QoE based Management and Control for Large-scale VoD System in the Cloud

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Chen Wang

B.S., Information and Communication Engineering, Xi'an Jiaotong University, China

M.S., Information and Communication Engineering, Xi'an Jiaotong University, China

M.S., Electrical and Computer Engineering, Carnegie Mellon University, USA

Carnegie Mellon University

Pittsburgh, PA

August 2017

ABSTRACT

The Cloud infrastructure has become an ideal platform for large-scale applications, such as Video-on-Demand (VoD). As VoD systems migrate to the Cloud, new challenges emerge. The complexity of the Cloud system due to virtualization and resource sharing complicates the Quality of Experience (QoE) management. Operational failures in the Cloud can lead to session crashes. In addition to the Cloud, there are many other systems involved in the large-scale video streaming. These systems include the Content Delivery Networks (CDNs), multiple transit networks, access networks, and user devices. Anomalies in any of these systems can affect users' Quality of Experience (QoE). Identifying the anomalous system that causes QoE degradation is challenging for VoD providers due to their limited visibility over these systems.

We propose to apply end user QoE in the management and control of large-scale VoD systems in the Cloud. We present a QoE-based management and control systems and validate them in production Clouds. **QMan**, a QoE based Management system for VoD in the Cloud, controls the server selection adaptively based on user QoE. **QWatch**, a scalable monitoring system, detects and locates anomalies based on the end-user QoE. **QRank**, a scalable anomaly identification system, identifies the anomalous systems causing QoE anomalies.

The proposed systems are developed and evaluated in production Clouds (Microsoft Azure, Google Cloud and Amazon Web Service). QMan provides 30% more users with QoE above the "good" Mean Opinion Score (MOS) than existing server selection systems. QMan discovers operational failures by QoE based server monitoring and prevents streaming session crashes. QWatch effectively detects and locates QoE anomalies in our extensive experiments in production

Clouds. We find numerous false positives and false negatives when system metric based anomaly detection methods are used. QRank identifies anomalous systems causing 99.98% of all QoE anomalies among transit networks, access networks and user devices. Our extensive experiments in production Clouds show that transit networks are the most common bottleneck causing QoE anomalies. Cloud provider should identify bottleneck transit networks and determine appropriate peering with Internet Service Providers (ISPs) to bypass these bottlenecks.

ACKNOWLEDGEMENT

I have great appreciations in mind to acknowledge many people who support me along the way. I am deeply indebted to my advisor Prof. Hyong Kim for his untiring mentorship throughout my Ph.D. studies. I have learnt precious lessons in both research and life from him, which will lead me toward a perseverant, productive and prosperous career in the future. I am also indebted to my co-advisor Prof. Ricardo Morla for his continued support. Not only do I benefit from his invaluable input to my research, but also from his readily available help all along my work. I am furthermore grateful to my committee members, Prof. Anthony Rowe and Prof. Aswin C Sankaranarayanan for their interest and useful feedback to my thesis.

I would like to express my gratitude to my excellent colleagues, Jiyeon, Tiago, Andy, John, Senbo, Liang, Andrew and Ke from the ONE research group, for their comments and help during the daily work. I also owe my sincere thanks to all the staff at ECE department and CMU Portugal program. I would like to acknowledge the financial support by the FCT through the Carnegie Mellon Portugal Program under Grant [SFRH/BD/51150/2010], CMU-SYSU CIRC, SYSU-CMU IJRI. I would also like to acknowledge the Azure Research grant provided by Microsoft Corporation and AWS Cloud Credit for Research program provided by Amazon.

Finally, I would like to thank my parents for their love, patience, support and understanding. A very special thank you to my husband, Qiwei Han. You unwaveringly provide me with encouragement, support, belief and company. No matter if near or far, night or day, you always show up through Wechat, Skype, or phone, whenever I need you. I would never make it without you being with me through this journey.

GLOSSARY

1. **Cache agent** is a management process running in each video server to collect users' QoE and to learn SQS for video servers.
2. **CDN** ***C****ontent* ***D****elivery* ***N****etwork* is an interconnected system of cache servers that use geographical proximity as a criterion to deliver content to users.
3. **Client agent** is a management process running in the video player to monitor and report user's QoE at run time.
4. **Chunk** is a segment of video file that is delivered in one HTTP range request in HTTP based video streaming. Each chunk usually consists of a few seconds of video and has its own URL.
5. **Client** is the software of video player or the process running in user's device to assist video streaming.
6. **Client group** is a group of clients who connect to the same cache agent as their "closest" cache agent. The "closest" denotes the shortest in network latencies.
7. **Crash fault** are various faults that cause a service stopped.
8. **DASH** ***D****ynamic* ***A****daptive bitrate* ***S****treaming over* ***HTTP***, is an adaptive bitrate streaming technique that enables high quality streaming of video content over the Internet using HTTP protocol.
9. **IaaS** ***I****nfrastracture* ***a****s* ***a*** ***S****ervice*, is a form of cloud computing that provides virtualized computing resources over the Internet.
10. **Multi-tenancy** is an architecture in which a single instance of a software application serves multiple customers. Each customer is called a **tenant**.
11. **P2P** ***P****eer to* ***P****eer*, is a distributed application architecture that partition work loads among peers.
12. **Peer** is a participant in a P2P system, who consumes and supplies resource such as processing power, disk storage or network bandwidth.
13. **Progressive download** is the transfer of video files from a server to a client using HTTP protocol.

