment in CloudLab, we show that Pytheas improves video QoE over a state-of-the-art prediction-based system by up to 31% on average and 78% on 90th percentile of per-session QoE.

This chapter is organized as follows. We begin with the limitation of prediction-based workflow (Section 7.1) and why real-time E2 is a better abstraction for DDN (Section 7.2). Then we articulate key technical challenges for real-time E2, and outline the design rationale behind Pytheas (Section 7.3) to address the challenges. We then present more details of Pytheas on its algorithm design (Section 7.4), system design (Section 7.5), and implementation issues (Section 7.6). Finally, we evaluate the performance of Pytheas in Section 7.7, discuss related work in Section 7.8, and summarize the chapter in Section 7.9.



(a) Example A: Suboptimal quality due to fixed random data collection.



(b) Example B: Overload and oscillations between decisions due to periodic prediction.

Figure 7.1: Limitations of prediction-oriented abstraction (e.g., CFA [126]) manifested in two real examples.

7.1 Limitations of Predictive Approaches

Many prior approaches (e.g., [101, 126, 146, 199]) for data-driven QoE optimization use a *prediction-based* workflow. That is, they periodically train a quality prediction model based on passive measurements to inform decisions for future sessions; e.g., using history data to decide what will be the best relay server for a Skype call or the best CDN for a video session? While such prediction-based approaches have proved useful, they suffer from well-known limitations, namely, *prediction bias* and *slow reaction* [110, 193]. Next, we highlight these issues using CDN selection in video streaming as a concrete use case.

7.1.1 Limitation 1: Prediction Bias

A well-known problem of prediction-based workflows is that the prediction can be biased by prior decisions. Because the input measurement data are based on previous set of best decisions, we will not have a reliable way to estimate the potential quality improvements of other decisions in the future [193]. A simple solution is to use a fixed percentage of sessions to explore different decisions. This could eliminate the above prediction bias. However, it can still be suboptimal, since it might either let too many sessions use suboptimal decisions when quality is stable, or collect insufficient data in presence of higher variance.

Example A: Figure 7.1a shows a trace-driven evaluation to highlight such prediction biases. We use a trace of one of the major video providers in US. As a baseline, we consider prior work called CFA [126], which uses a fixed fraction of 10% sessions to randomly explore suboptimal decisions.[1] We see that it leads to worse average video startup latency, or join time, than an optimal strategy that always picks the CDN with the best average quality in each minute. Each video session can pick CDN1 or CDN2, and in the hindsight, CDN1 is on average better CDN2, except between t=40 and t=120, when CDN2 has a large variance. Even when CDN1 is consistently better than CDN2, CFA is worse than optimal, since it always assigns 10% of sessions to use CDN2. At the same time, when CDN2 becomes a better choice, CFA cannot detect this change

[1]The process begins by assigning sessions uniformly at random to all decisions in the first minute, and after that, it assigns 90% sessions to the optimal decisions based on the last minute.

in a timely fashion as 10% is too small a fraction to reliably estimate quality of CDN2.

7.1.2 Limitation 2: Slow Reaction

Due to the time taken to aggregate sufficient data for model building, today's prediction-based systems update quality predictions periodically on coarse timescales; e.g., CFA updates models every tens of seconds [126], and VIA updates its models every several hours [125]. This means that they cannot quickly adapt to changes in operating conditions which can cause *model drifts*. First, if there are sudden quality changes (e.g., network congestion and service outage), prediction-based approaches might result in suboptimal quality due to its slow reaction. Furthermore, such model shifts might indeed be a consequence of the slow periodic predictions; e.g., the best predicted server or CDN will receive more requests and its performance may degrade as its load increases.

Example B: We consider an AS and two CDNs. For each CDN, if it receives most sessions from the AS, it will be overloaded, and the sessions served by it will have bad quality. Figure 7.1b shows that CFA, which always picks the CDN that has the best quality in the last minute, has worse quality than another strategy which assigns half of sessions to each CDN. This is because CFA always overloads the CDN that has the best historical performance by assigning most sessions to it, and CFA will switch decisions only *after* quality degradation occurs, leading to oscillations and suboptimal quality.

At a high level, these limitations of prediction-based approaches arise from the logical separation between measurement collection and decision making. Next, we discuss what the right abstraction for data-driven QoE optimization should be to avoid these limitations.

7.2 Casting QoE Optimization as a Exploration-Exploitation Process

To avoid these aforementioned limitations of prediction-based approaches, ideally we want a framework where decisions are updated in concert with measurement collection in real time. There is indeed a well-known abstraction in the machine learning community that captures this—exploration and exploitation (E2) processes [215]. Drawing on this parallel, we argue why data-driven QoE optimization should be cast instead as a *real-time E2* process rather than a prediction-based workflow.

Background on exploration and exploitation: An intuitive way to visualize the exploration and exploitation (E2) process is through the lens of a multi-armed bandit problem [215]. Here, a gambler pulls several slot machines, each associated with an unknown reward distribution. The goal of the gambler is to optimize the mean rewards over a sequence of pulls. Thus, there is some intrinsic *exploration* phase where the gambler tries to learn these hidden reward functions, and subsequent *exploitation* phase to maximize the reward. Note that the reward functions could change over time, and thus this is a continuous process rather than a one-time shot.

QoE optimization as E2 (Figure 7.2): Given this framework, we can see a natural mapping between E2 and data-driven QoE optimization. Like E2, data-driven QoE optimization observes

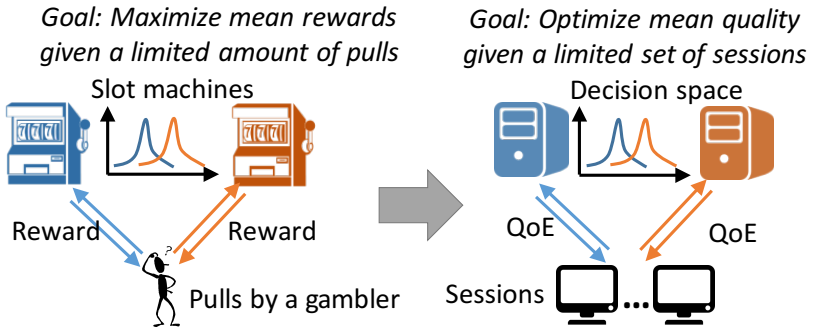


Figure 7.2: Casting data-driven QoE optimization into formulation of exploration and exploitation (E2).

the QoE (i.e., reward) of a decision every time the decision is used (i.e., pulled) by a session. Our goal is to maximize the overall QoE after a sequence of sessions.

Casting data-driven optimization as E2 not only provides a systematic framework for data-driven QoE optimization, but also allows us to leverage well-known algorithms (e.g., [48]) from the machine learning literature. As E2 integrates the measurement collection (exploration) and decision making (exploitation) in a joint process, we can dynamically explore decisions whose quality estimation has high uncertainty, and exploit decisions that are clearly better. For instance, in Example A, an E2 process could reduce traffic for exploration when QoE is stable (before 40 second and after 120 seconds), and raise it when QoE changes (between 40 second and 120 second). By running E2 in real time with the most up-to-date measurement data, we could detect QoE drift as soon as some sessions have experienced them, and adapt to them faster than prediction-based approaches. For instance, in Example B, real-time E2 would detect load-induced QoE degradation on CDN1 as soon as its QoE is worse than CDN2, and start switching sessions to CDN2 *before* overloading CDN1.[2]

7.2.1 Challenges of E2 in the Networking Context

While E2 offers the right abstraction in contrast to prediction-based approaches, applying it in network applications raises practical challenges:

- Traditional E2 techniques (e.g., [81, 141, 215]) need fresh measurements of all sessions, but getting such a fresh and global view is challenging, because application providers store fresh data in geo-distributed clusters, called *frontend clusters*, which only have a partial view across sessions. Existing analytics framework for such geo-distributed infrastructure, however, either trade global view for data freshness (e.g., [101]), or target query patterns of a traditional database (e.g., [170]), not millions of concurrent queries from geo-distributed clients, as in our case.

[2]Note that we do not need to know the capacity of each CDN, which is often unknown to content providers.

- Traditional E2 techniques also make strong assumptions about the context that affects the reward of a decisions, but they may not hold in network settings. For instance, they often assume some notion of continuity in context (e.g., [189]), but even when some video sessions match on all aspects of ISP, location, CDN resource availability, they may still see very different QoE, if they differ on certain key feature (e.g., last-hop connection) [126].

7.3 Overview of Pytheas Ideas

To address the practical challenges of applying E2 in network applications, we observe a key domain-specific insight in networked applications that enables us to address both challenges in practice. We highlight the intuition behind our insight, which we refer to as group-based E2, and then provide an overview of the Pytheas system which builds on this insight.

Insight of Group-based E2: Our insight is that the "network context" of application sessions is often aligned with their "network locality". That is, if two sessions share the context that determines their E2 decisions, they will be likely to match on some network-specific features. We see manifestations of this insight in many settings. For instance, video sessions with similar QoE from the same CDN/server tend to match on client IP prefix [126, 199]. Similarly, VoIP calls between the same ASes are likely to share the best relays [125], and clients from same /24 IP prefix will have similar web load time from the same edge proxy [146]. In Section7.5.2, we validate this insight with a real-world dataset.

This insight inspires the notion of *group-based E2*, which can address the above challenges by enabling an effective decomposition of the E2 process (Figure 7.3a). Specifically, instead of a global E2 process over all sessions, we group together sessions with similar context by network locality and other key features (such as device and location), and use one E2 process for each group. Since sessions within a group share network locality (e.g., in the same locations and IP prefixes), they are likely to be mapped to the same frontend cluster. By running the per-group E2 logic in this frontend cluster, we can update decisions with fresh data from other sessions in the group received by this frontend cluster. Furthermore, as each group consists of sessions with similar context, it is sufficient to use traditional E2 techniques based on the data of sessions in one group, without needing a global view of all sessions. It is important to note that sessions are not grouped entirely based on IP prefixes. The sessions in the same network locality could have very different QoE, depending on the device, last-hop connectivity, and other features. Therefore, we group sessions on a finer granularity than IP prefix.

System overview: Figure 7.3b shows how the group-based E2 is realized in the Pytheas architecture. Each session group is managed by one per-group E2 process run by one frontend cluster. When a session comes in, it sends a request for its control decisions, which includes its features, to the Pytheas system. The request will be received by a frontend cluster, which maps the session to a group based on its features, then gets the most up-to-date decision from the local per-group E2 process, and returns the decision to the session. Each session measures its QoE and reports it to the same frontend cluster. When this frontend receives the QoE measurement, it again maps the session to a group, and updates the E2 logic of the group with the new measurement. In most cases, the E2 logic of a group is run by the same cluster that receives the requests and measurements of its sessions, so E2 logic can be updated in real time.

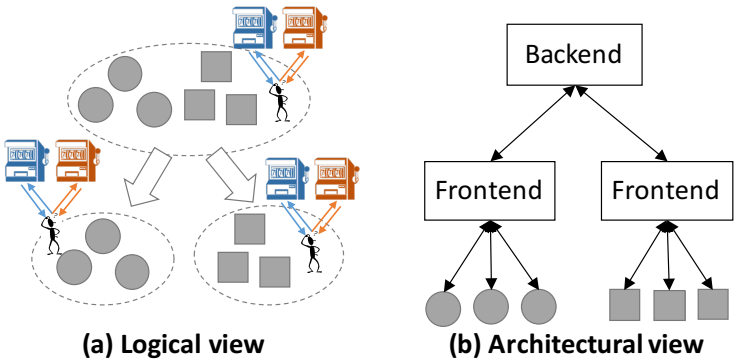


Figure 7.3: Illustration of group-based E2.

The backend cluster has a global, but slightly stale, view of QoE of all sessions, and it determines the session groups – which group each session belongs to and which frontend should run the per-group logic for each group. Normally, such grouping is updated periodically on a timescale of minutes. During sudden changes such as frontend cluster failures, it can also be triggered on demand to re-assign groups to frontend clusters.

The following sections will present the algorithms (Section 7.4) and system design (Section 7.5) of Pytheas, and how we implemented it (Section 7.6) in more details.

7.4 Pytheas Algorithms

Using group-based E2, Pytheas decouples real-time E2 into two parts: a *session-grouping* logic to partition sessions into groups, and a *per-group E2* logic that makes per-session decisions. This section presents the design of these two core algorithmic pieces and how we address two issues:[3] (i) Grouping drift: the session-grouping logic should dynamically regroup sessions based on the context that determines their QoE; and (ii) QoE drift: the per-group control logic should switch decisions when QoE of some decisions change.

7.4.1 Session-Grouping Logic

Recall that sessions of the same group share the same factors on which their QoE and best decisions depend. As a concrete example, let us consider CDN selection for video. Video sessions in the same AS whose QoE depends on the local servers of different CDNs should be in the same group. However, video sessions whose QoE is bottlenecked by home wireless and thus is independent to CDNs should not be in the same group. In other words, sessions in the same group share not only the best decision, but also the factors that determine the best decisions.

A natural starting point for this grouping decision is using the notion of critical features proposed in prior work [126]. At a high level, if session A and B have the same values of critical features, they will have similar QoE. Let $S(s,F,\Delta)$ denote the set of sessions that occur within

[3]We assume in this section that the per-group control logic is updated in real time (which will be made possible in the next section).

the last Δ time interval and share the same feature values as s on the set of features F, and let $Q(X)$ denote the QoE distribution of a session set X. Then, the critical feature set F^* of a session s:

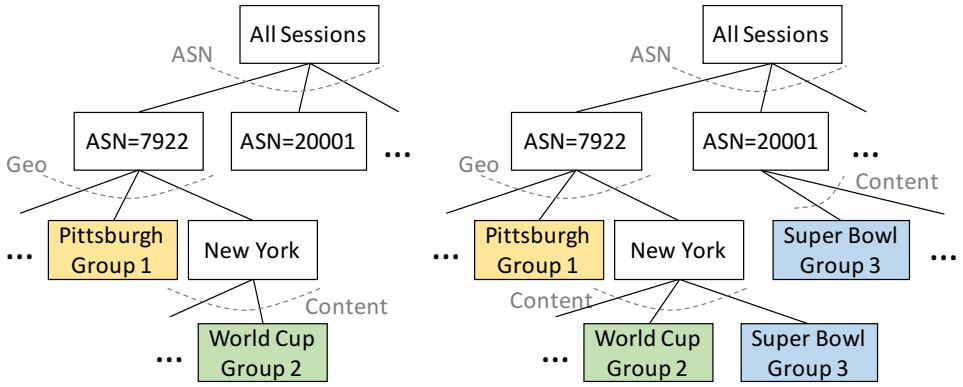
$$\text{argmin}_{F \subseteq F^{all}, |S(s,F,\delta)|>n} |Q(S(s,F^{all},\Delta)) - Q(S(s,F,\Delta))|$$

That is, the historical session who match values on critical features F^* with s have very similar QoE distribution to those matching on all features with s on a long timescale of Δ (say last hour). The clause $|S(s,F,\delta)| > n$ ensures that there is sufficient mass in that set to get a statistically significant estimate even on small timescales δ (e.g., minutes). Such a notion of critical features has also been (implicitly) used in many other applications; e.g., AS pairs in VoIP [125] and /24 prefixes for web page load time [146]. Thus, a natural strawman for grouping algorithm is to groups sessions who match on their critical features, i.e., they have similar QoE.

However, we observe two problems inherent to critical features, which make it unsuitable to directly group sessions based on critical features: (1) First, grouping sessions based on critical features may result in groups that consist of only sessions using similar decisions, so their measurement will be biased towards a subset of decisions. (2) Second, grouping sessions based on critical features will also create overlaps between groups, so E2 logic of different groups could make conflicting decisions on these overlapping sessions. For instance, consider two Comcast sessions, s_1 and s_2, if the critical feature of s_1 is ISP, and the critical feature of s_2 is its local WiFi connection, s_2 will be in both the "WiFi" group and the "Comcast" group.

To address these issues, we formulate the goal of session grouping as following. Given a session set, the session-grouping logic should output any non-overlapping partition of sessions so that if two sessions s_1 and s_2 are in the same group, s_1 and s_2 should match values on s_1 or s_2's non-decision-specific critical features. Non-decision-specific features are the features independent of decisions; e.g., "device" is a feature independent of decisions, since video sessions of the same device can make any decisions regarding CDN and bitrate.

Operationally, we use the following approach to achieve such a grouping. First, for each session, we learn its critical features, and then ignore decision-specific features from the set of critical features of each session. Then, we recursively group sessions based on the remaining critical features in a way that avoids overlaps between groups. We start with any session s_1, and create a group consisting of all sessions that match with s_1 on s_1's critical features. We then recursively do the two following steps until every session is in some group. We find a session s_2, who is not included in any existing group, and create a new group of all sessions that match with s_2 on s_2's critical features. If the new group does not overlap with any existing group, it will be a new individual group, otherwise, we will add it to the existing groups in the way illustrated in Figure 7.4. We organize the existing groups in a graph, where each node is split by values of a certain feature, and each group includes multiple leaf nodes. For instance, if we want to add a new group that consists of sessions whose "content" is "Super Bowl" to a graph of existing groups as shown in Figure 7.4a, we will fork a path to create a new leaf node whenever the new group overlap with a existing group. Note that, this means multiple leaf nodes may be belong to the same group (e.g., "Group 3" in Figure 7.4b contains two different leaf nodes).



(a) Existing groups organized in a graph

(b) After insertion of a group with critical feature {Content="Super Bowl"}

Figure 7.4: An illustrative example of session groups organized in a graph and how to a new group is added.

7.4.2 Per-Group E2 Logic

To run E2 in presence of QoE drift, we use Discounted UCB algorithm [102], a variant of the UCB algorithm [48], as the per-group E2 logic. UCB (Upper Confidence Bound) algorithms [48] are a family of algorithms to solve the multi-armed bandits problem. The core idea is to always opportunistically choose the arm that has the highest upper confidence bound of reward, and therefore, it will naturally tend to use arms with high expected rewards or high uncertainty. Note that the UCB algorithms do not explicitly assign sessions for "exploration" and "exploitation".

We use Discounted UCB algorithm to adapt to QoE drift, because it automatically gives more weight to more recent measurements by exponentially discounting historical measurements. Therefore, unlike other UCB algorithms which will (almost) converge to one decision, Discounted UCB is more likely to revisit suboptimal decisions to retain visibility across all decisions. We refer readers to [102] for more details. Given a session s, it returns a decision that has not been tried, if there is any. Otherwise, it calculates a score for each potential decision d by adding up an exponentially weighted moving average of d's history QoE and an estimation on the uncertainty of reward of d, and picks the decision with highest score.

7.5 Pytheas System Architecture

Given the algorithmic pieces from the previous section, next we discuss how we map them into a system architecture. At a high level, the E2 logic of each group is independently run by frontend clusters, while the session-to-group mapping is continuously updated by the backend.

7.5.1 Requirements

The Pytheas system design must meet four goals:

1. *Fresh data:* The per-group E2 logic should be updated every second with newest QoE measurements.
2. *Global scale:* It should handle millions of geo-distributed sessions per second.
3. *Responsiveness:* It should respond to requests for decisions from sessions within a few milliseconds.
4. *Fault tolerance:* QoE should not be significantly impacted when parts of the system fail.

A natural starting point to achieve this goals might be to adopt a "split control plane" approach advocated by prior work for prediction-based approaches [101, 126]. At a high level, this split control plane has two parts: (1) a backend cluster that generates centralized predictions based on global but stale data, and (2) a set of geodistributed frontend servers that use these predictions from the backend to make decisions on a per-session basis. This split control architecture achieves global scale and high responsiveness, but fundamentally sacrifices data freshness.

Pytheas preserves the scale and responsiveness of the split control approach, but extends in two key ways to run group-based E2 with fresh data. First, each frontend cluster runs an active E2 algorithm rather than merely executing the (stale) prediction decisions as in prior work. Second, the frontend clusters now run per-group logic, not per-session logic. This is inspired by the insight that sessions in the same group are very likely to be received by the same frontend cluster. Thus, group-based E2 could achieve high data freshness on the session group granularity, while having the same scale and responsiveness to split control. Next, we discuss the detailed design of the frontend and backend systems.