14.	**PM**	*Physical Machine* is a hardware-based device, such as a rack server.
15.	**QoS**	***Q****uality of* ***S****ervice,* is a quantitative measure of overall network performance seen by the users. QoS metrics include error rates, bit rate, throughput, transmission delay, availability, jitter, etc.
16.	**QoE**	Quality of Experience is a measure of a user's subjective experience with the VoD service.
17.	**QMan**	QoE based Management system for VoD in the Cloud.
18.	**QWatch**	A QoE based monitoring system for VoD in the Cloud.
19.	**QRank**	A QoE anomaly idenfication system for VoD in the Cloud.
20.	**QS**	***Q****oE* ***S****core*, is our propsed QoE based system performance metric.
21.	**Session QoE**	is the average of all chunks' QoE in a single video steaming session.
22.	**SLA**	***S****ervice* ***L****evel* ***A****greement,* is a contract between a service provider and service users that defines the level of service expected from the service provider. In our context, the service is either a Cloud IaaS service or a VoD service.
23.	**User**	is a customer who uses VoD service.
24.	**VM**	*Virtual Machine*, an emulation of an operating system or application environment that is installed on software, which imitates dedicated hardware.
25.	**VoD**	***V****ideo* ***o****n* ***D****emand*, are services that allow users to select and watch video content when they choose to.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Background and Motivation

In 2014, 52% of downstream traffic in North America were from popular Video-on-Demand (VoD) providers including Netflix, YouTube, Amazon Video, and Hulu [1]. VoD is expected to become majority of the Internet traffic. More users are expected to subscribe the VoD service and they would expect better video streaming experience. The management and control system of VoD service should be able to accommodate the continued growth of users and the increasing demand in Quality of Experience (QoE).

1.2 The architectures of large-scale VoD system

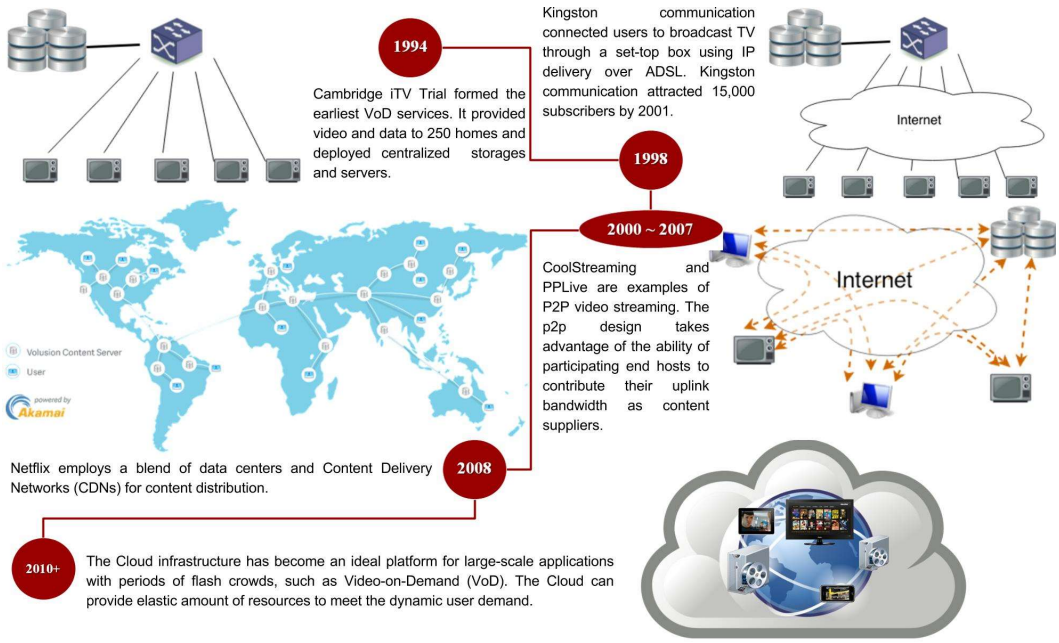


Figure 1: The Evolution of VoD Architectures

The architecture of VoD system evolves to support the growing number of users and the increasing expectation for user experience as shown in Figure 1. Back in the 1990s, a simple server-client architecture was used in early VoD systems. Centralized servers were deployed to store videos and provide streaming services. In early 2000s, Peer-to-peer (P2P) architecture was proposed and widely used to accommodate huge amount of traffic in VoD service [2][3][4]. In the P2P design, the uplink capacity of participating hosts, namely peers, was used to serve other clients. This greatly reduced the amount of bandwidth required at streaming servers in large scale. In late 2000s, VoD providers started using third party Content Delivery Networks (CDN) to provide VoD service [26]. CDN is an interconnected system of cache servers that use geographical proximity to deliver the content [5]. VoD users always stream videos from a cache server close to them to reduce the network latency in CDN. HTTP based video streaming such as progressive download and Dynamic Adaptive Streaming over HTTP (DASH) are widely supported in CDNs. Users stream videos via standard HTTP protocol. Lately the Cloud became popular as an infrastructure to provide on-demand server utilities to users anywhere anytime [7]. The Cloud infrastructure is a promising platform for VoD service because of its elasticity, reliability and cost effectiveness [8]. Netflix, as a pionior in using cloud infrastructure, started using Amazon Web Service (AWS) since 2010 [6]. However, due to the highly centralized design of Cloud datacenters, large-scale VoD systems today still have to rely on both the Cloud and the CDN to provide high quality video streaming for millions of users.

Cloud providers own CDNs as well, such as Microsoft Azure CDN, Amazon CloudFront and Google Peering & Content Delivery. As shown in Figure 2, a typical architecture of a commercial VoD system consists of servers in the Cloud, the Content Delivery Networks (CDNs),

and ISP networks connecting servers in both Cloud and CDN to end users. We use CDN to refer to both the third party CDN and the Cloud CDN.

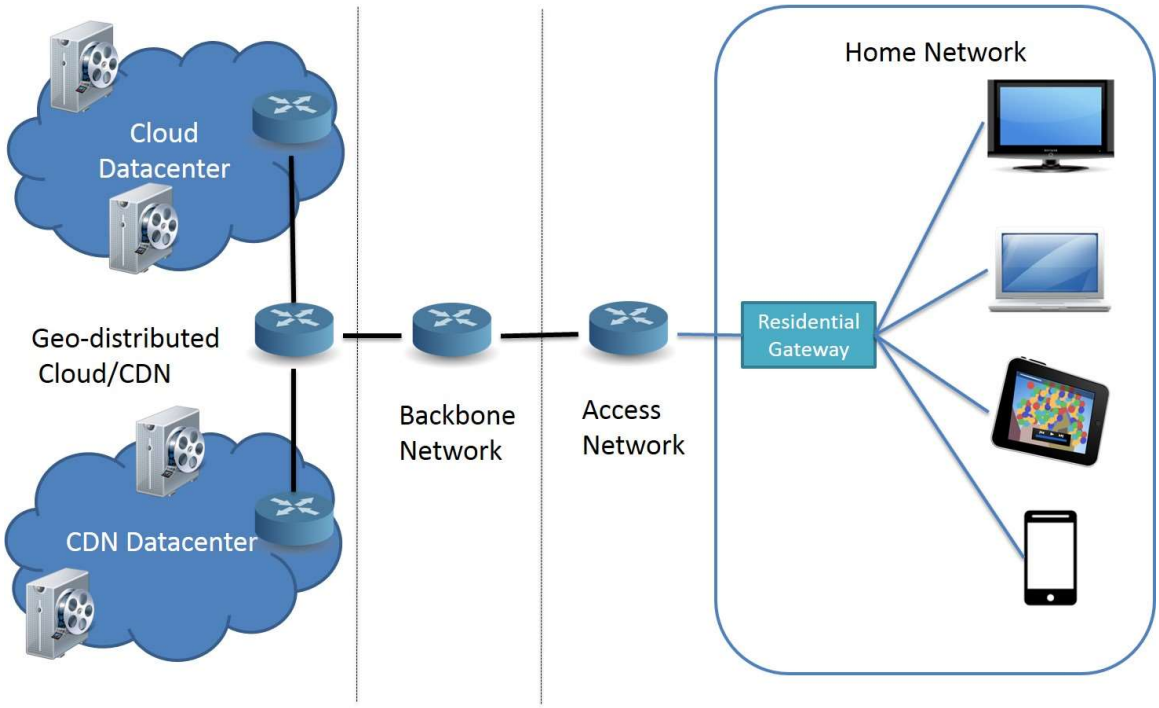


Figure 2: The Architecture of Commercial VoD System

1.3 Challenges

As VoD systems increase in scale and migrate into the Cloud, new challenges arise in the management of control of VoD system to provide consistently good Quality of Experience (QoE).

1. *Cloud is complex*. The performance of CDN and Virtual Machines (VMs) in the Cloud are dynamically changing due to interferences from other VMs on the same Physical Machine (PM). In addition, the performance metrics monitored in the Cloud do not adequately reflect QoE. Complexity of Cloud makes it difficult to manage user QoE for VoD in the Cloud.

2. *Various faults and anomalies can further degrade user QoE for VoD in the Cloud.* These faults include various types of faults in the video servers, the Cloud, the networks and the CDNs and user's device.
3. *Many systems are involved in large-scale VoD.* These systems include the Cloud/Content Delivery Network (CDN) servers and caches, the Cloud/CDN networks, multiple transit networks, access networks, and user devices. Anomalies in any one of these systems can affect users' Quality of Experience (QoE). Different stakeholders manage these systems. VoD providers have limited visibility over these systems.
4. *VoD systems have millions to billions of users [21][24].* The management and control of VoD system should be able to handle large number of users around the world.

1.3.1 Complexity of the Cloud

The Infrastructure as a Service (IaaS) in the Cloud is offered by a virtualized datacenter consisting of a cluster of physical machines (PMs) [7]. Each PM hosts multiple virtual machines (VMs) possibly belonging to different tenants. The video servers in a VoD system would be deployed in one of these VMs. Multiple VMs on the same PM are sharing physical resources such as CPU, disk, memory and network interface. Resource sharing may lead to performance interference from other VMs [8]. Cloud providers run monitoring system to record performance metrics for the resources provisioned to tenants [35]. These performance metrics are used to assist tenants to manage their applications. However, these performance metrics cannot be easily translated to end user QoE. Not all metrics showing good performance for a Virtual Machine (VM) can guarantee good QoE. Tenants in the Cloud have neither control nor visibility of the physical machines in the Cloud.

The Cloud provides CDN as IaaS with multi-tenancy. All tenants share the same Cloud CDN including edge servers and the global networks connecting edge servers. The traffic sent from one application impact the bandwith capacity of other applications. The available cache locations and caching schemes in the CDN are usually determined by the Cloud provider [10][19]. They greatly affect user QoE in VoD service. However, VoD provider as a Cloud tenant, does not have control over these. The Cloud monitoring system usually monitor the CDN using productivity metrics, such as load and throughput, but end user QoE can not be represented by these metrics.

1.3.2 Various faults and anomalies degrading QoE

Faults come from all components in a VoD system, including video servers in the Cloud, the CDNs, the networks connecting servers, and clients. According to the symptoms of the fault, we classify faults into two categories: *crash faults* and *performance degrading faults*. The most common one is the crash fault, which includes various faults that cause service stopped on one component. If a fault does not fully stop the service but temporarily disables the service or poorly performs the service, we denote the faults as performance degrading faults.

Provisioning Cloud resources on the fly may cause crash faults that are difficult to identify. The crash fault on a video server usually appear to be a server crash or hang. An exception in the server running in a VM in the Cloud can result in video server crash. VM crashes due to issues on the PM. Attacks exhausting resources in the VM (DoS attack) can also lead to a video server crash. Besides, some configuration errors would result in crash faults though the server is still running. For example, configuration errors of video file folder in the server can completely fail the service when the server is still up. Firewall configurations may block inbound traffic. However, for a server in a production cloud environment, faults rarely be simple “up” or “down”. A server can

also have various faults that cause service performance degradation. The server may respond slowly when there are memory leaks. The server would be slow in response if VMs running in the background exhausting CPU or memory in the PM. The reading of video files can be slow if there are disk errors. A server deployed in a PM with a bad network interface card would have low network connections.