7.5.2 Per-Group Control by Frontends

The best case for group-based E2 is when all sessions of the same group are received by the same frontend cluster. When this is true, we can run the per-group E2 logic (Section 7.4.2) in real time with fresh measurements of the same group. In fact, this also is the common case. To show this, we ran session-grouping logic (Section 7.4.1) on 8.5 million video sessions in a real-world trace, and found around 200 groups each minute. Among these groups, we found that (in Figure 7.5) for 95% of groups, all sessions are in the same AS, and for 88% of groups, all sessions are even in the same AS *and* same city. Since existing session-to-frontend mappings (e.g., DNS or Anycast-based mechnisms) are often based on AS and geographical location, this means that for most groups, their sessions will be verly likely to be received in the same frontend clusters.

In practice, however, it is possible that sessions of one group are spread across frontend clusters. We have two options in this case:

1. Pick one cluster as the *leader* cluster of this group and let it run the E2 logic of the group based on the measurements received by this cluster. Meanwhile, other clusters, called *proxy* clusters of the group, simply receive decisions periodically from the leader cluster.
2. Keep the leader cluster and proxy clusters, but let proxy clusters not only receive decisions from the leader, but also forward QoE measurements to the leader cluster.

We see a tradeoff between the two options. While Option 1 is less complex to implement than Option 2, the leader proxy in Option 1 runs per-group logic based on only a subset of sessions, especially when the sessions of a group are evenly spread across many frontend clusters. We

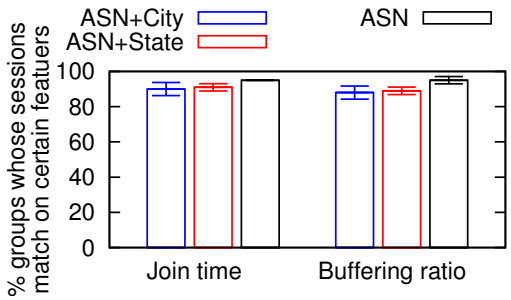


Figure 7.5: For most groups, the sessions are in the same ASN and even same city.

pick Option 2, because it is cleaner in that the per-group logic is based on all sessions in a group. In fact, implementing Option 2 does not add much complexity. Finally, Option 2 can easily fall back to Option 1 by stop forwarding measurements from proxy clusters to the leader cluster.

7.5.3 Updating Session Groups in the Backend

The backend cluster uses a global, stale view of measurements to update two tables, which are sent to the frontend to regroup sessions.

- First, the backend runs the session-grouping logic (Section 7.4.1) to decide which group each session belongs to, and outputs a *session-to-group table*.
- Second, it decides which frontend should be the leader cluster of each group and outputs a *group-to-leader table*. For each group, we select the frontend cluster that receives most sessions in the group as the leader.

The backend periodically (by default, every ten minutes) updates the frontend clusters with these two maps. The only exception for the maps to be updated in near real time is when one or more frontend clusters fail, which we discuss next.

7.5.4 Fault Tolerance

As we rely on fault-tolerant components for the individual components of Pytheas within each cluster (see Section 7.6), the residual failure mode of Pytheas is when some clusters are not available. Next, we discuss how we tackle three potential concerns in this setting.

First, if a failed frontend is the leader cluster of a group, the states of the E2 logic of the group will be lost, and we will not be able to update decisions for sessions of the group. To detect frontend failures and minimize their impact, each frontend sends frequent heartbeat messages through a "fast channel" every few seconds (by default, every five seconds) to the backend, so backend can detect frontend failures based on these heartbeat messages. Once the backend detects a frontend failure, it will select a new leader clusters for any group whose leader cluster has been the failed one, and recover the per-group logic in the new leader cluster. To recover the per-group states, each leader always shares the per-group states with its proxy clusters in the decision update messages, so that when a proxy cluster is selected as the new leader, it can recover the per-group states as they are cached locally. Note that even without a leader cluster,

a proxy cluster can still respond requests with the cached decisions made by the leader cluster before it fails.

Second, the sessions who are assigned to the failed frontend will not receive control decisions. To minimize this impact, Pytheas will fall back to the native control logic. Take video streaming as an example, when Pytheas is not available, the client-side video player can fall back to the control logic built into the client-side application (e.g., local bitrate adaptation) to achieve graceful QoE degradation, rather than crash [101].

Finally, if the backend cluster is not available, Pytheas will not be able to update groups. However, Pytheas does not rely on backend to makde decisions, so clients will still receive (albeit suboptimal) decisions made by Pytheas's frontend clusters.

7.6 Implementation and Optimization

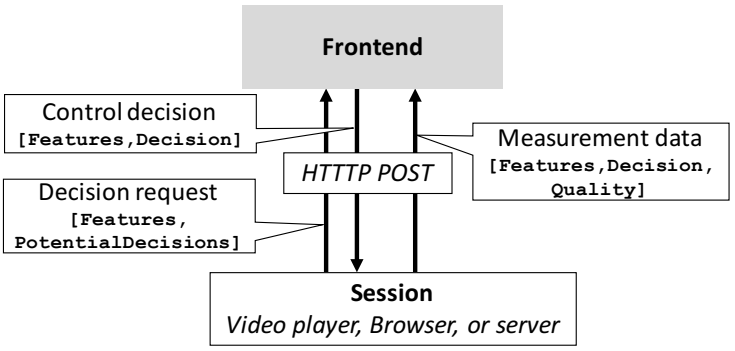
Pytheas is open source ($\approx$ 10K lines of code across Java, python, and PHP) and can be accessed at [13]. Next, we describe the APIs for applications to integrate with Pytheas, and then describe the implementation of frontend and backend, as well as optimizations we used to remove Pytheas's performance bottlenecks.

Pytheas APIs: Application sessions communicate with Pytheas through two APIs (Figure 7.6b): One for requesting control decisions, one for uploading measurement data. Both are implemented as standard HTTP POST messages. The session features are encoded in the data field of the POST message. Pytheas also needs content providers to provide the schema of the message that sessions send to the Pytheas frontend, as well as a list of potential decisions. Content providers may also provide QoE models that compute QoE metrics from the raw quality measurements sent by sessions.

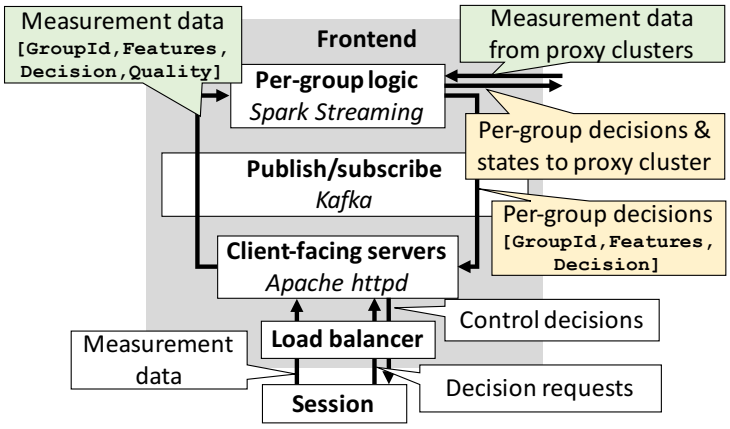
Frontend: Figure 7.6b shows the key components and interfaces of a frontend cluster. When a session sends a control request to Pytheas, the request will be received by one of the client-facing servers run by Apache httpd [17]. The server processes the request with a PHP script, which first maps the session to a group and its leader cluster by matching the session features with the session-to-group and group-to-leader tables. Then the server queries the E2 logic of the group (Section 7.4.2), for the most up-to-date decision of the group, and finally returns the decision to the session. The script to process the measurement data uploaded by a session is similar; a client-facing server maps it to a group, and then forwards it to the per-group E2 logic. The per-group E2 logic is a Spark Streaming [29] program. It maintains a key-value map (a Spark RDD) between group identification and the per-group E2 states (e.g., most recent decisions), and updates the states every second by the most recent measurement data in one MapReduce operation. The communication between these processes is through Kafka [21], a distributed publish/subscribe service. We used Apache httpd, Spark Streaming, and Kafka, mainly because they are horizontally scalable and highly resilient to failures of individual machines.

While the above implementation is functionally sufficient, we observed that the frontend throughput if implemented as-is is low. Next, we discuss the optimizations to overcome the performance bottlenecks.

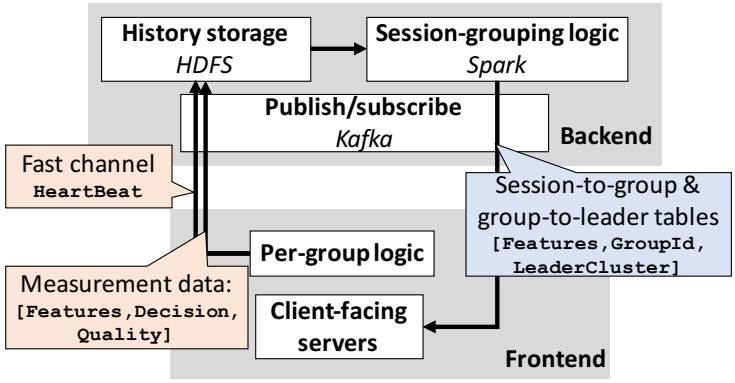
- *Separating logic from client-facing servers:* When a client-facing server queries the per-group control logic, the client-facing server is effectively blocked, which significantly re-



(a) API to application sessions



(b) Frontend



(c) Backend

Figure 7.6: Key components and interfaces of Pytheas implementation.

duces the throughput of client-facing servers. To remove this bottleneck, we add an intermediate process in each client-facing server to decouple querying control logic from responding requests. It frequently (by default every half second) pulls the fresh decision of each group from per-group logic and writes the decision in a local file of the client-facing server. Thus, the client-facing server can find the most up-to-date decisions from local cache without directly querying the control logic.

- *Replacing features with group identifier:* We found that when the number of session features increases, Spark Streaming has significantly lower throughput as it takes too long to copy the new measurement data from client-facing servers to Kafka and from Kafka to RDDs of Spark Streaming. This is avoidable, because once the features are mapped to a group by the client-facing servers, the remaining operations (updating and querying per-group logic) are completely feature-agnostic, so we can use group ID as the group identifier and remove all features from messages.

Backend: Figure 7.6c shows the key components and interfaces of the backend cluster. Once client-facing servers receive the measurement data from sessions, they will forward the measurement data to the backend cluster. On receiving these measurement data, the backend stores them in an HDFS for history data, and periodically (by default every 10 minutes) runs the session-grouping logic (Section 7.4.1) as a Spark [28] job to learn the session-group mapping and group-cluster mapping from the stored history data. These tables are sent to each frontend through Kafka, so that the future messages (requests and measurement data) from sessions will be matched against new tables. In addition, to detect failures of frontend clusters, each frontend cluster sends a small heartbeat message to the backend cluster every 5 seconds.

7.7 Evaluation

To evaluate Pytheas, we run our prototype [13] across multiple instances in CloudLab [8]. Each instance is a physical machine that has 8 cores (2.4 GHz) and 64GB RAM. These instances are grouped to form two frontend clusters and one backend cluster (each includes 5 to 35 instances). This testbed is an end-to-end implementation of Pytheas described in Section 7.6[4].

By running trace-driven evaluation and microbenchmarks on this testbed deployment, we show that:

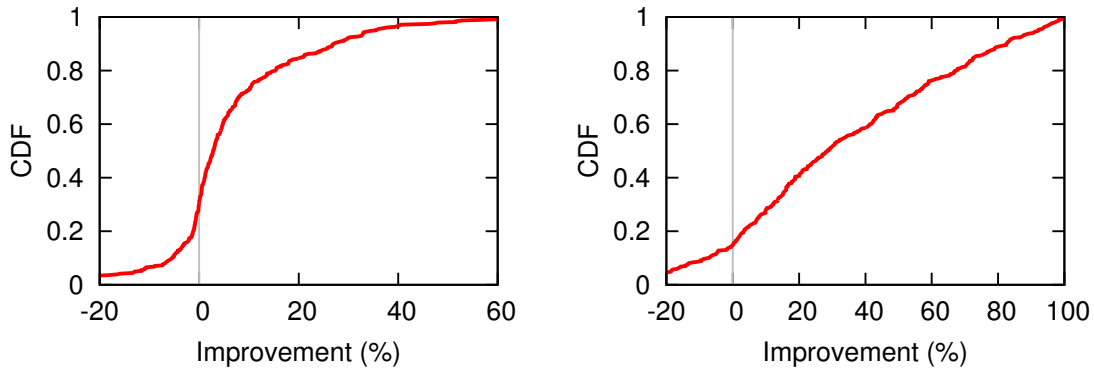
- In the use case of video streaming, Pytheas improves the mean QoE by up to 6-31% and the 90th percentile QoE by 24-78%, compared to a prediction-based baseline (Section 7.7.1).
- Pytheas is horizontally scalable and has similar low response delay to existing prediction-based systems (Section 7.7.2).
- Pytheas can tolerate failures on frontend clusters by letting clients fall back to local logic and rapidly recovering lost states from other frontends (Section 7.7.3).

7.7.1 End-to-End Evaluation

Methodology: To demonstrate the benefit of Pytheas on improving QoE, we use a real-world trace of 8.5 million video sessions collected from a major video streaming sites in US over a 24-hour period. Each video session can choose one of two CDNs. The sessions are replayed in the same chronological order as in the trace. We call a group of sessions a *unit* if they match values on AS, city, connection type, player type and content name.[5] We assume that when a

[4]Pytheas can use standard solutions such as DNS redirection to map clients to frontend clusters, and existing load balancing mechanisms provided by the host cloud service to select a frontend server instance for each client.

[5]The notion of unit is used to ensure statistical confidence of QoE evaluation, and is not used in Pytheas.



(a) Join time (b) Buffering ratio

Figure 7.7: Distribution of improvement of Pytheas over the prediction-based baseline.

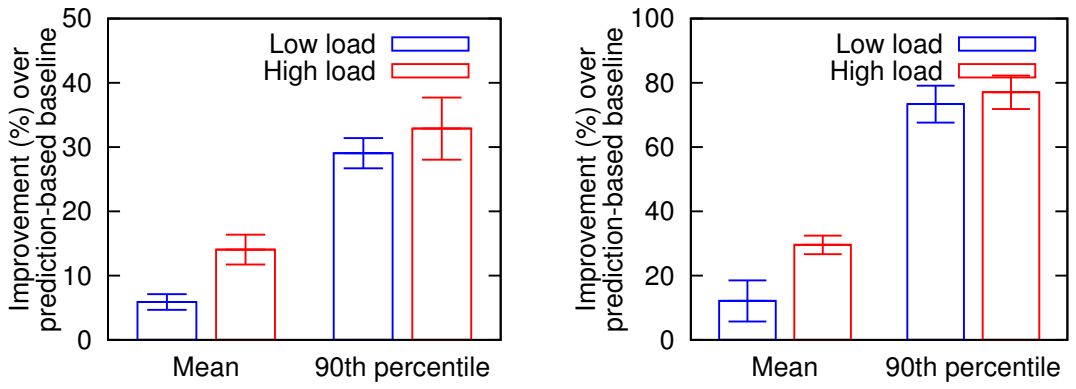
video session is assigned to a CDN, its QoE would be the same to the QoE of a session that is randomly sampled from the same unit who use the same CDN in the same one-minute time window. For statistical confidence, we focus on the units which have at least 10 sessions on each CDN in each one-minute time windows for at least ten minutes. We acknowledge that our trace-driven evaluation has constraints similar to the related work, such as the assumption that QoE in a small time window is relatively stable in each unit (e.g., [125]) and a small decision space (e.g., [126]).

For each video session in the dataset, we run a DASH.js video player [12], which communicates with Pytheas using the API described in Figure 7.6a. To estimate the QoE of a video session, the video player does not stream video content from the selected CDN. Instead, it gets the QoE measurement from the dataset as described above.

We use CFA [126] as the prediction-based baseline. It is implemented based on [126]. CFA updates QoE prediction in the backend every minute, and trains critical features every hour. The frontend clusters run a simple decision-making algorithm – for 90% sessions, it picks the decision that has the best predicted QoE, and for the rest sessions, it randomly assigns them to a CDN.

We consider two widely used video QoE metrics [88, 126]: join time (the start-up delay of a video session), and buffering ratio (the fraction of session duration spent on rebuffering). We define improvement of Pytheas for a particular unit at t-th minute by $Improve_{Pytheas}(t) = \frac{Q_{CFA}(t) - Q_{Pytheas}(t)}{Q_{CFA}(t)}$, where $Q_{Pytheas}(t)$ and $Q_{CFA}(t)$ are the average QoE of Pytheas in t-th minute and that of the baseline, respectively. Since we prefer smaller values on both metrics, a positive value means Pytheas has better QoE.

Overall improvement: Figure 7.7 shows the distribution of improvement of Pytheas across all sessions. We can see that the mean improvement is 6% for join time and 31% for buffering ratio, and the 90th percentile improvement is 24% for join time and 78% for buffering ratio. To put these numbers in context, prior studies show a 1% decrease in buffering can lead to more than a 3-minute increase in expected viewing time [88]. Note that Pytheas is not better than the baseline on every single session, because the E2 process inherently uses a (dynamic) fraction of traffic to explore suboptimal decisions.



(a) Join time

(b) Buffering ratio

Figure 7.8: Improvement in presence of load effect.

Impact of load-induced QoE degradation: We consider the units where QoE of a CDN could significantly degrade when most sessions of the unit are assigned to use the same CDN. We assume that the QoE of a session when using a CDN under a given load (defined by the number of sessions assigned to the CDN in one minute) is the same to the QoE of a session randomly chosen in the dataset which uses the same CDN when the CDN is under a similar load. Figure 7.8 shows that the improvement of Pytheas when the number of sessions is large enough to overload a CDN is greater than the improvement when the number of sessions is not large enough to overload any CDN. This is because the prediction-based baseline could overload a CDN when too many sessions are assigned to the same CDN before CFA updates the QoE prediction, whereas Pytheas avoids overloading CDNs by updating its decisions in real time.

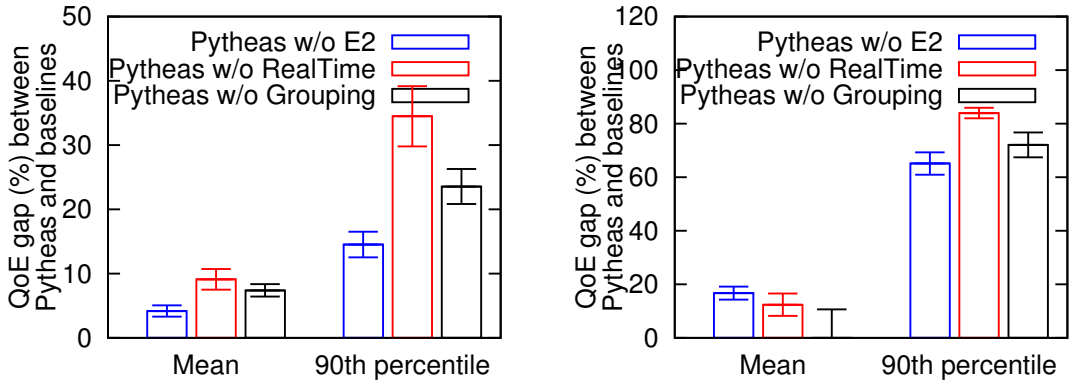
Contribution of Pytheas ideas: Having shown the overall benefit of Pytheas, we now evaluate the contribution of different components of Pytheas: (1) E2; (2) real-time update; and (3) grouping. We replace certain pieces of Pytheas by baseline solutions and compare their QoE with Pytheas's QoE. Specifically, for (1), we replace the per-group E2 logic by CFA's decision-making logic; for (2), we run Pytheas with data of one-minute staleness; and for (3), we run the same E2 process over all sessions, rather than on per-group basis. Figure 7.9 shows the improvement of Pytheas over each baseline, and it shows that each of these ideas contributes a nontrivial improvement to Pytheas; about 10-20% improvement on average QoE and 15-80% on the 90th percentiles.

7.7.2 Microbenchmarks

We create micro-benchmarks to evaluate the scalability and bandwidth consumption of Pytheas, as well as the benefits of various performance optimizations (Section 7.6).

Scalability

Frontend: Figure 7.10a shows the maximum number of sessions that can be served in one second, while keeping the update interval of per-group logic to be one second. Each session makes



(a) Join time

(b) Buffering ratio

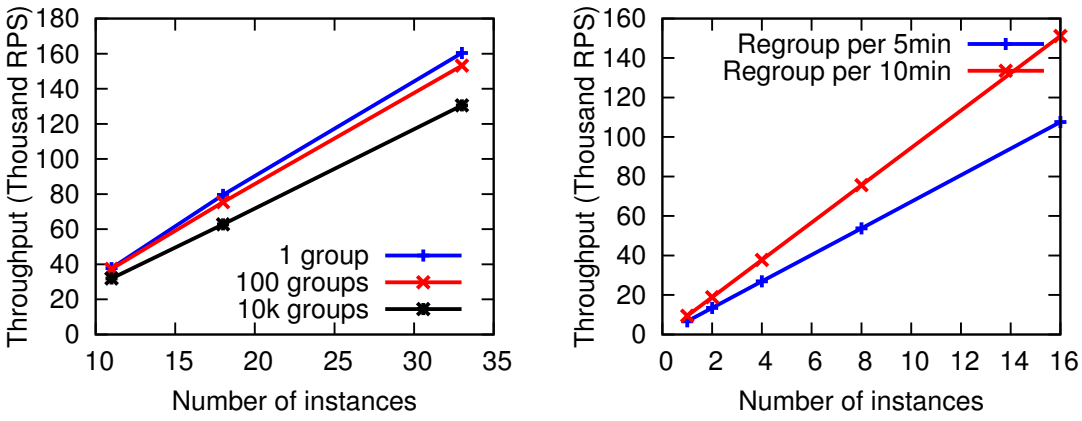
Figure 7.9: Factor analysis of Pytheas ideas

one control request and uploads QoE measurement once. Each group has the same amount of sessions. The size of control request message is 100B. We run Apache Benchmark [4] for 20 times and report the average throughput. We can see that the throughput is almost horizontally scalable to more frontend server instances. While the number of groups does impact the performance of frontend cluster, it is only to a limited extent; throughput of handling 10K groups is about 10% worse than that of handling one group.

Next, we evaluate the performance optimizations described in Section 7.6. We use a frontend cluster of 32 instances. Figure 7.11 shows that by separating E2 logic from client-facing servers, we can achieve 8x higher throughput, because each request reads cached decisions, which are still frequently updated. By replacing features by group identifiers, we can further increase throughput by 120%, because we can copy less data from client-facing servers to the servers running E2 logic. Note these results do not merely show the scalability of Spark Streaming or web servers; they show that our implementation of Pytheas introduces minimal additional cost, and can fully utilize existing data analytics platforms.

Backend: Figure 7.10b shows the maximum number of sessions that can be served in each second by a backend cluster, while keeping the completion time of session-grouping logic within 5 minutes or 10 minutes. We see that the throughput is also horizontally scalable with more instances in the backend cluster.

To put the scalability numbers of both frontend and backend in context, let us consider a content provider like YouTube which has 5 billion sessions per day [35] (i.e., 57K sessions per second). Pytheas can achieve this throughput using one frontend cluster of 18 instances and a backend cluster of 8 instances, which is a tiny portion compared to the sheer number of video servers (at least on the magnitude of hundreds of thousands [36]). This might make Spark Streaming and Kafka an overkill for Pytheas, but the scale of data rate can easily increase by one to two magnitudes in real world, e.g., tens of GB/s; for instance, each video session can request tens of mid-stream decisions during an hour-long video, instead of an initial request.



(a) Frontend (b) Backend

Figure 7.10: Pytheas throughput is horizontally scalable.

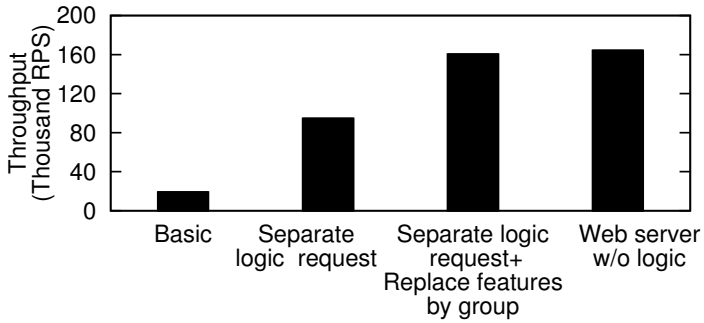


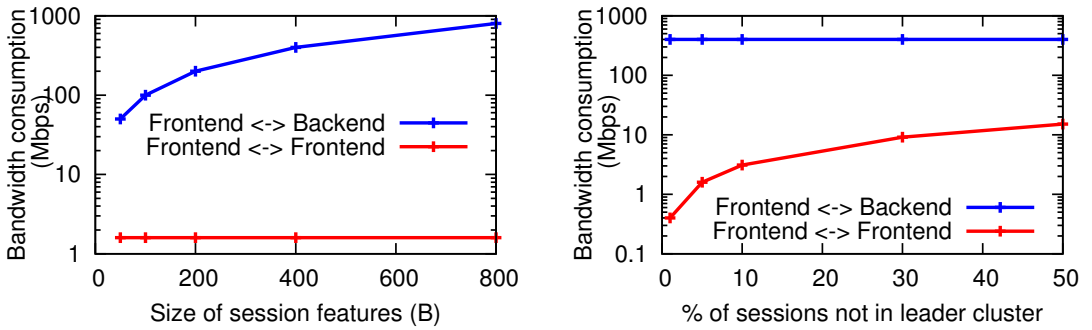
Figure 7.11: Optimizations of frontend throughput.

Bandwidth consumption

Since the inter-cluster bandwidth could be costly [170, 213], we now evaluate the inter-cluster bandwidth consumption of Pytheas. We consider one session group that has one proxy cluster and one leader cluster.

First, we evaluate the impact of message size. We set the fraction of sessions received by the proxy cluster to be 5% of the group, and increase the request message size by adding more features. Figure 7.12a shows that the bandwidth consumption between the frontend clusters does not grow with larger message size, because the session features are replaced by group identifiers by the client-facing servers. Only the bandwidth consumption between frontend and backend grows proportionally with the message size but such overhead is inherent in existing data collection systems and is not caused by Pytheas.

Next, we evaluate the impact of fraction of sessions received by the proxy cluster. We set the message size to be 400B, and change the fraction of sessions received by each proxy cluster. Figure 7.12a shows that the bandwidth consumption between frontend clusters raises as more measurement data need to be forwarded from proxy to the leader cluster, but it is much smaller than the bandwidth consumption between frontend and backend.



(a) Impact of message size

(b) Impact of % sessions not in the leader cluster

Figure 7.12: Bandwidth consumption between clusters.

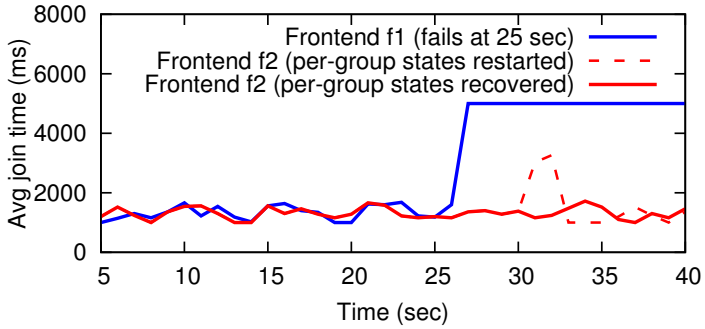


Figure 7.13: Pytheas can tolerate loss of a frontend cluster by falling back to player native logic gracefully, and recovering the logic states in a new cluster.

7.7.3 Fault Tolerance

Finally, we stress test the prototype under the condition that a leader frontend cluster fails. We set up 10 video players, each of which can stream content from two CDNs. CDN1 has 5000ms join time and CDN2 has 1000ms join time. By default, the player's native logic chooses CDN1. There are two frontend clusters, f1 and f2. The experiment begins with f1 being the leader cluster, and it loses connection at $t = 25$.

Figure 7.13 shows the time-series of QoE of sessions that are mapped to each frontend clusters. First, we see that the sessions mapped to f1 can fall back to the CDN chosen by the player's native logic, rather than crashing. Second, right after f1 fails, f2 should still be able to give cached decision (made by f1 before it fails) to its sessions. At $t = 30$, the backend selects f2 as the new leader for the group. At the point, a naive way to restart per-group logic in the new leader is to start it from scratch, but this will lead to suboptimal QoE at the beginning (the dotted line between $t = 30$ and $t = 35$). Pytheas avoids this cold-start problem by keeping a copy of the per-group states in the proxy cluster. This allows the proxy cluster to recover the per-group control states without QoE degradation.

7.8 Related Work

Data-driven QoE optimization: There is a large literature on using data-driven techniques to optimize QoE for a variety of applications, such as video streaming (e.g., [101, 126]), web service (e.g., [146, 193]), Internet telephony [109, 125], cloud services (e.g., [137]), and resource allocation (e.g., [56]). Some recent work also shows the possibility of using measurement traces to extrapolate the outcome of new system configurations [40]. Unlike these prediction-based approaches, we formulate QoE optimization as an real-time E2 process, and show that by avoiding measurement bias and enabling real-time updates, this new formulation achieves better QoE than prediction-based approaches.

Related machine learning techniques: E2 is closely related to reinforcement learning [215], where most techniques, including the per-group E2 logic used in Pytheas, are variants of the UCB1 algorithm [48], though other approaches (e.g., [39]) have been studied as well. Besides E2, Pytheas also shares the similar idea of clustering with linear mixed models [151], where a separate model is trained for each cluster of data points. While we borrow techniques from this rich literature [102, 174], our contribution is to shed light on the link between QoE optimization and the techniques of E2 and clustering, to highlight the practical challenges of adopting E2 in network applications, and to show group-based E2 as a practical way to solve these challenges. Though there have been prior attempts to cast data-driven optimization as multi-armed bandit processes in specific applications (e.g., [125]), they fall short of a practical system design.

Geo-distributed data analytics: Like Pytheas, recent work [170, 172, 213] also observes that for cost and legal considerations, many geo-distributed applications store client-generated data in globally distributed data centers. However, they focus on geo-distributed data analytics platforms that can handle general-purpose queries received by the centralized backend cluster. In contrast, Pytheas targets a different workload: data-driven QoE optimization uses a specific type of logic (i.e., E2), but has to handle requests from millions of geo-distributed sessions in real time.

7.9 Summary

While previous two chapters have shown impressive QoE improvement by formulating DDN as a prediction process, they have key limitations that curtail the potential of data-driven optimization. Drawing on a parallel from machine learning, we argue that real-time exploration and exploitation is a better abstraction for this domain. In designing Pytheas, we have addressed key practical challenges in applying real-time E2 to network applications. Our key idea is a *group-based E2* mechanism, inspired by the insight of persistent critical structures in QoE-determining factors. We observe that application sessions sharing the same features can be grouped so that we can run E2 at a coarser per-group granularity. Using an end-to-end implementation and proof-of-concept deployment of Pytheas in CloudLab, we showed that Pytheas improves video quality over state-of-the-art prediction-based system by 6-31% on mean, and 24-78% on tail QoE.

Chapter 8

Tackling Large Decision Spaces

In previous chapters, we have focused on addressing system and algorithmic challenges resulting from a complex relationship between session-level features and QoE. In this chapter, we examine another challenge caused by *large decision spaces*. As we will see, this challenge is especially salient in Internet telephony.

To alleviate call quality problems shown in Section 4.2, we present in this chapter an architecture called VIA, which revisits the use of traditional overlay techniques to relay calls using the emerging architecture of *managed overlay networks*, which use the well-connected cloud servers and private backbone network as VoIP relay points (supernodes) and network paths. Our trace-driven analysis shows that the number of calls whose quality is impacted by poor network performance can be reduce by 53% by optimally selecting the relay path in Microsoft's managed overlay network.

While it is tempting to realize this potential improvement by using a DDN-based solution similar to Pytheas, it is challenging to discover best relay paths due to the sheer number of possible relay paths (in hundreds) and their dynamic performance (which could change on timescales of minutes). To address the challenge of large decision spaces, we develop a practical relay selection system for VIA that intelligently combines prediction-based filtering with an online exploration-exploitation strategy. The key insight of VIA relay selection is that the decision space can be reduced by leveraging the persistent critical structures: for each pair of caller AS and callee AS, there is a small and stable subset of promising relays that contains the best one. Trace-driven analysis and a small-scale deployment shows that VIA cuts the incidence of poor network conditions for calls by 45% (and for some countries and ASes by over 80%) while staying within a budget for relaying traffic through the managed network.

This chapter is organized as follows. We begin by describing the architecture of relay selection in Internet telephony in Section 8.1, and quantify the potential benefits of a managed overlay network for improving audio call quality in Section 8.2. Then in Section 8.3, we then highlight the challenges in achieving these benefits and present a practical relay selection algorithm that delivers close-to-optimal performance. Section 8.4 uses simulation driven by real-traffic measurement and shows that VIA can significantly improve Skype performance on network metrics. Finally, we discuss related work in Section 8.6, and summarize the section in Section 8.7.

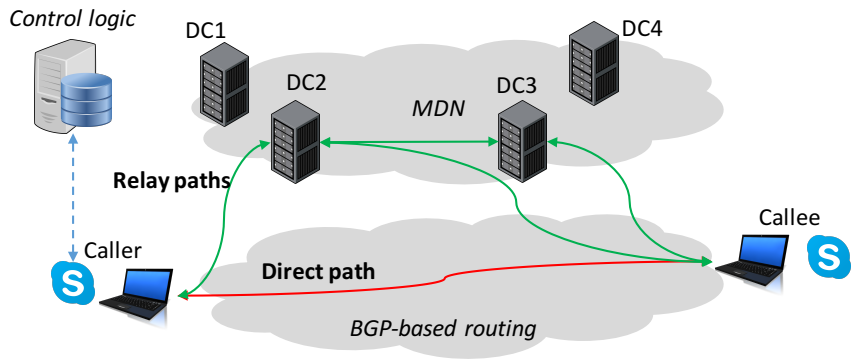


Figure 8.1: VIA architecture with relay nodes at globally distributed data centers. A call can either take "default path" (red) or a "relay path" (green).

8.1 VIA Architecture

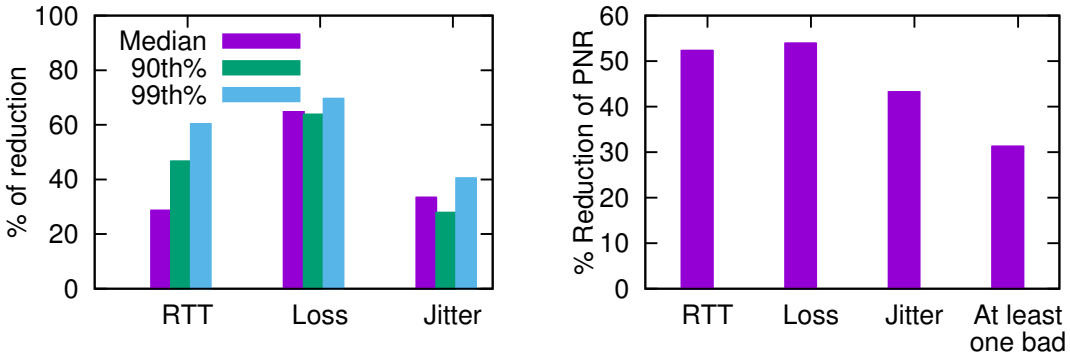
Figure 8.1 presents the VIA architecture that consists of relay nodes placed at globally distributed datacenters, such as those run by Amazon, Google, and Microsoft. Indeed, VIA's architecture bears similarities to those used by Google Hangouts and Skype [221], but with a key difference — today, the relays are typically used to provide connectivity between any two clients, while VIA is engineered to *explicitly* optimize network performance and call quality.

Each call can take either the "*default* path" (red arrow) or a "*relayed* path" (green arrows) that routes the traffic through one or more relay nodes in the DCs. Relayed paths could include a single relay to "*bounce* off" traffic or a pair of relays to enable traffic to "*transit* through" the private backbone of the managed overlay network.

In our study, we use all the relay nodes operated by Skype. They are all located in a single AS (so all inter-relay paths are within a private WAN) but spread across many tens of datacenters and edge clusters worldwide. We assume the caller (or callee) can reach these relays by explicitly addressing the particular relay(s). The network path between a relay and a client is determined by BGP.

When establishing a call, after the caller signals its callee, both the caller and callee contact a *controller* (Figure 8.1) to determine whether they should use the direct path or a relayed path, and, in case of the latter, which relay(s) they should use. The controller makes this decision based on the performance measurements from historical calls and policy constraints (such as those based on relay budget or current load), to be described in Section 8.3. To aid in this process, Skype clients periodically push the network metrics derived from their calls, to the controller.

The controller does not need to directly monitor the relay nodes because their performance (including degradation and failure) would be reflected in the end-to-end measurements made by clients who use the relays. To avoid overloading the controller, each client could cache the relaying decisions and refresh periodically though we do not consider this here. (We discuss implementation issues in Section 8.5).



(a) Performance distribution

(b) Poor Network Rate

Figure 8.2: Potential improvement of VIA.

8.2 Potential Relaying Improvement

Next, we quantify the potential gains of VIA, using an "oracle" control logic, which enjoys the benefit of foresight. For each call between a source-destination pair, it has knowledge of the average performance of each *relaying option* on a given day. As shown in Figure 8.1, a relaying option could be either the default (direct) path, a bouncing relay path, or a transit relay path. For each source-destination pair, the oracle picks the relaying option that has the best average performance (i.e., lowest RTT, loss rate, or jitter) for this source-destination pair on this day—either a relay path or the direct path.[1] We also have information from Skype on the RTT, loss and jitter between their relay nodes, which we use in estimating the performance of a transit relay path.

The oracle makes two simplifying assumptions: (1) there are no load restrictions on the relays or the network backbone, and (2) the performance measurements of each relaying option are indicative samples of its actual performance. In Section 8.3.6, we will relax the first assumption by introducing a budget constraint on the fraction of calls being relayed.

Gains from oracle approach: Figure 8.2 shows the improvement (i.e., reduction) in the values of RTT, loss and jitter individually as well as the PNR (defined in Section 4.2.1). Specifically, if a statistic goes from b to a, we define the *relative* improvement as $100 \times (\frac{b-a}{b})$, which lies between 0 and 100.

The oracle can help reduce RTT, loss and jitter by 30%-60% at median (Figure 8.2a). Reduction at the tail, which is of particular significance in interactive services, is nearly 40%-65% with the oracle's choice of relaying. All this translates to a healthy reduction in the PNR on each of RTT, loss, and jitter (Figure 8.2b, left three bars) of up to 53%. Source-destination pairs with fewer calls between them have a lower impact on the PNR and its improvement.

We also analyze the reduction in PNR when the three metrics are considered *together*, i.e., improving from a situation where at least one of the metrics is poor to a situation where *none*

[1]Picking a day's granularity gives us sufficient samples for most of the relaying options. Nevertheless, for the small fraction of source-destination pairs for which we had sufficient samples on a timescale of minutes, we found that the oracle still had a significant benefit.