The same crash and performance degrading faults occur in edge servers in the CDN. Besides, the CDNs usually balance load among multiple edge servers according to their non-disclosed internal schemes. It has been found that there are errors in the server selection schemes. These errors can degrade user QoE [10]. Recent measurement study on YouTube show that Google's CDN server selection scheme negatively impact the QoE of YouTube users at peak-load times [11]. Such faults can hardly be detected without knowing the end user QoE.

Faults in network have been studied for a long time. Common faults in a network includes configuration errors and link failures [12]. These faults usually result in network unreachability or congestion. When one network is unreachable, all traffic going through the network will be re-routed. This will result in congestion in other networks. For VoD users, network congestion always results in QoE degradation.Network faults are classified as performance degrading faults.

The software of video player is running in users' device to download, decompress and play videos for users. Various types of faults can occur in clients. We use ***"client"*** to denote the software of video player or the process running in user's device to assist video streaming. The ***"user"*** denotes the customer who watches video in VoD. A common crash fault is a user's device freezing due to CPU/memory exhaustion by some processes. A typical performance degrading fault is that the user device have low quality network connection. This would result in slow downloading speed

and bad QoE for users. For video players used in commercial VoD system, there are management processes running in the video player to assist the control [20].

1.3.3 SLA, monitoring systems and QoE

Existing Cloud platforms define service level agreements (SLAs) for different types of services. For example, Azure Cloud defines the uptime percentage as an SLA for virtual machines [76]. It refunds tenants when the monthly uptime percentage of a VM is below 95%. For Azure CDN service, Azure defines the percentage of HTTP transactions without error as its SLA. When the percentage of HTTP transactions without error is below 99.9%, it refunds users. These SLAs do not guarantee the QoE for users of video applications.

The monitoring in the Cloud do not reflect user QoE either. According to our experiments, server system metrics, such as utilization and throughput of CPU, memory, disk and network cannot fully reflect the user experience of VoD in the Cloud. Many other factors in the CDN, cloud network, transit networks, and access networks impact user QoE. Commercial CDN services offer their own monitoring systems. They log errors in the cache servers that could influence end user QoE. Common metrics are the HTTP response time, the cache request status (cache/miss), and the HTTP response code. Errors logged in the cache server do not cause all QoE degradations. Some QoE issues do not correlate with these errors either. Network systems monitor latencies and throughput. These metrics are related to user QoE but do not reflect QoE directly.

1.3.4 Multiple systems involved in video streaming

VoD systems deliver videos via many heterogeneous systems including servers, Cloud/CDN networks, transit networks, access networks, and user devices. Any of these systems

could have anomalies degrading user QoE. As their objectives vary, maintaining QoE would be challenging. For example, YouTube monitors user QoE per ISP as they assume that access ISPs to be the most likely capacity bottlenecks for high-definition (HD) video streaming [13]. Studies in a large European ISP show that Googles CDN server selection policy might be the cause of QoE degradation [11] [17]. VoD providers do not have enough visibility over other systems. The Cloud/CDN systems select servers for users. Depending on the selected server, video traffic could go through different networks to get to the access network. The Cloud network, multiple transit networks and the access network together determine the route to deliver videos. Load balancing in these networks further complicates the video routes. Video from the same server could go through different routes to the user depending on load balancing policies. Without knowing the underlying network topology for a particular video delivery, it is very difficult to find the system that incurs QoE anomalies.

1.4 Contributions

To addresses the aforementioned challenges, we propose to apply end user QoE in the management and control of large-scale VoD systems in the Cloud.

1.4.1 Using user QoE for VoD Management and Control

We believe that end-users have the best perception of server performance in terms of their QoE rather than the servers themselves. What users perceive incorporate performance of all elements, such as network delay and server response time in VoD service. Different from common system performance metrics, we develop a performance metric from end user QoE directly, referred to as QoE Score (QS) hereafter. Our proposed system collects real-time QoE from clients

in the same network that stream videos from the same server. Then the system learns and updates the QS for all systems in video streaming. In this regard, we include user QoE in the management and control of VoD.

1.4.2 QMan: The QoE based adaptive server selection system

We propose QMan, a QoE based Management system for VoD in the Cloud, which controls the server selection adaptively based on user QoE at run time. To achieve scalability, QMan deploys client agents in video players to monitor real-time QoE and deploys cache agents in video servers to monitor the status of servers. In order to improve overall user QoE, QMan applies reinforcement-learning techniques in the control of server selection. We implement QMan in 2 paradigms, a decentralized paradigm [107] that controls on cache agents and a fully distributed paradigm [55] that controls on client agents. We evaluate QMan in an experimental VoD system in Google Cloud with hundreds of users emulated in Planetlab. Results show that given the same amount of resources, QMan guarantees from 9% to 30% more users having QoE above the Mean Opinion Score (MOS) "good" level than existing measurement based server selection systems. QMan discovers operational failures by QoE based server monitoring and prevents streaming session crashes. By applying reinforcement learning techniques, QMan achieves a tradeoff between exploration and exploitation in the control of server selection, which is necessary for the highly dynimic Cloud environment. By comparing two paradigms of implementation, we show that the decentralized paradigm achieves better overall QoE with less server switches than the fully distributed paradigm. Overhead analysis proves that both QMan implementations can be adapted to large-scale systems consisting of thousands of servers.

1.4.3 QWatch: QoE based monitoring system

We propose QWatch, a scalable monitoring system for large-scale VoD in the Cloud. QWatch detects and locates anomalies using the end-user QoE in real time. We believe the end-user QoE best reflects VoD system performance. The user satisfaction is the ultimate performance measure of any complex systems. Regardless of what traditional performance parameters indicate, if the end user QoE is satisfactory, the system is deemed to be operating properly. The end-user QoE masks the complexity of understanding proper operation of VoD systems using numerous system parameters. In QWatch, the end user devices cooperate and share their QoE and path information in order to detect the locate anomalies. We validate QWatch through extensive experiments in a controlled VoD system in Microsoft Azure Cloud and Amazon CloudFront CDN. Our experiments show that QWatch correctly detects QoE anomalies that cannot be detected using various network/system metrics. QWatch also avoids false positives in anomaly detection methods based on system metrics. QWatch successfully locates QoE anomalies. We also share several insights obtained from running VoD system with 200 geographically separated users in production Cloud.

1.4.4 QRank: QoE anomaly identification system

We propose QRank, an anomaly identification system that identifies the bottleneck system causing QoE anomalies. QRank detects QoE anomalies based on QoEs monitored on users at run time. QRank discovers the underlying network topology and all systems involved in video streaming by traceroute measurements. We assume that the system with users that experience more QoE anomalies or lower QoEs is more likely to be the system causing QoE anomalies. QRank identifies the anomalous system by ranking the QoEs in different systems. We validate the

effectiveness of QRank through extensive experiments in a controlled VoD. In our experiments, we inject QoE anomalies in user device, access network, transit network, cloud network, and server to degrade user QoE. QRank correctly identifies the anomalous system in these cases. Existing work, QWatch system can only locate anomalies in a wide range of nodes in multiple systems that are suspected to cause QoE anomalies. QRank can successfully pinpoint the anomalous system, which can be a server, a user device or a network managed by a specific provider at a specific location. We run QRank in a production VoD deployed in Azure Cloud with 100 users emulated in PlanetLab and 24 users emulated in Azure Cloud. The results show that access, transit networks and user devices contribute mostly for QoE degradations. 61.97% of QoE anomalies identify access or transit networks as anomaly systems and 38.14% identify the user devices as anomaly systems. Cloud networks and servers seldom incur QoE anomalies.

1.4.5 Persistent and recurrent QoE anomalies for video streaming in the Cloud

We run QRank in extensive experiments in production Cloud. We find several interesting insights about QoE anomalies of video streaming in Cloud environments. 91.4% of QoE anomalies are detected on 15.32% of users. These users experience QoE anomalies persistently and recurrently. The Cloud servers and networks seldom cause QoE anomalies. More than 99.98% of QoE anomalies are identified in anomalous systems including the transit networks, the access networks and user devices. We infer that transit networks are the actual bottleneck systems for QoE anomalies in production Cloud. More than 95% of persistent and recurrent QoE anomalies are identified in less than 10 transit networks. We collect latency measurements to anomalous networks and the analysis indicates that the limited capacity in transit networks are the major cause of QoE anomalies. Resulting anomalies impair user QoEs persistently or recurrently. In order to

provide good user QoE, the Cloud provider should identify transit networks that may become bottlenecks for high quality video streaming and appropriate peering with Internet Service Providers (ISPs) to bypass these bottlenecks.

2. QOE BASED MANAGEMENT AND CONTROL

2.1 Problem Statement

Various performance metrics are monitored in the Cloud, the CDN, the networks and the user devices. These metrics barely reflect end user QoE in VoD service. When VoD systems are deployed in the Cloud, the video server would be deployed in one of VMs. Multiple VMs on the same PM are sharing physical resources such as CPU, disk, memory and network interface. Resource sharing may lead to performance interference from other VMs. The impact of interference on end user QoE cannot be reflected in performance measurements on one VM. Even if there are no other VMs in the physical host, the performance of a particular physical machine in the data center could be unpredictable as the tenant has limited or no access to the PM. Various faults in the Cloud degrade end user QoE but some of those cannot be detected by resource measurements based monitoring. A user request might be sent to a server that has been removed. A new server might boot with an outdated cache table and directs the user request to a wrong server. Servers may hang due to sudden high workload from background VMs. A system administrator could misconfigure content folders. Cloud monitoring systems that monitor the VMs via the network probing and server load cannot detect these faults [35]. When a fault occurs, the VoD provider may not be able to detect the fault. VM interference in the Cloud and various types of faults in the VoD system negatively affects end user QoE. However, they are not reflected in server measurements and usually cannot be detected in the Cloud monitoring system.

There are many other factors in the Cloud, the transit and access networks affecting the user QoE. The end-user device also plays a significant role in QoE. The VoD delivery chain consists of various application servers, CDN, ISP networks, local networks and user devices

including browsers. An anomaly in any of these components can degrade user experience. Each system in the VoD delivery chain only has a partial view of the VoD system. Different entities monitor anomalies independently. Thus, they fail to give a full picture of the VoD delivery chain. Detection and localization of anomalies are very challenging without a clear view of end-to-end VoD delivery chain.

Low capacity and traffic congestion in access and transit networks degrade the quality of streaming videos. Different stakeholders, namely the Cloud provider, the CDN provider, the transit ISPs, the access ISPs and the users, manage the systems involved in the video delivery. These providers have their own monitoring systems and target different Service Level Agreements (SLAs). VoD providers do not have enough visibility over other systems. To identify the anomalies, VoD providers need a unified system performance metric that reflects user QoE.

2.2 Related Work

2.2.1 Performance Metrics in the Cloud

Early work [32] on analyzing computer system performance use various types of metrics to monitor a system. There are three possible outcomes for each request made to the system: 1) the service is performed correctly; 2) incorrectly; and 3) the service is refused. According to these outcomes, system performance metrics are classified into three categories called **speed**, **reliability** and **availability**. Within the category associated with the outcome of successful service, the system performance can further be measured by the time taken to perform the service (**responsiveness**), the rate at which the service is performed (**productivity**), and the resource consumed while performing the service (**utilization**).

In the Cloud and the CDN, all above metrics are either defined in Service Level Agreement (SLA) or provided in their monitoring system. Generally, the Cloud providers define "Error Rates" for reliability and "Uptime Percentage" for availability [33] . They also propose a SLA on error rates and uptime percentage to guarantee tenants (namely application/service providers) a desirable Cloud/CDN service performance. However, for application providers, the Cloud/CDN SLA can not guarantee application performance for their customers. Recent works on Cloud monitoring system provide various types of VM performance metrics to assist the application/service mangement [35]. For responsiveness, servers' response time is monitored and is used to infer the application QoS [36][37][38] or to perform fault/anomaly detection [39]. For productivity, the server throughput or the server load (i.e. the number of requests served per second) are monitored. For utilization, the server is monitored by utilization metrics of all types of resources including CPU, memory, I/O, disk and bandwidth. However, neither of above metrics reflects end user QoE.