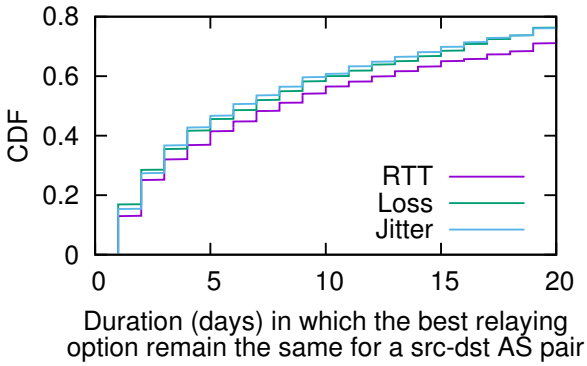


Figure 8.3: Distribution of how long the best relaying option (picked by oracle) lasts. The optimal relaying options for 30% of AS pairs last for less than 2 days.

of the three is poor (i.e., RTT $\leq$ 320ms, loss $\leq$ 1.2%, *and* jitter $\leq$ 12ms), while still optimizing for RTT, loss and jitter individually. Even while optimizing for each of the three metrics, we can obtain a PNR for "at least one bad" metric; we conservatively pick the *worst* among the three for our analysis. Despite this strict stipulation, we can achieve reduction of over 30% in PNR (Figure 8.2b, right-most bar).

Need for dynamic relay selection: Whether the controller should select relay dynamically depends on how often the relaying decisions need to be updated. Figure 8.3 shows the distribution of the median duration during which the oracle picks the same relaying option for a source-destination AS pair. The optimal relaying option for 30% of AS pairs lasts for less than 2 days, and only 20% of AS pairs have the same optimal relay option for more than 20 days. This, together with the observation on the relatively low persistence of poor performance (Figure 4.8), suggests that the relay selection should be done *dynamically*, rather than statically.

8.3 VIA Relay Selection

Having shown that relaying through VIA could provide significant gains, we now devise a *practical* algorithm for relay selection. We begin by formulating the problem of relay selection. We describe two classes of strawman approaches — purely *predictive* and *exploration-based* — and highlight limitations of both classes. We then present the core intuition behind our relay selection algorithm, called *prediction-guided exploration* and then describe the solution.

8.3.1 Problem Formulation

Our goal is to assign each call to a particular relaying option as discussed in Section 8.1. Recall that a relaying option can use the default path, use a specific one-hop relay node (i.e., bouncing relaying), or use a specific pair of relay nodes (i.e., transit relaying). Let C denote the set of calls we want to optimize and let R denote the set of available relaying options. We use $c \in C$ and $r \in R$ to denote a specific call and relaying option, respectively. Let $Q(c,r)$ denote the expected value of a network metric for c when using r (a smaller value is better). We assume that the

relaying decisions for calls are independent; i.e., the performance of a call is not impacted by the relaying decisions made for other calls.

The goal of VIA is to *assign* optimal relaying options for each $c \in C$. Let $\mathsf{Assign} : C \to R$ denote the assignment function output by some algorithm and let $\mathsf{Assign}(c)$ be the relaying option assigned for call $c \in C$. Formally, our objective is to find the optimal assignment

$$\arg \min_{\mathsf{Assign} \in R^C} \sum_{c \in C} Q(c, \mathsf{Assign}(c))$$

This is a minimization problem because a lower value is better for each of our network quality metrics Q.

8.3.2 Strawman Approaches

We can consider two classes of approaches for the optimal assignment of relaying options to calls:

1. *Exploration-based:* One approach is to set aside a fraction of the calls for measurement-based exploration of the performance of each possible relaying option for every source-destination pair. For instance, for every AS-pair and every possible relaying option r, we will explicitly use some of the calls to explore the option and measure the performance, $Q(c,r)$.
2. *Prediction-based:* An alternative to the exploration-based approach is to use the recent history of observed call performance. Suppose, VIA has available as input call records with measured performance H. Then, we can use suitable prediction algorithms to predict the performance $Q(c,r)$ for every combination, and select the option that has the best predicted performance.

Unfortunately, we observe in practice that both classes of approaches individually have very poor accuracy in predicting $Q(c,r)$. This ultimately results in a poor assignment strategy and poor call quality. There are two key reasons.

First, there is a fundamental problem because of *skew in data density.* Specifically, there is a substantial difference in the number of call samples available across different source-destination pairs, both for the direct path and for the various relayed paths. This variability arises because of the large space of choices: N end-points and M relay strategies lead to $O(N^2M)$ choices. Furthermore, certain end-points make/receive fewer calls, yielding fewer samples. Second, there is *inherent variability* in the observed performance. Consequently, to estimate $Q(c,r)$, we need a significant number of samples before the empirically observed values can converge to the true values.

The skew and the variability make prediction inaccurate and exploration ineffective and/or expensive (in terms of the effort to be expended).

8.3.3 Overview of VIA

The key intuition behind our solution is the empirical observation that even though a prediction-based approach may not predict the optimal choice, the optimal is likely in the top few of its

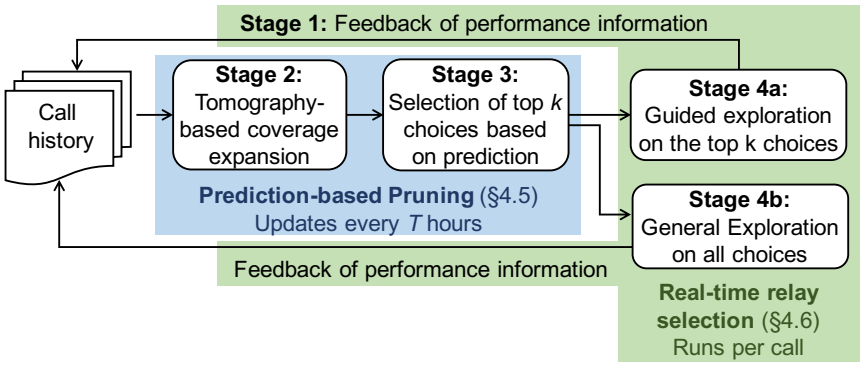


Figure 8.4: Overview of VIA relay selection based on prediction-guided exploration.

predictions. In other words, if we look at the top-k choices (those who have the best predicted performances), the optimal choice will likely be a member of that set.

We can exploit this observation to *prune* the search space for our exploration step. That is, the exploration approach does not need to blindly explore the set R of all possible strategies, but instead can focus on a much smaller set of top-k predictions. We refer to this as a *prediction-guided exploration* approach. The top-k pruning is not to be confused with a similar machine learning problem which seeks to find k best options (e.g., [68]). In contrast, we care more about the best relaying option – our top-k candidates may have bad options, but the best relaying option is very likely to be among them, and can be found by exploration techniques.

Figure 8.4 depicts the main stages in VIA, and Algorithm 4 shows the pseudocode. In a nutshell, the logical stages are:

1. gathering performance information from call history,
2. using network tomography to expand the coverage of the information from call history,
3. using the (expanded) history information to predict performance and prune all but the most promising top-k relaying options, and
4. perform exploration-exploitation on the top-k relaying options as well as all relaying options using multi-armed bandit (MAB) techniques.

Finally, the observed performance of each call will be stored in call history, i.e., fed back to stage 1. Stages 2 and 3 (shown in light blue) are performed at a periodicity of T hours (by default 24 hours), i.e., the pruned list of candidate relaying options are refreshed every T hours. Stages 1 and 4 (shown in light green) are performed on a per-call basis. We discuss these stages in the sub-sections that follow.

8.3.4 Prediction-Based Pruning

Using call history data, VIA proceeds to predict, with confidence intervals, the performance between a source-destination pair over each relaying option: direct paths, and each transit and bouncing relay.

Expanding coverage by network tomography: Call history tells us the performance of paths that were actually used. As there is skew in call distribution, there might be "holes", i.e., no call

Input: Set of calls C to be assigned to relaying options R, and set of historical calls H
Output: A relay assignment, Assign, where each call $c \in C$ is assigned a relay option $\mathsf{Assign}(c) \in R$

```
   /* Stage 2: Tomography-based performance predictor trained from H                    */
1  Pred ← BuildPredictor(H)
   /* Stage 3: Pick Top-k candidates based on history-based prediction.                 */
2  Assign ← ∅
3  for (s,d) do
4      TopK← GetTopK(s,d,R,Pred)
       /* See Algorithm 5                                                               */
5      for c ∈ C do
6          if RandomFloat(0,1) < ε then
               /* Stage 4a: Explore the Top-k candidates                                */
7              r ← Explore(c,s,d,TopK,Assign,Pred)
8          else
               /* Stage 4b: Randomly explore all relaying options                       */
9              Assign(c) ← Random(R)
10         Assign(c) ← r
11 return Assign
```

Algorithm 4: Relay selection algorithm of VIA

history for the network path between a source-destination pair through a specific relaying option. Can we learn about the performance of these network paths?

If we knew the performance of the individual network segments (e.g., client to relay) that comprise an end-to-end path, we could compose these to estimate the performance of the path. In principle, measurements of the individual network segments could be made by the relays themselves. However, the relays in Skype were only designed to forward traffic and we were not in a position to add new functionality to these relay nodes (and potentially impose additional overheads).

Network tomography provides an alternative. By combining end-to-end measurements across several, partially-overlapping paths, network tomography can help estimate the performance of each network segment. Then, by stitching together the estimates for the individual segments, we can estimate the performance of a path not seen before.

Figure 8.5 shows a simple example of how network tomography expands coverage. We use linear tomography, and apply it to individual

Given a relay path that uses relaying option r and between source AS s and destination AS d, our tomography algorithm models it as a path consisting of two segments: a segment between s and r and a segment between d and r. Modeling network end-points on AS level is a pragmatic trade-off: it gives us sufficient data on many source-destination pairs, and still produce significant improvement (see Section 8.4.4 for comparison between different granularities). The prediction algorithm can work at a finer granularity (e.g., /24 IP prefix) when more data are available.

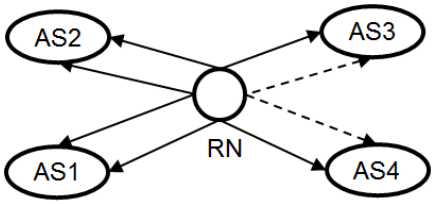


Figure 8.5: Path stitching in VIA to estimate performance through relay RN. Solid lines represent historical call samples that we use to predict performance between AS3 and AS4 (dotted line). $\text{RTT}_{\text{AS3}\leftrightarrow\text{AS4}} = \text{RTT}_{\text{AS1}\leftrightarrow\text{AS4}} + \text{RTT}_{\text{AS2}\leftrightarrow\text{AS3}} - \text{RTT}_{\text{AS1}\leftrightarrow\text{AS2}}$.

The Pred module (Algorithm 4, line 2) predicts for a source-destination pair (s and d) both the mean performance $\text{Pred}_{mean}(s,d,r)$ for a specific relaying option r, and its standard error of mean (SEM) $\text{Pred}_{sem}(s,d,r)$. Based on these, Pred estimates both the lower and higher 95% confidence bounds: $\text{Pred}_{lower}(s,d,r) = \text{Pred}_{mean}(s,d,r) - 1.96\text{Pred}_{sem}(s,d,r)$ and $\text{Pred}_{upper}(s,d,r) = \text{Pred}_{mean}(s,d,r) + 1.96\text{Pred}_{sem}(s,d,r)$.

Pruning to get top-k choices: Pruning does not necessarily narrow down to the single best relaying option. However, we see that the best relaying option is often among the top-k predicted options for a small value of k. For instance, the probability of the option with the minimum RTT being included even in top three or four ($k = 3$ or 4) is $60\% - 80\%$ as against just 29% if we were to pick only the option with the predicted minimal RTT ($k = 1$). Therefore, we adopt the approach of using our predictor to pick the top-k relaying options and use that for *guided exploration*.

Instead of using a fixed value of k, VIA *dynamically decides* k based on the lower and higher confidence bounds for each relay r on the particular source-destination pair s and d. Algorithm 5 shows the pseudocode. Specifically, we define top-k to be the minimal set of relaying options such that the lower 95% confidence bound ($\text{Pred}_{lower}(s,d,r)$) of any relay option not in the top-k is higher than the upper 95% confidence bound ($\text{Pred}_{upper}(s,d,r)$) of any relay option in the top-k. (Recall that the lower the value of a network metric, the better it is.) In other words, we are very sure that any relay option that is not included in the top-k is worse than any that is. For instance, the probability of the option with minimal RTT being included in such top-k is over 90%.

8.3.5 Exploration-Exploitation Step

Exploring the top-k choices for each source-destination pair (Explore of line 10 in Algorithm 4) can be formulated as an instance of the classic multi-armed bandit problem, where each of the relaying options is an "arm" of the bandit and the network performance obtained is the "reward". While bandit selection is a much studied problem, doing so under high-variance and dynamically changing performance distributions (i.e., rewards) of the bandits, and also limited budget for each bandit, requires interesting adaptations, as outlined below.

Relay options selected by the basic *exploration-exploitation* process assigns a fraction of calls to explore different relay options (ε-greedy) and the rest to exploit the best decision.[2] As

[2]Exploration-exploitation could also be invoked on per-packet basis within the call. However, this would require

Input: Source AS s, destination AS d, relaying options R, and predictor Pred (from Algorithm 4)
Output: Top k relaying options TopK for (s, d) calls

Function GetTopK(s, d, R, Pred)
    /* Initializing variables */
    TopK$\leftarrow \emptyset$; Remained $\leftarrow R$; $h \leftarrow \infty$
    while *true* **do**
        $r \leftarrow \text{argmin}_{r \in \text{Remained}}\, \text{Pred}_{\text{upper}}(s,d,r)$
        if $\text{Pred}_{lower}(s,d,r) > h$ **then**
            break
        else
            $h \leftarrow \text{Pred}_{\text{high}}(s,d,r)$ Remained $\leftarrow$ Remained $\setminus \{r\}$ TopK $\leftarrow$ TopK $\cup \{r\}$
    return TopK

Algorithm 5: Predicting the top-k choices.

briefly mentioned earlier, standard exploratory approaches are slow to converge (Section 8.3.2) and often fail to select the best decision (Section 8.4.3). This is because exploring in presence of high variability requires a lot of samples, which is infeasible due to data sparseness and skew.

Algorithm 6 shows the pseudocode of our approach. Here, we choose the UCB1 algorithm [48] as our basic starting point. UCB1 is well-suited for our purpose because it does not require explicitly specifying the fraction of samples for exploration. Instead, it transparently combines both exploration as well as its exploitation decisions. We make two modifications to the basic algorithm in order to make it work well in our context.

1. UCB1 normalizes rewards (i.e., performance) from each bandit (i.e., relay option) to be between 0 and 1. In our situation, however, normalizing based on the full range of values of each performance metric is problematic due to the large variance in distribution (e.g., unusually large RTT). Normalizing all values based on such a wide range leads to poor decisions because the difference between values in the common case become hard to discern. Instead, we normalize the rewards by dividing them by the average of upper 95% confidence bounds ($\text{Pred}_{upper}(s,d,r)$) of the top-$k$ candidates.
2. The top-k pruning in Section 8.3.4 is a function of only the samples explored. Therefore, to avoid being blindsided by dynamically changing performance distributions, VIA also sets aside ε fraction of calls to random relays (outside of the top-k) for *general* exploration. This step is not required in traditional exploration-exploitation techniques as they assume the reward (performance) distribution of each bandit (relay option) is static, which may not hold in our context.

8.3.6 Budgeted Relaying

We extend VIA's relaying decision to consider budget constraints: so the fraction of calls being relayed must be less than a certain limit, B (e.g., 30%). While such an overall budget on relayed

packet-level control, which is out of the scope of this paper.

Input: Call c, source AS s, destination AS d, top k relaying options TopK, relay assignment Assign, predictor Pred
Output: Relay selection of c

Function Explore$(c, s, d, TopK, \text{Assign}, \text{Pred})$
  /* Initializing variables */
  $\text{ucb}_{\min} \leftarrow \infty$; $r_{\text{top}} \leftarrow$ null
  /* To avoid outliers, we do not use maximum performance as normalizer w. */
  $w \leftarrow \frac{1}{|\text{TopK}|} \sum_{r \in \text{TopK}} (\text{Pred}_{\text{upper}}(s, d, r))$
  /* Following is the standard UCB1, except for the normalization scheme. */
  $T \leftarrow |\text{Assign}| + 1$
  for $r \in TopK$ **do**
    $C_r \leftarrow \{c' | \text{Assign}(c') = r\}$
    /* Q is the quality function. */
    $\text{ucb} \leftarrow \frac{1}{w|C_r|} \sum_{c' \in C_r} Q(c', r) - \sqrt{\frac{0.1 \log T}{|C_r|}}$
    if $ucb < ucb_{min}$ **then**
      $r_{\text{top}} \leftarrow r$
      $\text{ucb}_{\min} \leftarrow \text{ucb}$
  return r_{top}

Algorithm 6: Exploring the top-k candidates in real time using modified UCB1.

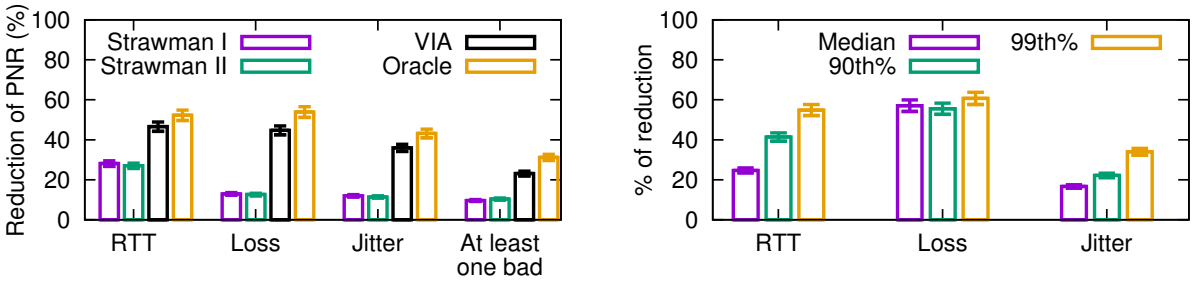
calls is simple, in general it may also be of interest to consider other budget models, such as per-relay limits or bandwidth cap on call-related traffic.

VIA utilizes the budget using a simple extension to the heuristic in Section 8.3.5. It decides to relay a call only if the benefit of relaying is *sufficiently* high. If the overall budget for relaying calls is B percent, a call should be relayed only if the benefit of relaying it is within the top B percentile of calls. VIA uses historical call information (of relaying benefits) to keep track of the percentiles. It decides to relay a call only if the expected benefit is above the B^{th} percentile benefit.

8.4 Evaluation

In this section, we show that VIA can significantly improve performance on network metrics. Specifically, we show that:

- VIA achieves substantial improvement on all network metrics — $20\% - 58\%$ reduction on median (compared to the oracle's 30%-60%; Section 8.2) and $35\% - 60\%$ on 99^{th} percentile. VIA reduces PNR by $39\% - 45\%$ for the individual metrics (compared to the oracle's 53% in Figure 8.2b), and by 23% when PNR is computed on an "at least one bad" metric (compared to the oracle's 30%).
- VIA achieves close-to-optimal performance under budget constraints by selectively relaying calls that have higher potential benefit (Section 8.3.6).



(a) VIA, strawmen, oracle vs. default

(b) VIA improvement on percentiles

Figure 8.6: Improvement of VIA. PNR on individual metrics improve by 39% – 45% and on the "at least one bad" metric by 23%.

- VIA's improvement increases as relay decisions are made at finer spatial granularities and more dynamically. However, we start to see diminishing gains at granularities finer than AS-pair and daily.

8.4.1 Methodology

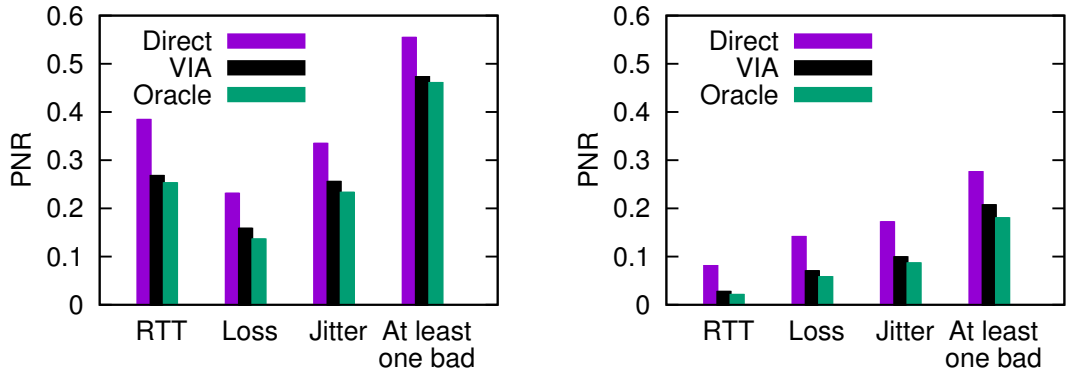
We perform data-driven simulations based on 430 million Skype calls (Section 6.1.2). The calls are replayed in the same chronological order as in the trace thereby allowing VIA to gain knowledge as it goes along (using newer call measurements). We assume that when a call is assigned to certain relay option, its performance would be the same as that of a call which is randomly sampled from the set of calls between the same AS pair through the same relay option in the same 24-hour window. Tomography-based performance prediction is made based on call performance in the last 24-hour window. For statistical confidence, in each 24-hour window, we focus on AS pairs where there are at least 10 calls on at least 5 relay options [3]. Also, the relaying options considered for a call are only those with at least 10 call samples. To quantify the confidence in the results, we also add error bars (of standard error of mean) to the graphs. Note that even with the aggregation, we used *distribution* (e.g., mean, percentiles) of the metrics and not per-call values for evaluation.

This section shows how much VIA can reduce PNRs (fraction of calls having poor performance on the individual network metrics or on the "at least one bad" metric), compared with the oracle approach and a strawman, such as using the default paths for all calls ("default strategy").

8.4.2 Improvement of VIA

PNR reduction: Figure 8.6a shows the PNR reduction of VIA over default strategy (always using default paths), and compares it with the PNR reduction of pure prediction-based selection, based just on history (Strawman I), pure exploration-based selection without any pruning of the options up front (Strawman II), and oracle. Across all three performance metrics, we see that VIA

[3]Otherwise, selecting relays from a handful of candidates would be trivial. 32 million calls remain after these filters.



(a) International (b) Domestic

Figure 8.7: VIA improvement on international and domestic calls. We also have similar observation regarding inter-domain and intra-domain calls.

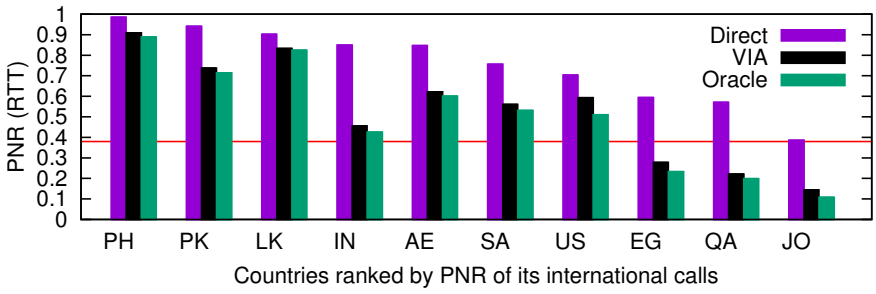
achieves close-to-oracle performance and significantly outperforms both the default strategy and the two strawman approaches. The strawman approaches yield much less improvement, which confirms the inefficiency of the pure predictive and pure exploratory strategies (Section 8.3.2).

Improvement on percentiles: Figure 8.6b shows the improvement over default strategy on different percentiles. We first calculate the percentiles of performance of each strategy and calculate the improvement between these percentiles (which avoids the bias of calculating improvement on each call). We see that VIA has improved performance on both median (by $20\% - 58\%$) and the extreme tail (by $20\% - 57\%$ on 90^{th} percentile), which shows VIA is able to improve the performance of a wide spectrum of calls.

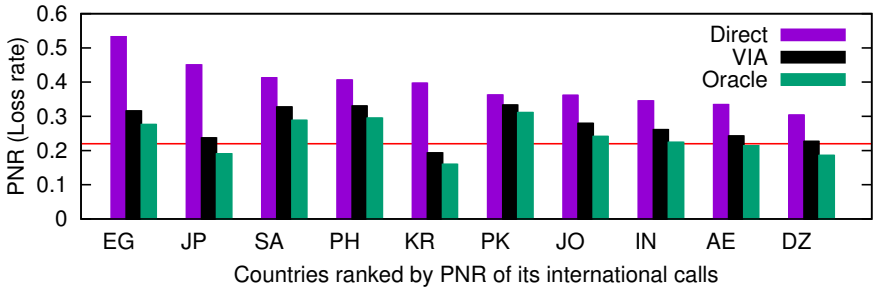
Transit vs. bouncing relay: Finally, we find that also using transit relaying (i.e., using inter-DC connection between the ingress and egress relays as part of the path) usually results in higher improvement on PNR than only using bouncing relays (i.e., using one relay node to bounce off traffic). On AS pairs which have used both bouncing and transit relays, we see 50% lower PNR when both transit and bouncing relays are available than when transit relays are excluded. We also find that VIA sends about 54% calls to bouncing relays, 38% to transit relays, 8% to default paths, with a marginal difference in the distribution across network metrics.

International vs. domestic: Figure 8.7 compares PNR of international and domestic calls under strategies of default, VIA and oracle.We see significant improvement of VIA on both international and domestic calls, while international calls have a slightly higher magnitude of improvement than domestic calls. This can be explained by the fact that relaying has limited benefits when the bottleneck is the last-mile ISP or the last-hop connection.

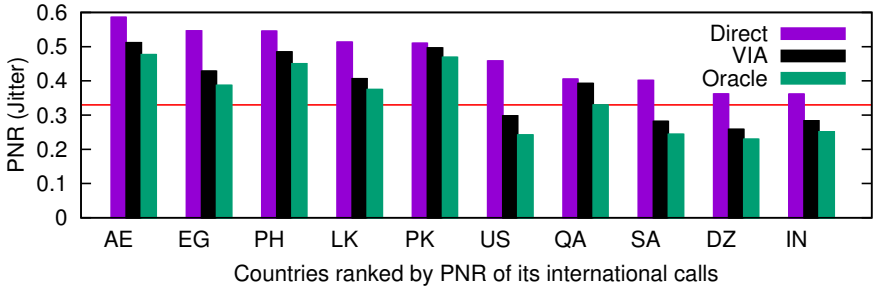
Benefits by countries: Figure 8.8 further dissects the improvement of VIA by countries (with one side of the international call in that country) with worst (direct) PNR. It shows that the worst countries have a much higher (direct) PNR than the global PNR, shown by the horizontal red line, and that the performance of VIA is closer to the oracle than to the default for most of these countries.



(a) PNR of RTT



(b) PNR of loss rate



(c) PNR of jitter

Figure 8.8: Dissecting VIA improvement on PNR by country of one side. There is a substantial diversity on VIA improvement across different countries.

8.4.3 VIA's Design Choices

Prediction accuracy of relay-based tomography: As a first step, VIA uses relay-based tomography (Section 8.3.4) to predict the performance each relaying option. We evaluated the accuracy of tomography-based predictions on the different metrics and found that on 71% of calls, the predicted performance is within 20% from the actual performance. However, for 14% of the calls, the error can be $\geq 50\%$. This non-negligible prediction error explains the poor performance of Strawman I (pure prediction-based) that we have seen in Figure 8.6a, and also motivates real-time exploration.

Benefits of prediction-guided exploration: As discussed in Section 8.3, VIA is not a simple

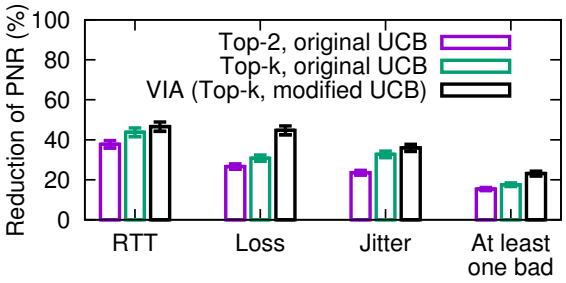


Figure 8.9: Comparing guided-exploration strategies.

combination of prediction and exploration approach. First, instead of picking a fixed number top candidates, VIA pick top candidates by taking variance of prediction into account. Second, instead of using the original UCB1 algorithm, which assumes a normal distribution of rewards, we adopt a different way to normalize values to cope with performance outliers. Figure 8.9 quantifies the incremental contribution of both modifications on PNR of the three metrics. It shows that each modification makes a significant contribution to VIA's improvement. With the "at least one bad" metric, picking top k and using the normalized reward reduces PNR by 24% compared to 15% with just the top 2 (loss rate PNR by 44% compared to 26%).

8.4.4 Practical Relaying Factors

Relaying budget: Being able to use relays judiciously within a budget for relayed calls is an inherent requirement in the context of managed overlay networks such as VIA. Here, we define budget as the maximum fraction of calls being relayed. We only impose an overall budget, not a per-relay one. Figure 8.10 shows the impact of budget on PNR (of at least one bad metric) of three strategies: oracle, budget-unaware VIA and budget-aware VIA. The budget-unaware VIA, which selects relays based on Algorithm 4, will relay calls whenever there is potential benefit of doing so, without taking into consideration the overall budget of relaying. Therefore, there is a risk of the budget getting used up by calls with only small benefit. In contrast, budget-aware VIA (Section 8.3.6) relays a call only when the benefit is larger than a threshold, which depends on the actual budget. That means calls with minimal benefit will not be relayed, saving resources for the calls that would benefit the most by relaying. From Figure 8.10, we see that the budget-aware VIA (Section 8.3.6) can use budget much more efficiently than the budget-unaware VIA. Also, budget-aware VIA can achieve about half of the maximum benefit (i.e., when budget is 100% of calls) with a budget of 0.3 (i.e., only relying 30% of calls).

Relaying decision granularities: We show performance improvement as a function of the spatial and temporal granularity at which VIA operates. First, to show the impact of spatial granularity, Figure 8.11a fixes the temporal granularity to running stage (2) and (3) of VIA every 24 hours, i.e., $T = 24$ hours (Figure 8.4) and compares the PNR if different relay options could be selected for calls in different spatial granularities. For fair comparison, the PNR are calculated based on the same set of calls.

We see two consistent trends. First, making decision at granularities coarser than a per AS

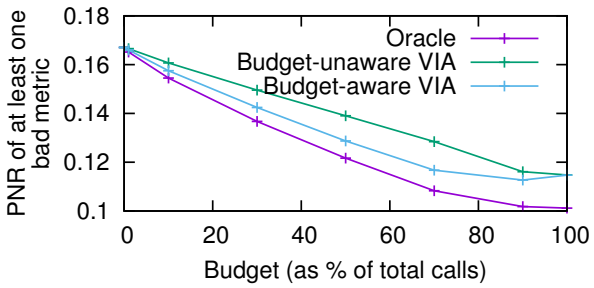


Figure 8.10: Impact of budget constraint on VIA.

pair results in a smaller reduction in PNR. For instance, different ISPs within a country have different peering relationships, and thus may have different optimal relay options, but such opportunities will not be exploited when making decision per country. Second, making decisions on finer granularities does not help much, though for a different reason. At finer granularities, the coverage becomes much smaller, which make VIA unable to predict many potential relay options. In future work we hope to analyze a much larger data set In Figure 8.11b, we see a similar pattern when comparing PNR of different temporal granularities, i.e., different values of T (Section 8.3.3).

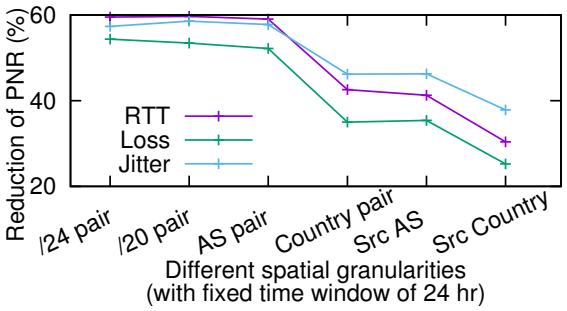
Relay usage: Figure 8.11c shows reduction of PNR when a subset of (least used) relays is excluded. We see that the contribution of benefits from different relay nodes are highly skewed. Removing 50% of the (least used) relays causes little drop in VIA's gains. This suggests that new relays should be deployed carefully in future.

8.4.5 Real-World Controlled Deployment

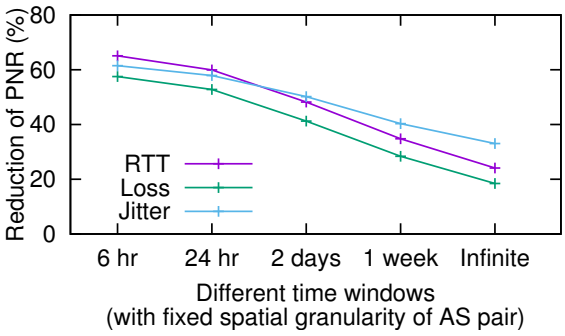
We implemented and deployed a prototype containing the relevant components of VIA at a small scale using modified Skype clients and using Skype's production relays. The central controller of our prototype (Figure 8.1), deployed on the public Microsoft Azure cloud, aggregated performance measurements from instrumented Skype clients and implemented the relay selection algorithm. The instrumented Skype clients contacted the controller to decide which of the relays of Skype, if any, to use for their calls. We deploy the instrumented client on 14 machines across Singapore, India, USA, UK and Sri Lanka. Overall, we required minimal modifications to the Skype client.

The controller also orchestrated each client to make calls to the other clients. In total, it created around 1000 calls between 18 caller-callee pairs. Specifically, it instructed each caller-callee pair to make (short) back-to-back calls using $9-20$ different relaying options, $4-5$ times each. Since our testbed is at a small scale, such back-to-back calling provides us with high density performance samples between source-destination pairs through many different relays. We use these samples to perform a controlled experiment on VIA's relaying heuristic with accurate ground truth. For simplicity, we omit the direct path as an option.

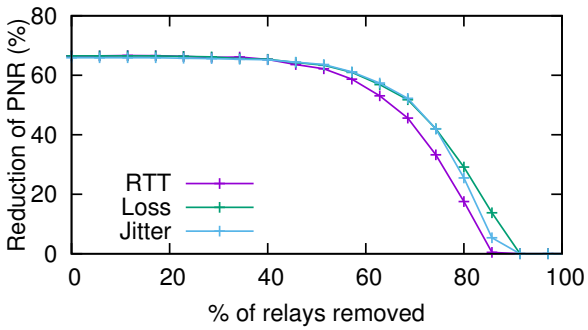
The results are shown in Figure 8.12, where each curve shows the CDF of "sub-optimality" of VIA's performance on each call, defined by $\frac{\text{Perf}_{\text{VIA}}-\text{Perf}_{\text{oracle}}}{\text{Perf}_{\text{oracle}}}$. We found that VIA's relaying decision is within 20% of an oracle's performance for 70% of the calls. Note that this is despite



(a) Impact of spatial granularity



(b) Impact of temporal granularity



(c) Impact of relay deployment

Figure 8.11: Sensitivity analysis of VIA improvement. Figure 8.11a and 8.11b compares PNR under different control granularities. Figure 8.11c shows PNR when some of the (least used) relays are excluded.

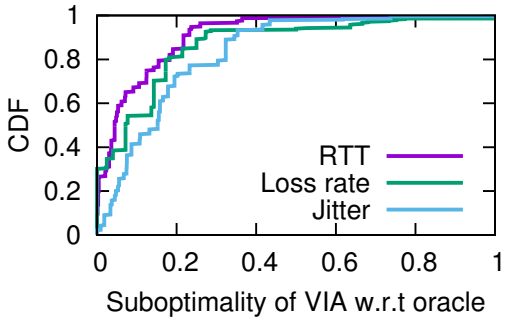


Figure 8.12: Deployment results. CDF, over calls, of sub-optimality (lower is better) of VIA's performance.

picking the *best* relay (i.e., sub-optimality of 0) for no more than 30% of the calls. When there are multiple relaying options with similar performance, temporal fluctuations may lead to not always picking the best option. But VIA usually picks the option that is close in performance to the best.

8.5 Discussion

Cost of centralized control in VIA: Our pilot deployment and client modifications suggest a feasible path to a large-scale deployment from a software update and engineering perspective. One potential concern, however, is the scalability and responsiveness of the control platform. On the one hand, VIA introduces minimal per call overhead, since the client-controller communication need only consist of one measurement update and one control message exchange per call and can be further reduced if the clients cache the best relaying options. On the other hand, handling a large number of call connections at one logical controller presents a scalability challenge, though partitioning techniques provide a good starting point. Also, we conjecture that approaches similar to the split-control architecture employed in C3 [101] might offer a scalable realization, since the measurement and control exchange of the C3 controller (which directs clients to video CDNs) is similar to the measurement and control needed for a large-scale VOIP relay server.

Hybrid reactive decentralized approaches: A natural alternative to relay selection is to simply have clients try a list of relay options sequentially or in parallel, and pick the best option. Such an approach may be good enough for long-lived calls. This would avoid the overhead of data collection and generating the network map. However, as we discussed earlier, this may not be feasible given the large search space of relaying options. An interesting hybrid approach is using the prediction-guided exploration observations as a means to *prioritize or prune* this approach. We intend to explore this approach going forward.

Active Measurements: While our current solution relied entirely on passive measurements from client calls, there is an opportunity to augment it with *active* measurements (by making mock calls between users or from users to relays), especially since the client software can be readily controlled to make them. Active measurements can be intelligently orchestrated to fill "holes" in the passively obtained measurements, thereby making our prediction-guided exploration (both its aspects—tomography as well as bandit solution) more effective. Doing so will require considering the additional load imposed on the clients due to the collection.

8.6 Related Work

Overlay routing: Overlay networking has been explored in a variety of contexts, such as virtual private networks (VPNs) and multicast [55, 93, 164]. Of interest to us here is work focused on overlay routing with a view to improving performance [47, 180]. This work showed that performance in terms of network metrics such as delay and packet loss, and also reliability, could be improved by using an overlay path that traverses well-chosen waypoints.

Despite this promise, overlay routing for performance gains has not seen much adoption in practice, for several reasons including the last-mile performance bottlenecks encountered in using client nodes as peers and the policy issues involved in turning stub networks (e.g., university campus networks) into de facto transit networks. Perhaps most importantly, these efforts involved building up overlay networks from scratch, both in terms of physical infrastructure and network probing, which limited their scale.

Our work revisits the idea of overlay routing in the context of (a) global-scale managed networks, so the global infrastructure already exists and need not be built up from scratch, and (b) a large-scale interactive real-time service, Skype, which provides both a compelling need for improving performance and (passive) measurements to obviate the need for active network probing.

Evolution of AV conferencing services: The architecture of audio-video conferencing services has been evolving, with a trend towards leveraging cloud resources. A case in point is Skype, which started off with a peer-to-peer approach to NAT and firewall traversal, with some well-connected clients with public IP addresses serving as super-nodes [133]. However, in the recent years, Skype has moved to a hybrid model [221], with some super-nodes hosted in the cloud [1]. It has been reported that Google Hangouts uses relays in the cloud for all calls, and moreover also has streams traverse the cloud backbone from one relay to another [221].

Our work is in line with these trends, but focused on performance rather than NAT/firewall traversal. Also, since we focus on managed networks, being selective in which streams are routed via the cloud is crucial in our context.

CDN server selection: Optimal server selection is a much-studied problem, especially in the context of content distribution networks [209, 216]. The main considerations in the selection process are typically proximity of the client to replicas and the load on the replicas. The main distinction of our work is our focus on client-to-client communication, which means that relay selection needs to focus on end-to-end performance rather than just between the cloud edges and the client.

Internet performance prediction: There is a large body of work on Internet performance prediction [98, 148, 157], with a focus on metrics such as bandwidth, delay, and packet loss rate. The general approach is to probe the network selectively, at chosen times and along chosen paths, and then to use the measurements to either embed the network nodes in a coordinate space [83] or estimate the performance of network segments using network tomography techniques [69]. Since we have access to network metrics for a large volume of calls, our work focuses on leveraging this data rather than performing active measurements.

Measurement studies: Over the years, there have been a number of measurement studies of large Internet services, including web sites [160], CDNs [178], and video-on-demand streaming [77, 88]. There have also been studies of audio-video conferencing by working outside the system, say by running active measurements to Skype super-nodes [220] or sniffing traffic in modest-size deployments [221]. To our knowledge, this work is the first study of a commercial VoIP service at scale by directly working with end-to-end performance metrics recorded by the communicating peers themselves.