2.2.2 Existing QoE Models

Quality-of-experience (QoE) is a subjective perception of user's acceptability of an application or a service [31]. There have been many works attempting to model QoE by quantitative quality-of-service (QoS) metrics. Some works conducted subjective experiments for the video streaming under various network impairments. They applied machine learning methods or statistical analysis to model the QoE [42]. Other works developed analytical models for QoE over a measurable QoS metric, such as freezing time [43] or bitrate [44]. They then used subjective experiments to validate their assumptions. As video streaming over HTTP becomes popular and standardized these years, most recent works focused on QoE modeling for HTTP based video streaming. Ricky et al [49][46] studied how network QoS including network bandwidth, latency

and loss rate affect the user QoE in HTTP based video streaming. Work in Bell labs [50] modeled per chunk MOS value to estimate QoE for Dynamic Adaptive video streaming over HTTP (DASH), however, they only considered the impact of picture quality and ignored possible freezing time.

2.3 Chunk based Quality of Experience (QoE) Model

Existing QoE models generally consider QoE for a complete video session. Because we need a QoE model that can compute user QoE in real time, we model user QoE as a function of several time changing QoS metrics. We use DASH streaming through the whole project, so our QoE model is proposed for DASH streaming.

2.3.1 Factors impacting QoE in DASH

Existing works obtain QoE from various QoS metrics in DASH streaming. These metrics include the streaming bitrate [47], the frequency of bitrate switching [48], the freezing time [43], and the join time [42]. Our system requires a real-time QoE model, so we ignore the join time because it does not change once the streaming starts. The frequency of bitrate switching is only determined by the bitrate adaptation logic implemented in the video player [47], so it is not considered as the rate adaptation logic is out of our scope. DASH encodes a video in multiple bitrates and split each bitrate version into a series of fixed length segments, called Chunks. During streaming, the DASH player detects the network throughput in real time and adaptively selects the bitrate for every Chunk. We define QoE per each Chunk.

2.3.2 QoE Model for Chunk

A psychology study shows that user perception in video streaming follows a logarithm law [44] on the bitrate as shown in equation (2.1), where r is the bitrate of video streaming and $r_{\max}$ is the maximum possible bitrate for reference. $a_1 = 1.3554$ and $a_2 = 40$ are empirical parameters learned from subjective experiments according to [44].

$$q_{bit-rate}(r) = a_1 \cdot \ln \frac{a_2 r}{r_{\max}} \tag{2.1}$$

A human vision study finds that user experience follows a logistic model of freezing time [43] as shown in equation (2.2).

$$q_{freezing}(\tau) = \begin{cases} 5 - \dfrac{c_1}{1 + \left(\dfrac{c_2}{\tau}\right)^{c_3}} & \tau > 0 \\ 5 & \tau = 0 \end{cases} \tag{2.2}$$

τ in equation (2.2) is freezing time caused by a single buffering event. $c_1 = 6.3484$, $c_2 = 4.4$ and $c_3 = 0.72134$ are parameters learned by subjective experiments in [43]. Both above models follow Mean Opinion Score (MOS) standard [40] to value QoE on a scale of 1 to 5 that ranges bad to excellent.

2.3.2.1 Linear QoE Model for Chunk

We intuitively combine above models on bitrate in equation (2.1) and freezing time in equation (2.2) as our Chunk QoE model in equation (2.3).

$$q(\tau, r) = \delta \cdot q_{freezing}(\tau) + (1 - \delta) \cdot q_{bit-rate}(r) \tag{2.3}$$

Ideally, in DASH streaming, the bitrate lower as much as possible to avoid freezing. There would be no freezing until the bit-rate drops to the lowest level. δ denotes the ratio of the lowest

QoE without freezing over the possible best QoE. We use the linear combination of existing models as our Chunk QoE model.

2.3.2.2 Cascading QoE Model for Chunk

Linear QoE model combine two factor additively, but the impact of these two factors on QoE are multiplicative. We therefore model the QoE as a multiplication of the freezing time model and the bitrate model in.

$$q(\tau, r) = \frac{1}{5} q_{freezing}(\tau) \cdot q_{bit-rate}(r) \tag{2.4}$$

The coefficient $1/5$ is to normalize the QoE value within $[0,5]$. We later denote this QoE model as the cascading QoE model as it cascades the impact of the freezing time on the bitrate QoE model. In this dissertation, we only test above QoE models that are combinations of existing models. However, other real-time QoE models can be adapted to our system when necessary.

2.4 System QoE Score

2.4.1 QoE based Monitoring

The key idea of our QoE based monitoring system is to learn a server's SQS in real time from QoE of clients in the same network.

2.4.2 QoE as a Performance Metric ---- System QoE Score (QS)

We propose a QoE Score (SQS) as a QoE based performance metric to represent the value of QoE one system can provide for users using the system. We assume clients connecting using the same system are in a client group. They can be using the same server, the same router, the same

transit network, the same access network, or the same type of devices. The QS of one system is learnt from Chunk QoEs reported from the system's client group. The QS should always update to reflect the latest QoE the system provides. Besides, as the system performance is non-stationary due to the changing workload, the varying network condition, and the dynamic background interference. Therefore, QS should be able to track non-stationary changes.

2.4.3 Learning of QoE Score (QS)

We collect QoE from users in each system. The system can be a server, a router, an access network, a transit network, a cloud network or a type of user devices. We assume any systems with lower QoE Scores have performance issues. The QoE itself is like the reward of using the system. The idea of learning system QoE score is from the reinforcement learning. In the multi-arm bandit problem in the reinforcement learning [54], the rewards of choosing a bandit can be used to evaluate and predict the expected reward the bandit can give in the future.

Collected QoE on a particular system in VoD system presents as a series of rewards of using the system. The action in our system is the server selection and the total reward gain is the total QoE for all clients in the client group. The naive technique to learn QS is the averaging method that computes the value of an action by the average of all past rewards of the action. In VoD, it can be computed by all chunk QoE from users in a system, as shown in (2.5),

$$Q^t(s) = \frac{\sum_{t_i \in [t-\tau, t]} q_{t_i}(s)}{\sum_{t_i \in [t-\tau, t]} 1} \tag{2.5}$$

where $q_{t_i}(s)$ denotes the QoE of a user in system s at time t_i. (2.5) counts the average of all QoEs for system s during time $[t-\tau, t]$. τ denotes the size of the time window to count the QoE score. The averaging method is appropriate for a stationary system.

To track the non-stationary system performance, we use a learning method to weight recent rewards more heavily than long-past ones. Specifically, we use an incrementl rule with a constant step-size parameter to update Server QoE Score (SQS) as shown in equation(2.6).

$$Q^t(s) = Q^{t'}(s) + \alpha \cdot \left(q^t(s) - Q^{t'}(s)\right) \tag{2.6}$$

$Q^t(s)$ is the system s 's QS at time t. $q^t(s)$ is a QoE value received from a user in the system s at time t. α is the weight of the latest QoE reward of using s. $Q^{t'}(s)$ denotes the QS of the system before t.

2.5 QoE Anomaly

End-user QoE reflects the performance of complete end-to-end systems. Users' perception of QoE reveals anomalies. Let q_0 be the minimum value of QoE that users would accept. Any QoE below q_0 would influence users' decision to continue the VoD service. VoD providers need to maintain at least q_0 to retain users . VoD providers often conduct subjective studies to obtain q_0 for QoE [77].

We define QoE anomaly to be any fault or congestion that degrades end user QoE such that users' QoE values to below q_0 . Any possible faults and temporary congestion that do not degrade user QoE below q_0 are not considered to be a QoE anomaly.

2.6 Evaluation

We only evaluate the QoE models in this chapter. The QoE scores and the QoE anomalies are used in systems described in later chapters.

2.6.1 Evaluation of QoE models

We compare QoE models in two streaming sessions. The server's outbound bandwidth is throttled occasionally to varying degrees to incur QoE degradations. The client agent monitors the bitrate and the freezing time for each chunk and computes the linear QoE and the cascading QoE according to Equation (2.3) and (2.4) in Figure 3. In the linear QoE model, we weight the impact of the freezing time and the bitrate equally, i.e. $\delta = 0.5$.

In the streaming session shown in Figure 3 (a), the video player switches the chunk bitrate to the lowest level at around 7 minutes. The linear QoE evaluates the real-time QoE as "Fair" because the impact of bitrate is only weighted as half of the QoE and there is no freez-ing at that time. Comparably, the cascading QoE model considers both the impact of the bitrate dropping and the freezing, and evaluates the QoE as "Bad" because the bitrate drops. Similarly, in the streaming session shown in Figure 3 (b), the server bandwidth is throttled to emulate various length of freezing events. At those freezing events that last longer than 10 seconds, the cascading model evaluates the QoE as 0, while the linear model evaluates the QoE as "Poor" to "Fair". The linear model cannot fully reflect the QoE degradation if the impact of one factor dominates. In contrast, the cascading QoE model can reflect the QoE degradation when either of the factors dominates. We expect that the adaptive control using the cascading QoE model is sensitive to both the bitrate dropping and the freezing events.

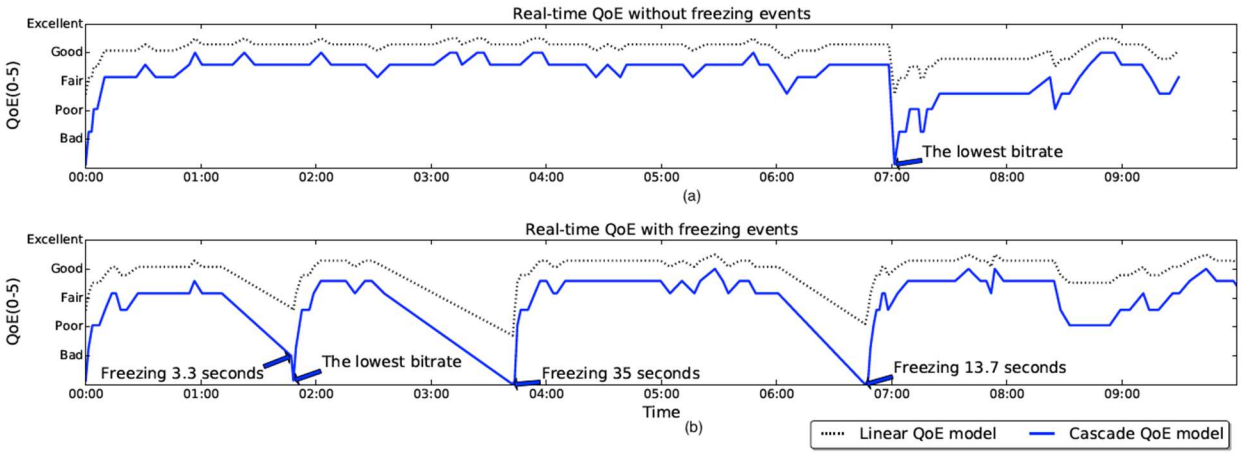


Figure 3: The linear QoE model vs. the cascading QoE model in streaming sessions with and without freezing

3. QMAN: A QOE BASED MANGEMENT SYSTEM FOR VOD IN THE CLOUD

3.1 Introduction

The Cloud infrastructure has become an ideal platform for large-scale applications with periods of flash crowds, such as Video-on-Demand (VoD). The Cloud can provide elastic amount of resources to meet the dynamic user demand [8]. As VoD systems migrate to the Cloud, new challenges emerge for VoD providers to manage user Quality-of-Experience (QoE) [79]. Extra complexities due to virtualization, resource sharing and operational failures add to the challenge.