Estimating VoIP Quality: Several models have been proposed and studied for estimating VoIP quality, typically the Mean Opinion Score (MOS), based on network performance metrics [72, 73, 75, 80, 220]. These models vary in the particular network metrics and codecs they consider. In Section 8.2, we used the model proposed in [80], which is based on the E-Model defined by the ITU [20].

8.7 Summary

By some estimates, the call volume of Internet telephony has surpassed that of traditional telephony. Given its importance, in this chapter, we have applied DDN paradigm to improve VoIP QoE. To mitigate calls with poor network performance, we revisit the classical overlay network techniques with the emerging managed overlay networks operated by large cloud providers. Calls between users with poor network conditions can be *selectively relayed* via the managed overlay network. To fully realize the benefit of a managed overlay infrastructure, we have presented the design of VIA, a system that dynamically selects a close-to-optimal relay path in the managed overlay for each call.

In doing so, VIA addresses a missing piece in previous DDN-based solutions (e.g., CFA or Pytheas): data sparsity caused by a large decision space. The key insight of VIA to address this challenge is that, for each pair of caller ISP and callee ISP, there is a *small and stable subset* of relays that almost always contains the best relay. Inspired by the insight, VIA uses a guided exploration procedure using predicted performance derived from end-to-end measurements collected by the clients, while dealing with variances in real-world estimates and keeping the volume of relayed calls within a budget. Data-driven evaluation shows that VIA improves call quality by 45%, which closely matches the potential benefits indicated by an oracle relay selection strategy that has hindsight information.

Chapter 9

Lessons, Limitations, and Future Work

In this chapter, we conclude the dissertation by summarizing its key contributions (Section 9.1), the lessons learned from our implementation and deployment of the proposed solutions (Section 9.2), and fundamental limitations of our solutions (Section 9.3). We will end with identifying the key future research topics and outlining a research agenda to help the networking community recognize and embrace the data-driven networking paradigm (Section 9.4).

9.1 Summary of Contributions

A historical perspective: The past decades have seen intense research efforts towards ensuring high quality of Internet applications by following two traditional approach: the *in-network* approach and the *endpoint-based* approach. To optimize QoE under dynamic network conditions, ideally we need to accurate information regarding both *user-perceived QoE* and *network conditions*, but both in-network and endpoint approaches have fundamental limitations on at least one front, and are ill-suited to meet QoE requirements of today's applications. On one hand, the in-network approach has limited visibility to user-perceived QoE, and requires costly re-architecting ISPs' infrastructure. On the other hand, the endpoint approach has to infer network conditions from limited, and often noisy, local information of individual end users. As a result, we have seen that a substantial fraction of video viewers and VoIP users suffer from suboptimal quality.

Our thesis: The key contribution of this dissertation is to improve application QoE by bridging the long-standing gap between the visibility to user-perceived QoE and the visibility to network conditions. Our thesis is that *one can substantially improve QoE by maintaining a global view of up-to-date network conditions based on the QoE information collected from many endpoints*. In essence, we revisit the question of "where to implement the functionality of QoE optimization", and demonstrate the feasibility and benefit of a different design choice, called data-driven networking (DDN): one can optimize QoE by using a logically centralized controller that retains the visibility to QoE while attaining a global view of real-time network conditions by consolidating information from many endpoints.

Challenges: This dissertation provides a suite of solutions to make DDN practical by addressing its two fundamental challenges: (1) the need for *expressive models* to capture complex relations

among sessions who share similar QoE-determining factors, and (2) the need for *scalable control platforms* to make real-time decisions with fresh data from geo-distributed clients.

Solutions inspired by the insight of persistent critical structures: The key insight underlying this dissertation is that there are some domain-specific *persistent critical structures* in the relationship among session-level features, decisions, and QoE.

- *Expressive models:* At a high level, the persistent critical structures allow us to build expressive models that can identify a subset of network sessions with similar QoE-determining factors and a subset of decisions that are most promising. For instance, CFA (Chapter 5) and DDA (Chapter 6) use a small subset of critical features on which a session's QoE really depends to discover the video sessions who have similar quality. VIA (Chapter 8) uses Guided Exploration to reduces the large decision spaces to a small subset of the most promising relay paths. Moreover, the fact that these structures tend to persistent on longer timescales than QoE itself suggests that we can use a longer time window and learn these structures from more history data.
- *Scalable control platforms:* Because the persistent critical structures often correlate with network locality, they enable an effective spatial decomposition of the global data-driven process into smaller-scale subprocesses which are naturally amenable for a scale-out implementation in today's cloud infrastructure. Pytheas (Chapter 7) uses Group-based E2 to decompose the global E2 process of all sessions into independent E2 subprocesses, each of which controls a group of sessions sharing network locality as well as critical features and runs in a geo-distributed frontend cluster with fresh data of these sessions. Pytheas can scale horizontally and can make decision for a population of a site like YouTube (5billion users per day) with measurement data of concurrent sessions in less than a second of delay.

QoE improvement: Using real-world deployment and offline emulation driven by large-scale measurement of real traffic, we have shown that the solutions proposed in this dissertation can lead to substantial QoE improvement in video streaming and VoIP. For instance, in video streaming, we demonstrate that CFA leads to 32% less buffering time than a baseline random decision maker, and Pytheas leads to a further 31% reduction on buffering time over CFA. In Internet telephony, we use trace-driven analysis and a small-scale deployment shows that VIA cuts the incidence of poor network conditions for calls by 45% (and for some countries and ASes by over 80%) while staying within a budget for relaying traffic through the managed overlay.

9.2 Lessons Learned

We summarize four lessons drew from the interaction with industry (Conviva and Microsoft Skype) in our attempts to deploy the proposed solutions in the real world.

Lesson #1: Need for offline evaluation One of the practical challenges we encountered in working with Conviva and Microsoft Skype has been how to demonstrate the DDN's benefit with sufficient confidence in an offline fashion, i.e., *before* implementing our solutions in the production system. A natural solution is to use simulation driven by measurement trace collected from real traffic, but this could be greatly biased by how the trace were collected; e.g.,

we would not able to evaluate the performance of Akamai in Pittsburgh if Pittsburgh users have only used other CDNs. While a full discussion on how to build an accurate trace-driven simulator (e.g., [204]) is beyond our scope, we found that it would be very useful to randomize the decisions even on a small portion of sessions when collecting trace for offline evaluation, as it allows us to estimate the outcome of alternative decisions.

Lesson #2: Need for gradual deployment Another consideration equally critical to these application providers is the ability to gradually roll-out the proposed solutions in real production settings to gain sufficient confidence before deploying them on a large scale. Besides the natural solution of A/B testing, surprisingly, we found that even deploying part of the proposed solution could reveal useful information, and potentially lead to more confidence for deploying the end-to-end solution. For instance, it is difficult to deploy VIA's online relay selection in today's Skype relaying system which does not support changing relays on a per-call basis as needed by VIA, but it is practical to deploy VIA's offline relay pruning part which updates the prediction on relay performance on a coarse timescale. While the resulting performance is suboptimal compared to that of the full system of VIA, it is sometimes necessary to trade some performance benefit for an implementation compatible to the existing systems for practical reasons.

Lesson #3: Leveraging application-specific resilience A key enabler for the scale-out and fault-tolerant architecture of Pytheas and parallel industry efforts such as C3 [101] is that we were able to exploit domain-specific properties that allows us to weaken some requirements. For instance, Pytheas tolerates controller failures by leveraging the fact that without the Pytheas controller, video players can still stream videos and quickly fall back to use the built-in local adaptation logic react to network conditions, albeit with suboptimal quality. In essence, our insight of persistent critical structures can be also viewed as an instantiation of application-level resilience. Though it is always more desirable to have a general-purpose solution, we believe it is possible to leverage application-specific resilience and engineer simpler schemes to meet the requirements of specific applications.

Lesson #4: Need for interpretability Our conversations with domain experts revealed that they were apprehensive about using complex machine learning models (e.g., SVM, PCA or neural network-based techniques) that use non-intuitive "projections" of features, because the decisions made by these algorithms are not mappable to real-world effects (e.g., which CDN), which is critical to diagnostics and incident response. Therefore, we strove to integrate our solutions with interpretability [212], allowing domain experts to combine their knowledge and diagnose prediction errors or resolve incidents. For instance, a practical benefit of CFA is that when we observe high prediction errors, we can check the critical features based on which predictions were made, and try to correlate them with known network incidents.

9.3 Limitations of Proposed Solutions

This section examines the limitations in our work.

Limitation #1: Root cause diagnosis While our algorithms provide the explanatory power of how a prediction or a decision is made, they are not design for root cause diagnosis. For instance, in CFA, critical features are not a minimal set of factors that determine the quality (i.e., root cause). That is, they can include both features that reflect the root cause as well as additional features. For example, if all HBO sessions use Level3, their critical features may include both *CDN* and *Site*, even if *CDN* is redundant, since including it does not alter predictions. The primary objective of CFA is accurate prediction; root cause diagnosis may be an added benefit. In essence, this dissertation aims at building a statistics correlation between features, decisions, and QoE to help improve QoE, but causal analysis is fundamentally different and is known to be generally more difficult than identifying statistic correlations.

Limitation #2: Skewed visibility While driving decision making by QoE observed by millions of application sessions enjoys many advantages, the view on network conditions provided by these sessions is fundamentally skewed in two aspects. (1) Since many applications such as video streaming rely heavily on CDNs to push content closer to clients, network paths of today's application sessions are skewed towards the paths from edge servers/caches to clients. (2) Application quality is generally imbalanced with most sessions having good quality, but it is the identification of bad quality that leads to more QoE improvement; e.g., problem clusters usually do not have over 50% bad-quality sessions, but the fraction is still significantly higher than the global average (Section 4.1).

Limitation #3: Handling flash crowds Flash crowds happen when many sessions join at the same time and cause part of the resources (decisions) to be overloaded. While Pytheas can handle load-induced QoE fluctuations that occur in individual groups, overloads caused by flash crowds are different in that they could affect sessions in multiple groups. A natural solution is to regroup those affected sessions immediately after a flashcrowd occurs, but Pytheas does not support such real-time update on groups. To handle flash crowds, Pytheas would need a separate mechanism to detect flash crowds and create groups for the affected sessions in real time.

Limitation #4: Agnostic to cost of switching decisions Our solutions do not consider the cost of switching decisions during the course of an application session. While they are reasonable to today's DASH-based video streaming protocols [11], in which switching bitrate and CDN merely needs a change of in the HTTP request, other applications (e.g., VoIP) may have significant cost when switching decisions in the middle of a session, so to avoid excessive switching of decisions, the control logic must not too sensitive to QoE fluctuations. Moreover, our solutions are agnostic to the cost of switching a large portion of clients from one decision to another, and this may cause practical issues as well. For instance, content providers usually pay CDNs by the 95th percentile traffic, so we must carefully take the traffic distribution into account.

Limitation #5: Limited decision space A final limitation of the DDN approach is that its potential room of improvement is fundamentally limited by the granularity of the "control knobs" exposed by the underlying delivery system. For instance, in video streaming, our solutions currently select resources at the CDN granularity. This means we cannot do much if the CDN

redirects the client based on its location and the servers the CDN redirects the client to are congested. However, if the client were able to specify the server to stream from, we could avoid the overloaded servers and improve quality.

9.4 Future Work

The application of data-driven paradigm in networked and distributed systems is vast, and still in its infancy. We believe this dissertation is merely a beginning and in the near future, there will be tremendous opportunities for data-driven techniques, driven by both "use pulls" (high QoE/performance requirements and the increasing complexity of networked systems) and "technology pushes" (availability of big data in networking and new data analytics capability).

This section identifies two broad research directions to explore this confluence of data-driven paradigms and networked and distributed systems.

- The data-driven paradigm inspires rethinking many classic problems in networking as well as enabling new services (Section 9.4.1).
- As diverse applications realize the benefit of the data-driven paradigm, there will be an increasing need for a common, principled architecture to address common challenges and extract reusable design principles. (Section 9.4.2).

9.4.1 Rethinking Classic and New Challenges in Networking

End-to-end adaptation: As the Internet grows more complex and diverse, it is untenable for traditional end-to-end protocols, such as TCP congestion control, to react in real time to changes in network conditions and resource availability based on local information. We believe data-driven techniques offer a promising alternative. Consider TCP; we can envision a new service that aggregates real-time performance of similar TCP connections and *predicts* the largest window size for which a TCP connection will not experience congestion losses. This service can potentially lead to better TCP performance than many state-of-the-art techniques, as it needs little active probing and can train the logic in near real time, rather than offline. An early promise of this approach was shown in our prior work of CS2P [199] which can accurately predict HTTP throughput using information of similar HTTP sessions, and CS2P could potentially be extended to inferring optimal window sizes from the HTTP throughput prediction.

Internet traffic map: Many Internet services can benefit from a traffic map service that can predict performance (e.g., available bandwidth) of any network path, but prior efforts towards such a service suffer from limited coverage, high probing overhead, or significant data staleness. Fortunately, passive measurements of applications like video streaming offer a new enabler for such a service, as they provide real-time measurements of network state from millions of vantage points without incurring any probing overhead! My prior work [125, 198] shows it is feasible to predict coarse-grained performance metrics (e.g., RTT between ISPs) by mapping video quality to simple network models. Extending these maps to predicting fine-grained (e.g., link-level) metrics provide an interesting next step.

Eliminating biases in trace-driven evaluation: Before implementing any DDN-based algorithm, the first question asked by any content provider is always "can I quantify how much DDN would actually improve my application's quality?" A natural solution is to use the trace already available to content providers to extrapolate the counterfactual outcome of a different algorithm, but this is challenging, since the data could be biased by the trace collection process or could have high variance due to data sparseness. This is known in ML literature as *off-policy evaluation* problem, and a promising solution is the recently proposed *doubly robust* (DR) estimator [91]. While the DR is a promising starting point, there are network-specific factors that it does not consider. For instance, the DR estimator will not identify quality degradation due to server overload, if such overload never happens in the dataset.

Data-Driven Configurations in Internet of Things: The past few years have witnessed a dramatic increase in Internet of Things (IoT) devices and their applications. One distinctive feature of these IoT applications is that they emphasize more on data analytics that has diverse requirements (e.g., delay sensitive vs. accuracy sensitive), and operates on devices of heterogeneous capacity and power duration (e.g., general-purpose machines vs. specialized hardware). Moreover, as IoT devices need to constantly interact with users, their workloads are more dynamic, if not more unpredictable, than traditional Internet hosts. Fortunately, the underlying techniques used in IoT data analytics offer many "knobs" that could potentially cope with such a heterogeneity; e.g., tuning hype-parameters of deep neural networks, flexible consistency guarantees in database, and various types of cloud resources. The challenge, however, is how to dynamically identify the configurations that are suitable to each IoT analytics task at any point of time. A case in point is the selection of configurations in surveillance video analytics [228], where it is challenging to strike a dynamic balance between timeliness and fidelity for queries with diverse requirements. Drawing on a parallel to similar problems (e.g., parameter selection in congestion control [217] and cloud configuration selection [44], we argue we should leverage data-driven techniques to learn the best configurations for these knobs based on real-time workload and monitoring data. Despite these conceptual similarity, IoT data analytics poses system challenges that have not been fully understood; e.g., how to scale the process to millions of devices who has limited bandwidth and capacity.

9.4.2 Towards a Principled Architecture for Data-Driven Networking

Custom Data Analytics Stack for Networking: While data-driven techniques can be applied to many problems, there are *common* challenges across these applications. First, while it is tempting to use general-purpose techniques to analyze networking data, one must take into account *network-specific knowledge* to avoid undesirable outcomes; e.g., to optimize overall performance, ESPN may use small ISPs to use suboptimal CDNs and let Comcast users use optimal CDNs, causing a reverse network neutrality violation in which content providers discriminate against ISPs! Second, one also needs to leverage *application-specific resilience*; e.g., to probe both optimal and suboptimal servers without affecting quality, a video player can use the optimal server to fetch most content, but use suboptimal servers when the buffer is full. Therefore, we envision a *custom data analytics stack for networking* built on top of state-of-the-art analytics stacks (e.g., Spark), which leverages network-/application-specific opportunities, and offers ab-

stractions to express network/application concepts (e.g., sessions) and requirements (e.g., data staleness). It is helpful to draw a parallel to the parallel efforts of building an analytics stack for graph processing [104], where researchers realized the limitations of Spark in supporting graph processing operations, and customized Spark and built a software layer over it to support APIs specific for graph processing.

Architecture of Cross-Provider Data Sharing: So far our work has been primarily within the scope of a single service provider. Looking forward, we argue that there is greater room for improving application QoE by bringing *more* service providers into the loop. These providers can be ISPs, CDNs, and cloud services, whose revenue is also driven, albeit indirectly, by high QoE. Unfortunately, a fundamental limitation of today's federated Internet structure is that many problems arise because each service provider has little visibility into either the QoE of its sessions or the decisions made by others. For instance, ESPN clients use HTTP-based bitrate-adaptation video players, which can select both CDN (e.g., Akamai and Limelight) and bitrate, and in many cases these players perform poorly because ESPN cannot know whether the bottleneck link is at edge ISPs (e.g., Comcast) or CDN servers (e.g., Akamai). Now, if Comcast shares with ESPN additional information that attributes bottlenecks to Comcast rather than Akamai, the players can react to ISP congestion by lowering the bitrate, rather than blindly switching between Akamai and Limelight. Building on this intuition and our early work [123], we argue that the fundamental problem lies in the limited interfaces between service providers, and that a principled approach to re-architecting these interfaces with explicit QoE optimization in mind is needed. We envision an *Experience-Oriented Network Architecture* (*EONA*), where service providers share QoE information and key configuration changes in order to optimize user-perceived QoE in a coordinated way, while still keeping the federated structure. EONA can be realized without changing the data/control plane protocols; instead, it adds new interfaces between the "controllers" of service providers (e.g., an SDN controller, a cloud orchestrator, or a global video control plane [147]). Despite EONA's potential, there are many open-ended questions: How to incentivize EONA to attract service providers? What is the minimal set of information that has to be shared? How to preserve privacy when sharing data between providers?

Infonomics of Network Measurement Data: A key practical challenge to democratize the benefit of data-driven paradigm in networking is that most measurement data are independently collected and maintained by individual domains (e.g., video sites, web sites, ISP, CDNs). It would be impractical to assume they will voluntarily share the insight extracted from their data for a common goal. While prior studies have largely focused on how to ensure data privacy by techniques like differential privacy [71, 162]. they have overlooked many key issues, including the incentive structure of data sharing, and how data fidelity (timeliness, granularity, etc) affect the willingness of others to use these data. We argue that a more effective and comprehensive solution is to develop a marketplace of network measurement data, where the key is a *pricing* framework to quantify the value of these network measurement data as well as the incentive of others to purchase them as oppose to learning it from what they already have. Such marketplace of network measurement data can be viewed as an instance of infonomics [138], an emerging notion in economics to measure, manage and monetize data as a real asset.

9.5 Final Remark

The past decade has witnessed the coming of age of data-driven paradigm in various aspects of computing (partly) empowered by advances in networked and distributed systems (cloud computing, MapReduce, etc). The overarching argument of this dissertation is that the benefits can flow the opposite direction as well: *networked systems can be improved by data-driven paradigm* – by leveraging a dramatically increased amount of measurement data and the state-of-the-art ML and large-scale data analytics techniques, we can unleash the "unreasonable effectiveness of data" in networking.

Bibliography

[1] Microsoft: Skype runs on Windows Azure; SkyDrive up next. http://www.zdnet.com/article/microsoft-skype-runs-on-windows-azure-skydrive-up-next/. 2.2.2, 8.6

[2] SmoothStreaming Protocol. http://go.microsoft.com/?linkid=9682896. 2.2.1, 2.3.2

[3] Akamai HD Adaptive Streaming. http://wwwns.akamai.com/hdnetwork/demo/index.html. 2.3.2

[4] Apache HTTP Server Benchmarking. https://httpd.apache.org/docs/2.4/programs/ab.html. 7.7.2

[5] Artiza Networks. http://www.artizanetworks.com/. 1, 1.2, 3.1.1

[6] White paper: Cisco VNI Forecast and Methodology, 2015-2020. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, . 1

[7] Quality of Service for Voice over IP. http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.pdf, . 1, 2.1.2, 4.2.1

[8] CloudLab. https://www.cloudlab.us/. 7.7

[9] Conviva Inc. http://www.conviva.com/, . 1, 5.1.1

[10] Personal communication with aditya ganjam from conviva, who is an expert on video qoe., . 5.5.2

[11] Overview of mpeg-dash standard. http://dashif.org/mpeg-dash/, . 9.3

[12] dash.js. https://github.com/Dash-Industry-Forum/dash.js/wiki, . 7.7.1

[13] Source code of Pytheas. https://github.com/nsdi2017-ddn/ddn. 7.6, 7.7

[14] Measuring Broadband America 2014. https://www.fcc.gov/measuring-broadband-america/2014/validated-data-fixed-2014, . 6, 6.1.2

[15] 2014 Measuring Broadband America Report Technical Appendix. http:

//data.fcc.gov/download/measuring-broadband-america/2014/Technical-Appendix-fixed-2014.pdf, . 6.1.2

[16] Adobe HTTP Dynamic Streaming. www.adobe.com/products/hds-dynamic-streaming.html. 2.2.1

[17] Apache HTTP Server Project. https://httpd.apache.org/. 7.6

[18] Information Gain. http://www.autonlab.org/tutorials/infogain11.pdf. 3

[19] G.114: ITU Recommendation of One-way Transmission Time. https://www.itu.int/rec/T-REC-G.114/en, . 1, 2.1.2, 4.2.1

[20] G.107: The E-Model, a computational model for use in transmission planning. https://www.itu.int/rec/T-REC-G.107-201506-I/en, . 8.6

[21] Apache Kafka. https://kafka.apache.org/. 7.6

[22] Mail Service Costs Netflix 20 Times More Than Streaming. https://www.techspot.com/news/42036-mail-service-costs-netflix-20-times-more-than-streaming.html. 2.3.2, 6.1.1

[23] Apple QuickTime. www.apple.com/quicktime/download/. 2.2.1

[24] Quova. http://developer.quova.com/. 7

[25] Real-Time Messaging Protocol. www.adobe.com/devnet/rtmp.html. 2.2.1

[26] The ACM SIGCOMM Workshop on QoE-based Analysis and Management of Data Communication Networks (Internet-QoE 2016). http://conferences.sigcomm.org/sigcomm/2016/qoe.php. 3.1.2

[27] Skype Users Hit 2 Billion Minutes A Day Milestone. http://www.silicon.co.uk/workspace/skype-users-2-billion-minutes-112054. 1

[28] Spark. http://spark.incubator.apache.org/, . 1.2, 3.1.1, 5.4.1, 5.8, 7.6

[29] Apache Spark Streaming. http://spark.apache.org/streaming/, . 7.6

[30] The Weka Manual 3.6.10. https://katie.mtech.edu/classes/csci347/Resources/WekaManual-3-6-10.pdf. 5.1.2, 2, 6.3.1

[31] The ACM SIGCOMM 2013 Workshop on Future Human-Centric Multimedia Networking (FhMN 2013). http://conferences.sigcomm.org/sigcomm/2013/fhmn.php, . 1

[32] The ACM SIGCOMM Workshop on QoE-based Analysis and Management of Data Communication Networks (Internet-QoE 2016). http://conferences.sigcomm.org/sigcomm/2016/qoe.php, . 1

[33] The ACM SIGCOMM 2012 Workshop on Measurements Up and Down the Stack (W-MUST 2012). http://conferences.sigcomm.org/sigcomm/2012/wmust.php, . 1

[34] Recommended Upload Encoding Settings. https://support.google.com/

youtube/answer/1722171?hl=en, . 1, 6.3.2

[35] Youtube Statistics. http://fortunelords.com/youtube-statistics/, . 7.7.2

[36] How many servers does youtube. https://atkinsbookshelf.wordpress.com/tag/how-many-servers-does-youtube/, . 7.7.2

[37] Vijay Kumar Adhikari, Yingying Chen, Sourabh Jain, and Zhi-Li Zhang. Where Do You 'Tube'? Uncovering YouTube Server Selection Strategy. In *ICCCN*, 2011. 2.3.2

[38] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Volker Hilt, , and Zhi-Li Zhang. A Tale of Three CDNs: An Active Measurement Study of Hulu and Its CDNs. In *IEEE Global Internet Symposium*, 2012. 5.5.2

[39] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the Monster: A Fast and Simple Algorithm for Contextual Bandits. In *ICML*. 7.8

[40] Alekh Agarwal, Sarah Bird, Markus Cozowicz, Luong Hoang, John Langford, Stephen Lee, Jiaji Li, Dan Melamed, Gal Oshri, and Oswaldo Ribas. A Multiworld Testing Decision Service. *arXiv preprint arXiv:1606.03966*, 2016. 7.8

[41] Vaneet Aggarwal, Emir Halepovic, Jeffrey Pang, Shobha Venkataraman, and He Yan. Prometheus: Toward Quality-Of-Experience Estimation for Mobile Apps from Passive Network Measurements. In *MobiCom*, 2014. 2.3, 2.3.3, 5.1.1, 5.8

[42] Saamer Akhshabi, Ali C Begen, and Constantine Dovrolis. An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP. In *ACM MMSys*, 2011. 2.3.2

[43] Saamer Akhshabi, Lakshmi Anantakrishnan, Constantine Dovrolis, and Ali C. Begen. What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth? In *NOSSDAV*, 2012. 2.3.2

[44] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics. In *NSDI*, 2017. 2.4.2, 9.4.1

[45] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data Center TCP (DCTCP). In *ACM SIGCOMM CCR*, volume 40, pages 63–74, 2010. 2.3.2

[46] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee-Pink Tan. Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications. *IEEE Communications Surveys & Tutorials*, 16(4):1996–2018, 2014. 2.4.1

[47] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *SOSP*, 2001. 2.3.2, 8.6

[48] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256, 2002. 7.2, 7.4.2, 7.8, 8.3.5

[49] Peter C Austin. An Introduction to Propensity Score Methods for Reducing the Effects of Confounding in Observational Studies. *Multivariate Behavioral Research*, 46(3):399–

424, 2011. 5.5.2

[50] Athula Balachandran, Vyas Sekar, Aditya Akella, and Srinivasan Seshan. Analyzing the Potential Benefits of CDN Augmentation Strategies for Internet Video Workloads. In *IMC*, 2013. 5.1.1, 5.8

[51] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a Predictive Model of Quality of Experience for Internet Video. In *SIGCOMM*, 2013. 1, 5.8, 6.4

[52] Athula Balachandran, Vaneet Aggarwal, Emir Halepovic, Jeffrey Pang, Srinivasan Seshan, Shobha Venkataraman, and He Yan. Modeling Web Quality-Of-Experience on Cellular Networks. In *MobiCom*, 2014. 2.3, 2.3.3

[53] Hari Balakrishnan, Mark Stemm, Srinivasan Seshan, and Randy H Katz. Analyzing Stability in Wide-Area Network Performance. In *SIGMETRICS*, 1997. 2.3.2, 6.4

[54] Hari Balakrishnan, Hariharan S Rahul, and Srinivasan Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *ACM SIGCOMM CCR*, volume 29, pages 175–187, 1999. 2.3.2

[55] S. Banerji, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *SIGCOMM*, 2002. 2.3.2, 8.6

[56] Yanan Bao, Xin Liu, and Amit Pande. Data-Guided Approach for Learning and Improving User Experience in Computer Networks. In *Asian Conference on Machine Learning*, 2015. 7.8

[57] Salman A Baset and Henning Schulzrinne. An Analysis of the Skype Peer-To-Peer Internet Telephony Protocol. *arXiv preprint cs/0412017*, 2004. 2, 2.3.2

[58] Abdelhak Bentaleb, Ali C Begen, and Roger Zimmermann. SDNDASH: Improving QoE of HTTP Adaptive Streaming Using Software Defined Networking. In *ACM Multimedia*. 2.3, 2.3.1

[59] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian OConnor, Pavlin Radoslavov, and William Snow. ONOS: Towards an Open, Distributed SDN OS. In *HotSDN*. 2.3.1

[60] Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah. Adtributor: Revenue Debugging in Advertising Systems. In *NSDI*, 2014. 5.8

[61] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 2.4.1, 3.3.1

[62] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An Architecture for Differentiated Services. Technical report, RFC 2475, 1998. 2.3.1

[63] Anna Bouch, Allan Kuchinsky, and Nina Bhatti. Quality Is in the Eye of the Beholder: Meeting Users' Requirements for Internet Quality of Service. In *ACM CHI*, 2000. 4.1.1

[64] Robert Braden, David Clark, and Scott Shenker. Integrated Services in the Internet Architecture: An Overview. Technical report, RFC 1633, 1994. 2.3.1

[65] Lawrence S Brakmo, Sean W O'Malley, and Larry L Peterson. *TCP Vegas: New Techniques for Congestion Detection and Avoidance*, volume 24. ACM, 1994. 2.3.2

[66] Michael Butkiewicz, Daimeng Wang, Zhe Wu, Harsha V Madhyastha, and Vyas Sekar. Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices. In *NSDI*, 2015. 1.1, 2.3.2

[67] Carlo Caini and Rosario Firrincieli. TCP Hybla: A TCP Enhancement for Heterogeneous Networks. *International Journal of Satellite Communications and Networking*, 22(5): 547–566, 2004. 2.3.2

[68] Wei Cao, Jian Li, Yufei Tao, and Zhize Li. On Top-K Selection in Multi-Armed Bandits and Hidden Bipartite Graphs. In *NIPS*, 2015. 8.3.3

[69] Rui Castro, Mark Coates, Gang Liang, Robert Nowak, and Bin Yu. Network Tomography: Recent Developments. *Statistical Science*, 19(3):499–517, 2004. 8.6

[70] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *IMC*, 2007. 2.3.3

[71] Ang Chen and Andreas Haeberlen. PRISM: Private Retrieval of the Internets Sensitive Metadata. In *CSET*, 2015. 9.4.2

[72] Chien-Nan Chen, Cing-Yu Chu, Su-Ling Yeh, Hao hua Chu, and Polly Huang. Modeling the QoE of Rate Changes in Skype/SILK VoIP Calls. In *ACM Multimedia*, 2012. 8.6

[73] Chien-Nan Chen, Cing-Yu Chu, Su-Ling Yeh, Hao hua Chu, and Polly Huang. Measuring the Perceptual Quality of Skype Sources. In *SIGCOMM W-MUST*, 2012. 8.6

[74] Fangfei Chen, Ramesh K Sitaraman, and Marcelo Torres. End-User Mapping: Next Generation Request Routing for Content Delivery. In *ACM SIGCOMM CCR*, volume 45, pages 167–181, 2015. 1.2, 3.1.1

[75] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. Quantifying Skype User Satisfaction. In *SIGCOMM*, 2006. 8.6

[76] Yanjiao Chen, Kaishun Wu, and Qian Zhang. From QoS to QoE: A Tutorial on Video Quality Assessment. *IEEE Communications Surveys & Tutorials*, 17(2):1126–1165, 2015. 2.3

[77] Maureen Chesire, Alec Wolman, Geoffrey M Voelker, and Henry M Levy. Measurement and Analysis of a Streaming Media Workload. In *USENIX USITS*, 2001. 8.6

[78] David R Choffnes, Fabián E Bustamante, and Zihui Ge. Crowdsourcing Service-Level Network Event Monitoring. In *ACM SIGCOMM CCR*, volume 40, pages 387–398, 2010. 5.1.1, 5.8

[79] Yang-hua Chu, Sanjay G Rao, Srinivasan Seshan, and Hui Zhang. A Case for End System Multicast. *IEEE JSAC*, 20(8):1456–1471, 2002. 2.3.2

[80] Robert G Cole and Joshua H Rosenbluth. Voice over IP Performance Monitoring. *ACM SIGCOMM CCR*, 31(2):9–24, 2001. 1, 8.6

[81] Daniel Crankshaw, Peter Bailis, Joseph E Gonzalez, Haoyuan Li, Zhao Zhang, Michael J

Franklin, Ali Ghodsi, and Michael I Jordan. The Missing Piece in Complex Analytics: Low Latency, Scalable Model Management and Serving with Velox. In *CIDR*, 2015. 5.1.1, 5.8, 7.2.1

[82] Nicola Cranley, Philip Perry, and Liam Murphy. User Perception of Adapting Video Quality. *International Journal of Human-Computer Studies*, 64(8), 2006. 2.3.3

[83] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. In *SIGCOMM*, 2004. 2.3.2, 6.4, 8.6

[84] Luca De Cicco and Saverio Mascolo. An Experimental Investigation of the Akamai Adaptive Video Streaming. *HCI in Work and Learning, Life and Leisure*, pages 447–464, 2010. 2.3.2

[85] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Skype Video Responsiveness to Bandwidth Variations. In *NOSSDAV*, 2008. 2.3.2

[86] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *SIGCOMM*, 1989. 1.1

[87] Advait Dixit, Fang Hao, Sarit Mukherjee, TV Lakshman, and Ramana Kompella. Towards an Elastic Distributed Sdn Controller. In *HotSDN*, 2013. 5.8

[88] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Antony Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the Impact of Video Quality on User Engagement. In *SIGCOMM*, 2011. 1, 1.2, 1.4, 1, 2, 2.1.1, 2.3.3, 3.1.1, 4.1.1, 5.5.1, 5.5.2, 5.8, 7.7.1, 7.7.1, 8.6

[89] Mo Dong, Qingxi Li, Doron Zarchy, P Brighten Godfrey, and Michael Schapira. PCC: Re-Architecting Congestion Control for Consistent High Performance. In *NSDI*, 2015. 1.1, 2.3.2, 2.4.2

[90] Idilio Drago, Marco Mellia, Maurizio M Munafo, Anna Sperotto, Ramin Sadre, and Aiko Pras. Inside Dropbox: Understanding Personal Cloud Storage Services. In *IMC*, 2012. 3.1.3

[91] Miroslav Dudík, John Langford, and Lihong Li. Doubly Robust Policy Evaluation and Learning. In *ICML*, 2011. 9.4.1

[92] Nandita Dukkipati, Tiziana Refice, Yuchung Cheng, Jerry Chu, Tom Herbert, Amit Agarwal, Arvind Jain, and Natalia Sutin. An Argument for Increasing TCP's Initial Congestion Window. *ACM SIGCOMM CCR*, 40(3):26–33, 2010. 2.4.1

[93] Hans Eriksson. Mbone: The Multicast Backbone. *Communications of the ACM*, 37(8): 54–61, 1994. 2.3.1, 8.6

[94] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn: an intellectual history of programmable networks. *ACM SIGCOMM CCR*, 44(2):87–98, 2014. 1.1, 2.3.1

[95] SDN Floodlight. OpenFlow Controller. *Web: https://github.com/floodlight/floodlight*. 2.3.1

[96] Sally Floyd. TCP and Explicit Congestion Notification. *ACM SIGCOMM CCR*, 24(5): 8–23, 1994. 1.1, 2.3.1

[97] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993. 1.1, 2.3.1

[98] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transaction on Networking*, 9(5):525–540, October 2001. 8.6

[99] Benjamin Frank, Ingmar Poese, Yin Lin, Georgios Smaragdakis, Anja Feldmann, Bruce Maggs, Jannis Rake, Steve Uhlig, and Rick Weber. Pushing CDN-ISP Collaboration to the Limit. *ACM SIGCOMM CCR*, 43(3), 2013. 5.8

[100] Aditya Ganjam, Vyas Sekar, and Hui Zhang. In-Situ Quality of Experience Monitoring: The Case for Prioritizing Coverage Over Fidelity. 3.1.1, 3.1.2

[101] Aditya Ganjam, Faisal Siddiqi, Jibin Zhan, Ion Stoica, Junchen Jiang, Vyas Sekar, and Hui Zhang. C3: Internet-Scale Control Plane for Video Quality Optimization. In *NSDI*, 2015. 1.2, 1.4, 3.1.1, 5.1.1, 5.1.1, 5.3, 5.5.2, 5.5.3, 5.8, 7.1, 7.2.1, 7.5.1, 7.5.4, 7.8, 8.5, 9.2

[102] Aurélien Garivier and Eric Moulines. On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems. *arXiv preprint arXiv:0805.3415*, 2008. 7.4.2, 7.8

[103] Monia Ghobadi, Yuchung Cheng, Ankur Jain, and Matt Mathis. Trickle: Rate Limiting YouTube Video Streaming. In *ATC*, 2012. 2.3.2

[104] Joseph E Gonzalez, Reynold S Xin, Ankur Dave, Daniel Crankshaw, Michael J Franklin, and Ion Stoica. GraphX: Graph Processing in a Distributed Dataflow Framework. In *OSDI*, 2014. 9.4.2

[105] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. NOX: Towards an Operating System for Networks. *ACM SIGCOMM CCR*, 38(3):105–110, 2008. 2.3.1

[106] Isabelle Guyon and André Elisseeff. Pattern Recognition and Machine Learning. *The Journal of Machine Learning Research*, 3:1157–1182, 2003. 5.7

[107] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008. 1.1, 2.3.2

[108] Emir Halepovic, Jeffrey Pang, and Oliver Spatscheck. Can You GET Me Now?: Estimating the Time-to-first-byte of HTTP Transactions with Passive Measurements. In *IMC*, 2012. 2.3, 2.3.3

[109] Osama Haq and Fahad R Dogar. Leveraging the Power of Cloud for Reliable Wide Area Communication. In *HotNets*, 2015. 3.1.3, 7.8

[110] Haibo He and Edwardo A Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. 7.1

[111] Qi He, Constantine Dovrolis, and Mostafa Ammar. On the Predictability of Large Transfer TCP Throughput. *ACM SIGCOMM CCR*, 35(4):145–156, 2005. 2.3.2, 5.8, 6.4

[112] Ningning Hu, Li Erran Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. Locating Internet Bottlenecks: Algorithms, Measurements, and Implications. In *SIGCOMM*, 2004. 2.3.2, 6.4

[113] Ningning Hu, Li Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. A Measurement Study of Internet Bottlenecks. In *INFOCOM*, 2005. 2.3.2, 6.4

[114] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *IMC*, 2012. 2.3.2

[115] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *SIGCOMM*, 2014. 5.8, 6.4

[116] Van Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM CCR*, volume 18, pages 314–329, 1988. 1.1, 2.3.2

[117] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking Named Content. In *CoNext*, 2009. 1.1, 2.3.1

[118] Akanksha Jain and Calvin Lin. Back to the Future: Leveraging Belady's Algorithm for Improved Cache Replacement. In *ISCA*. 2.4.1

[119] Manish Jain and Constantinos Dovrolis. End-To-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM Transactions on Networking*, 11(4):537–549, 2003. 2.3.2, 6.4

[120] Manish Jain and Constantinos Dovrolis. End-To-End Estimation of the Available Bandwidth Variation Range. In *SIGMETRICS*, 2005. 2.3.2, 6.4

[121] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Streaming with Festive . In *CoNEXT*, 2012. 1.1, 2.2.1, 2.3.2, 5.8, 6.1.1, 6.4

[122] Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. Shedding Light on the Structure of Internet Video Quality Problems in the Wild. In *CoNEXT*, 2013. 1, 2.1.1, 5.8

[123] Junchen Jiang, Xi Liu, Vyas Sekar, Ion Stoica, and Hui Zhang. EONA: Experience-Oriented Network Architecture. In *HotNets*, 2014. 1, 5.8, 9.4.2

[124] Junchen Jiang, Vyas Sekar, and Yi Sun. DDA: Cross-Session Throughput Prediction with Applications to Video Bitrate Selection. *arXiv preprint arXiv:1505.02056*, 2015. 1.3.3

[125] Junchen Jiang, Rajdeep Das, Ganesh Anathanarayanan, Philip Chou, Venkata Padmanabhan, Vyas Sekar, Esbjorn Dominique, Marcin Goliszewski, Dalibor Kukoleca, Renat Vafin, and Hui Zhang. Via: Improving Internet Telephony Call Quality Using Predictive Relay Selection. In *SIGCOMM*, 2016. 1, 1.2, 1.3.3, 1, 2.2.3, 3.1.1, 3.4.2, 7.1.2, 7.3, 7.4.1, 7.7.1, 7.8, 9.4.1

[126] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui Zhang. CFA: a practical prediction system for video QoE optimization. In *NSDI*, 2016. (document), 1.3.3, 3.3, 3.4.1, 3.4.2, 7.1, 7.1.1, 7.1.2, 7.2.1, 7.3, 7.4.1, 7.5.1, 7.7.1, 7.8

[127] Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. Unleashing the potential of data-driven networking. In *COMSNET*, 2017. 1.2

[128] Junchen Jiang, Shijie Sun, Vyas Sekar, and Hui Zhang. Pytheas: Enabling Data-Driven Quality of Experience Optimization Using Group-Based Exploration-Exploitation. In *NSDI*, 2017. 1.3.3

[129] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. *ACM SIGCOMM CCR*, 32(4):89–102, 2002. 2.3.1

[130] Rajat Kateja, Nimantha Baranasuriya, Vishnu Navda, and Venkata N Padmanabhan. Diversifi: Robust Multi-Link Interactive Streaming. In *CoNEXT*, 2015. 1.1, 2.3.2

[131] Naga Praveen Katta, Jennifer Rexford, and David Walker. Incremental Consistent Updates. In *HotSDN*, 2013. 2.3.1

[132] Srinivasan Keshav. *Mathematical Foundations of Computer Networking*. Addison-Wesley, 2012. 3.3

[133] Wookyun Kho, Salman Abdul Baset, and Henning Schulzrinne. Skype Relay Calls: Measurements and Experiments. In *INFOCOM Global Internet Workshop*, 2008. 2.2.2, 2.3.2, 8.6

[134] Jay Kreps, Neha Narkhede, and Jun Rao. Kafka: A Distributed Messaging System for Log Processing. In *NetDB*, 2011. 5.4.1

[135] S. Shunmuga Krishnan and Ramesh K. Sitaraman. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs. In *IMC*, 2012. 1, 1.2, 1.4, 2, 3.1.1, 4.1.1, 5.8

[136] Srisankar Kunniyur and Rayadurgam Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *ACM SIGCOMM CCR*, volume 31, pages 123–134, 2001. 2.3.1

[137] Katrina LaCurts, Jeffrey C Mogul, Hari Balakrishnan, and Yoshio Turner. Cicada: Introducing Predictive Guarantees for Cloud Networks. In *HotCloud*, 2014. 7.8

[138] Doug Laney. Infonomics: The Economics of Information and Principles of Information Asset Management. In *The Fifth MIT Information Quality Industry Symposium. Cambridge*, 2011. 9.4.2

[139] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521(7553): 436–444, 2015. 5.7

[140] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *WWW*, 2010. 5.5.2

[141] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *WWW*, 2010. 7.2.1

[142] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased Offline Evaluation of Contextual-Bandit-Based News Article Recommendation Algorithms. In *ACM WSDM*, 2011. 5.5.2

[143] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C Begen, and David Oran. Probe and Adapt: Rate Adaptation for HTTP Video Streaming at Scale. *IEEE JSAC*, 32 (4):719–733, 2014. 2.3.2

[144] XinAn Lin and Dong Zhang. Kemy: An AQM Generator Based on Machine Learning. In *ChinaCom*, 2015. 2.4.1

[145] Harry Liu, Ye Wang, Yang Richard Yang, Alexander Tian, and Hao Wang. Optimizing Cost and Performance for Content Multihoming. In *SIGCOMM*, 2012. 2.3.2, 5.1.1, 5.1.1, 5.8

[146] Hongqiang Harry Liu, Raajay Viswanathan, Matt Calder, Aditya Akella, Ratul Mahajan, Jitendra Padhye, and Ming Zhang. Efficiently Delivering Online Services over Integrated Infrastructure. In *NSDI*, 2016. 1.2, 1.3.3, 2.4.2, 3.1.1, 3.1.3, 3.3, 3.4.2, 7.1, 7.3, 7.4.1, 7.8

[147] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A Case for a Coordinated Internet Video Control Plane. In *SIGCOMM*, 2012. (document), 1, 1.3.3, 2.1.1, 2.5, 2.2.3, 2.3.2, 3.1.3, 5.1.1, 5.1.1, 5.1.3, 5.5.2, 5.8, 9.4.2

[148] Harsha V Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An Information Plane for Distributed Services. In *OSDI*, 2006. 2.3.2, 3.1.2, 6.4, 8.6

[149] Ajay Anil Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Joanne Emmons. Detecting the Performance Impact of Upgrades in Large Operational Networks. In *SIGCOMM*, 2010. 2.3.3

[150] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural Adaptive Video Streaming with Pensieve. In *SIGCOMM*, 2017. 2.4.2

[151] Charles E McCulloch and John M Neuhaus. *Generalized Linear Mixed Models*. Wiley Online Library, 2001. 7.8

[152] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM CCR*, 38(2):69–74, 2008. 2.3.1

[153] Konstantin Miller, Abdel-Karim Al-Tamimi, and Adam Wolisz. Low-Delay Adaptive Video Streaming Based on Short-Term TCP Throughput Prediction. *arXiv preprint arXiv:1503.02955*, 2015. 6.4

[154] Mariyam Mirza, Joel Sommers, Paul Barford, and Xiaojin Zhu. A Machine Learning Approach to TCP Throughput Prediction. In *SIGMETRICS*, 2007. 2.3.2, 5.8, 6.4

[155] Ricky K. P. Mok, Edmond W. W. Chan, Xiapu Luo, and Rocky K. C. Chang. Inferring the QoE of HTTP Video Streaming from User-Viewing Activities . In *SIGCOMM W-MUST*, 2011. 2.3.3

[156] Matthew K Mukerjee, David Naylor, Junchen Jiang, Dongsu Han, Srinivasan Seshan, and Hui Zhang. Practical, real-time centralized control for cdn-based live video delivery. In *SIGCOMM*, 2015. 1.2, 1.3.3, 3.1.1, 5.1.1, 5.8

[157] TS Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-based Approaches. In *INFOCOM*, 2002. 8.6

[158] Kathleen Nichols and Van Jacobson. Controlling Queue Delay. *Communications of the ACM*, 55(7):42–50, 2012. 2.3.1

[159] Ashkan Nikravesh, David R Choffnes, Ethan Katz-Bassett, Z Morley Mao, and Matt Welsh. Mobile Network Performance from User Devices: A Longitudinal, Multidimensional Analysis. In *PAM*, 2014. 2.3.2, 6.4

[160] Venkata N. Padmanabhan and Lili Qiu. The Content and Access Dynamics of a Busy Web Site: Findings and Implications. In *SIGCOMM*, 2000. 8.6

[161] Aurojit Panda, Colin Scott, Ali Ghodsi, Teemu Koponen, and Scott Shenker. CAP for Networks. In *HotSDN*, 2013. 5.8

[162] Antonis Papadimitriou, Arjun Narayan, and Andreas Haeberlen. DStress: Efficient Differentially Private Computations on Distributed Data. In *EuroSys*, 2017. 9.4.2

[163] Martin Pelikan, David E Goldberg, and Erick Cantú-Paz. BOA: The Bayesian optimization algorithm. In *Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 1999. 2.4.2

[164] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel. ALMI: An Application Level Multicast Infrastructure. In *USENIX USITS*, 2001. 2.3.2, 8.6

[165] Larry Peterson and Bruce Davie. Framework for CDN Interconnection. 2013. 5.8

[166] Louis Plissonneau and Ernst Biersack. A Longitudinal View of HTTP Video Streaming Performance. In *MMSys*, 2012. 2.3.3, 5.8

[167] Lucian Popa, Ali Ghodsi, and Ion Stoica. HTTP as the narrow waist of the future internet. In *HotNets*, 2010. 2.2.1

[168] Warren B Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, volume 703. John Wiley & Sons, 2007. 3

[169] Ravi Prasad, Constantinos Dovrolis, Margaret Murray, and KC Claffy. Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. In *IEEE Network*, 2003. 2.3.2, 6.4

[170] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. Low Latency Geo-Distributed Data Analytics. In *SIGCOMM*, 2015. 7.2.1, 7.7.2, 7.8

[171] R. Pantos. Http live streaming. Mar. 2011. IEFT Draft. 2.2.1, 2.3.2

[172] Ariel Rabkin, Matvey Arye, Siddhartha Sen, Vivek S Pai, and Michael J Freedman. Aggregation and Degradation in Jetstream: Streaming Analytics in the Wide Area. In *NSDI*, 2014. 7.8

[173] Venugopalan Ramasubramanian, Dahlia Malkhi, Fabian Kuhn, Mahesh Balakrishnan, Archit Gupta, and Aditya Akella. On the Treeness of Internet Latency and Bandwidth. In *SIGMETRICS*, 2009. 2.3.2, 6.4

[174] Philippe Rigollet and Assaf Zeevi. Nonparametric Bandits with Covariates. In *Conference on Learning Theory*, 2010. 7.8

[175] Jorma Rissanen. A Universal Prior for Integers and Estimation by Minimum Description Length. *The Annals of Statistics*, pages 416–431, 1983. 5.7

[176] Eric Rosen, Arun Viswanathan, and Ross Callon. Multiprotocol Label Switching Archi-

tecture. Technical report, RFC 3031, 2000. 2.3.1

[177] Raja R Sambasivan, Alice X Zheng, Michael De Rosa, Elie Krevat, Spencer Whitman, Michael Stroucken, William Wang, Lianghong Xu, and Gregory R Ganger. Diagnosing Performance Changes by Comparing Request Flows. In *NSDI*, 2011. 5.1.1, 5.8

[178] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An Analysis of Internet Content Delivery Systems. In *OSDI*, 2002. 8.6

[179] Stefan Savage, Neal Cardwell, and Tom Anderson. The Case for Informed Transport Protocols. In *HotOS*, 1999. 5.8

[180] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Tom Anderson. The End-to-end Effects of Internet Path Selection. In *SIGCOMM*, 1999. 2.3.2, 8.6

[181] Michael Schapira, Yaping Zhu, and Jennifer Rexford. Putting BGP on the Right Path: A Case for Next-Hop Routing. In *HotNets*, 2013. 2.3.1, 2.4.2

[182] Bernhard Schölkopf and Alexander J Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2002. 5.7

[183] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hobfeld, and Phuoc Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492, 2015. 2

[184] Muhammad Zubair Shafiq, Jeffrey Erman, Lusheng Ji, Alex X. Liu, Jeffrey Pang, and Jia Wang. Understanding the Impact of Network Dynamics on Mobile Video User Engagement. In *SIGMETRICS*. 2.3.3

[185] Muhammad Saleh Shah, Asim Imdad Wagan, and Mukhtiar Ali Unar. SAM: Support Vector Machine Based Active Queue Management. *arXiv preprint arXiv:1604.00557*, 2016. 2.4.1

[186] Deepak K Sharma, Sanjay K Dhurandher, Isaac Woungang, Rohit K Srivastava, Anhad Mohananey, and Joel JPC Rodrigues. A Machine Learning-Based Protocol for Efficient Routing in Opportunistic Networks. *IEEE Systems Journal*, 2017. 2.4.1

[187] Anirudh Sivaraman, Alvin Cheung, Mihai Budiu, Changhoon Kim, Mohammad Alizadeh, Hari Balakrishnan, George Varghese, Nick McKeown, and Steve Licking. Packet Transactions: High-Level Programming for Line-Rate Switches. In *SIGCOMM*, 2016. 2.3.1

[188] Anirudh Sivaraman, Suvinay Subramanian, Mohammad Alizadeh, Sharad Chole, Shang-Tse Chuang, Anurag Agrawal, Hari Balakrishnan, Tom Edsall, Sachin Katti, and Nick McKeown. Programmable Packet Scheduling at Line Rate. In *SIGCOMM*, 2016. 2.3.1

[189] Aleksandrs Slivkins. Contextual Bandits with Similarity Information. *The Journal of Machine Learning Research*, 15(1):2533–2568, 2014. 5.7, 7.2.1

[190] Iraj Sodagar. The Mpeg-Dash Standard for Multimedia Streaming over the Internet. *IEEE MultiMedia*, 18(4):62–67, 2011. 1.1, 2.3.2, 5.1.1, 6.1.1

[191] Han Hee Song, Zihui Ge, Ajay Mahimkar, Jia Wang, Jennifer Yates, Yin Zhang, Andrea Basso, and Min Chen. Q-score: Proactive Service Quality Assessment in a Large IPTV System. In *IMC*, 2011. 2.3.3, 5.8

[192] Kun Tan Jingmin Song, Q Zhang, and M Sridharan. Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-Speed Networks. *PFLDnet*, 2006. 2.3.2

[193] Mark Stemm, Randy Katz, and Srinivasan Seshan. A Network Measurement Architecture for Adaptive Applications. In *INFOCOM*, 2000. 1.2, 2.3.2, 3.1.2, 3.1.3, 5.1.1, 5.8, 6.4, 7.1, 7.1.1, 7.8

[194] Mark Stemm, Randy Katz, and Srinivasan Seshan. A Network Measurement Architecture for Adaptive Applications. In *INFOCOM*, 2000. 5.8

[195] Ion Stoica, Scott Shenker, and Hui Zhang. Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *SIGCOMM*, 1998. 1.1, 2.3.1

[196] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A Measurement Study of Available Bandwidth Estimation Tools. In *SIGCOMM Conference on Internet Measurement*, 2003. 2.3.2, 6.4

[197] Ao-Jan Su, David R Choffnes, Aleksandar Kuzmanovic, and Fabián E Bustamante. Drafting Behind Akamai (Travelocity-Based Detouring). In *ACM SIGCOMM CCR*, volume 36, pages 435–446, 2006. 5.8

[198] Yi Sun, Junchen Jiang, Vyas Sekar, Hui Zhang, Fuyuan Lin, and Nanshu Wang. Using Video-Based Measurements to Generate a Real-Time Network Traffic Map. In *HotNets*, 2014. 9.4.1

[199] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction. In *SIGCOMM*, 2016. 3.1.3, 3.4.2, 7.1, 7.3, 9.4.1

[200] Lalith Suresh, Marco Canini, Stefan Schmid, and Anja Feldmann. C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection. In *NSDI*, 2015. 5.8

[201] Martin Swany and Rich Wolski. Multivariate Resource Performance Forecasting in the Network Weather Service. In *ACM/IEEE Conference on Supercomputing*, 2002. 2.3.2, 6.4

[202] C-H Tai, Jiang Zhu, and Nandita Dukkipati. Learning from Imbalanced DataMaking Large Scale Deployment of RCP Practical for Real Networks. In *INFOCOM*, 2008. 2.3.1, 5.8

[203] Dixin Tang, Hao Jiang, and Aaron J. Elmore. Adaptive Concurrency Control: Despite the Looking Glass, One Concurrency Control Does Not Fit All. In *CIDR*, 2017. 2.4.1

[204] Mukarram Tariq, Amgad Zeitoun, Vytautas Valancius, Nick Feamster, and Mostafa Ammar. Answering What-If Deployment and Configuration Questions with WISE. In *ACM SIGCOMM CCR*, volume 38, pages 99–110, 2008. 9.2

[205] David L Tennenhouse and David J Wetherall. Towards an Active Network Architecture. In *DARPA Active Networks Conference and Exposition*, 2002. 1.1, 2.3.1

[206] George R Terrell and David W Scott. Variable Kernel Density Estimation. *The Annals of Statistics*, pages 1236–1265, 1992. 5.7, 5.8

[207] Guibin Tian and Yong Liu. Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming. In *CoNEXT*. 6.4

[208] Amin Tootoonchian, Sergey Gorbunov, Yashar Ganjali, Martin Casado, and Rob Sherwood. On Controller Performance in Software-Defined Networks. In *USENIX Hot-ICE*, 2012. 5.8

[209] Ruben Torres, Alessandro Finamore, Jin Ryong Kim, Marco Mellia, Maurizio M. Munafo, and Sanjay Rao. Dissecting Video Server Selection Strategies in the YouTube CDN. In *ICDCS*, 2011. 2.3.2, 5.8, 8.6

[210] Dana Van Aken, Andrew Pavlo, Geoffrey J Gordon, and Bohan Zhang. Automatic Database Management System Tuning Through Large-scale Machine Learning. In *ICDM*. 2.4.1

[211] Sudharshan Vazhkudai, Jennifer M Schopf, and Ian Foster. Predicting the performance of wide area data transfers. In *IPDPS*, 2001. 2.3.2, 6.4

[212] Alfredo Vellido, JD Martin-Guerroro, and P Lisboa. Making Machine Learning Models Interpretable. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN). Bruges, Belgium*, 2012. 4, 9.2

[213] Ashish Vulimiri, Carlo Curino, P Brighten Godfrey, Thomas Jungblut, Jitu Padhye, and George Varghese. Global Analytics in the Face of Bandwidth and Regulatory Constraints. In *NSDI*, 2015. 7.7.2, 7.8

[214] Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall. How Speedy is SPDY. In *NSDI*, 2014. 1, 2.2.1

[215] Richard Weber. On the Gittins Index for Multiarmed Bandits. *The Annals of Applied Probability*, pages 1024–1033, 1992. 2.4.2, 5.7, 7.2, 7.2.1, 7.8

[216] Patrick Wendell, Joe Wenjie Jiang, Michael J. Freedman, and Jennifer Rexford. DONAR: Decentralized Server Selection for Cloud Services. In *SIGCOMM*, 2010. 8.6

[217] Keith Winstein and Hari Balakrishnan. TCP ex Machina: Computer-Generated Congestion Control. In *SIGCOMM*, 2013. 1.1, 2.3.2, 2.4.1, 9.4.1

[218] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *NSDI*, 2013. 1.1, 2.3.2

[219] Yong Xia, Lakshminarayanan Subramanian, Ion Stoica, and Shivkumar Kalyanaraman. One More Bit is Enough. *ACM SIGCOMM CCR*, 35(4):37–48, 2005. 2.3.1

[220] Haiyong Xie and Yang Richard Yang. A Measurement-based Study of the Skype Peer-to-Peer VoIP Performance. In *IPTPS*, 2012. 8.6

[221] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. Video Telephony for End-consumers: Measurement Study of Google+, iChat, and Skype. In *IMC*, 2012. 2.2.2, 8.1, 8.6

[222] Hong Yan, David A Maltz, TS Eugene Ng, Hemant Gogineni, Hui Zhang, and Zheng Cai. Tesseract: A 4D Network Control Plane. In *NSDI*, 2007. 5.8

[223] Hao Yin, Xuening Liu, Feng Qiu, Ning Xia, Chuang Lin, Hui Zhang, Vyas Sekar, and

Geyong Min. Inside the bird's nest: measurements of large-scale live VoD from the 2008 olympics. In *IMC*, 2009. 2.3.3, 5.8

[224] Xiaoqi Yin, Vyas Sekar, and Bruno Sinopoli. Toward a Principled Framework to Design Dynamic Adaptive Streaming Algorithms over HTTP. In *HotNets*, 2014. 6.4

[225] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *SIGCOMM*, 2015. 5.8

[226] Minlan Yu, Wenjie Jiang, Haoyuan Li, and Ion Stoica. Tradeoffs in CDN Designs for Throughput Oriented Traffic. In *CoNEXT*, 2012. 5.8

[227] Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. Discretized Streams: Fault-Tolerant Streaming Computation at Scale. In *SOSP*. 1.2, 3.1.1

[228] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *NSDI*, 2017. 9.4.1

[229] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Network*, 7(5):8–18, 1993. 2.3.1

[230] Yin Zhang and Nick Duffield. On the constancy of internet path properties. In *SIGCOMM Workshop on Internet Measurement*, 2001. 2.3.2, 3.1.3, 3.3, 6.4