The Cloud itself is complex. The Infrastructure as a Service (IaaS) in the Cloud is usually offered by a virtualized datacenter consisting of a cluster of physical ma-chines (PMs). Each PM hosts multiple virtual machines (VMs) possibly belonging to different customers. The video server would be deployed in one of these VMs. Multiple VMs on the same PM are sharing physical re-sources such as CPU, disk, memory and network inter-face. Resource sharing may lead to performance interferences. The applications running on other customers' VMs are not visible to the VoD provider. A VoD provider can neither predict its video server performance nor control the resource sharing among VMs. Even if there are no other VMs sharing the same physical host, the performance of a particular physical host in the data center can still be unpredictable, as the VoD provider has limited access to the data center. Existing works study the impact of interference by benchmarking the Cloud using CPU/disk/memory/network intensive tasks. However, how the Cloud interference affects the QoE of video streaming is unclear and can hardly be quantified [30]. Thus, we believe that the best way to understand such impacts is to observe the end-user QoE directly. Instead of modeling complex systems in the Cloud, we believe that the dynamics in QoE would re-flect the system performances in real time.

Provisioning Cloud resources on the fly may cause failures that are difficult to identify. A user request might be sent to a server that has been removed. A new server might boot with an outdated cache table and direct the user request to a wrong server. Operational failures happen often. Servers may hang due to surge in user demand. A system administrator could misconfig-ure content folders. Simple monitoring schemes like the network probing and the server load monitoring cannot identify these failures [31]. When a failure happens, the VoD provider may not know the cause of the failure but users can definitely experience early symptoms. For example, before a streaming session crashes, the video player on the user side may undergo several video Chunk request timeouts, observe buffer depleting, or experience the video freezing. One can take advantage of these early symptoms to prevent streaming crashes by adaptively selecting an alternative server for the user.

We propose QMan, a QoE based management system that 1) monitors individual user QoE; 2) infers server performance from users' QoE; 3) adaptively selects servers based on server QoE; and 4) effectively responds to various failures according to QoE. To scale beyond millions of users, QMan adopts an agent-based design and controls the VoD system adaptively. To monitor end users' QoE, agents are deployed in video players on the client side, referred to as client agents. To monitor the operational changes in the VoD system, such as content placement and the resource deployment, agents are de-ployed in video servers, referred to as cache agents. Cache agents are organized in a location aware overlay network. They communicate with neighbors in the overlay to discover videos cached in neighboring servers. For each video discovered, one cache agent can obtain the ad-dresses of neighboring servers that cache the video, also referred to as candidate servers. QMan implements the control of server selection for all users in two paradigms. A decentralized paradigm implements the control on cache agents. A fully distributed paradigm

implements the control on client agents. For evaluation purpose, we test our system in an experimental VoD system deployed in Google Cloud with hundreds of users emulated in PlanetLab. We also evaluate the robustness of our system under various levels of injected interference and failures. Results show that our QMan outperforms existing measurement based server selection systems by improving the 90th percentile QoE up to 30% in production Cloud environment. By monitoring server performance using QoE, QMan discovers server failures timely and prevents session crashes accordingly. We also conduct experiments to emulate highly dynamic Cloud environment. Results show that the reinforcement learning used in QMan finds a good tradeoff between the exploration and the exploitation in server selection, which is necessary when servers' performance change dynamically. Lastly, we compare two paradigms of QMan implementation and show that the decentralized paradigm achieved better overall QoE with less server switches than the fully distributed paradigm. Overhead analysis shows that both implementations are scalable.

3.2 Related Work

3.2.1 QoE Based Controls

With the recent advances in QoE modeling, existing work directly use QoE as a feedback in the control and management of video streaming system. [48] studies the QoE of the quality transitions and propose a QoE-aware rate-adaptation system for DASH streaming. In [47], the logarithm law is used to model the user QoE over bitrate. It proposes an optimal caching algorithm to maximize users' QoE in wireless network. A varying QoE served from different servers in CDN is studied in [82]. It designs a client-side QoE based server selection algorithm for each client.

This method assumes that a special “iBox” device can be deployed in the client’s residential network and the device is pre-configured with addresses of servers in CDN. These assumptions limit its applicability in production systems as the installations of “iBox” in large-scale would be difficult.

3.2.2 Server Selection in VoD

The main task of a control system in VoD is to select servers for user requests. Server selection schemes are extensively studied for improving user QoE.

3.2.2.1 Studies of client-side server selection schemes

Client-side server selection schemes are proposed to control the server selection for individual users. Before the DNS based server selection [52] and HTTP redirection [63] were widely used, early researches focused on client-side server selection for web and video streaming services [62]. The benefit of client-side server selection is that the client can dynamically probe servers and can select a server considering both the server load and the network proximity to the client himself. In addition, the client-side server selection can achieve a flow-based control in video streaming service. It has an advantage when the group of servers is heterogeneous or widely dispersed across the network and when users are different in their network conditions and their proximities to servers vary. Flow based control can optimize the QoS for a single user and can adaptively tolerate faults that only affect an individual user. However, following researches found that the client-side server selection requires special software to be deployed in the client side. It also involved dynamic probing of multiple servers, which is very costly as the number of servers scaled up. Therefore, the DNS based server selection and HTTP redirection are combined to

perform server selection in modern content delivery network [52]. In present VoD system, video player such as DASH are running in client side. Powerful client-side scripting languages supported in browsers have made the client-side based control possible. In addition, clients can rely on their own QoE on servers to select servers thus avoiding the costly probing of servers.

3.2.2.2 Popular server selection systems

For VoD systems using CDN, DNS based server selection is used as a proximity aware server selection scheme [52]. To balance load among servers, various redirection schemes are combined with DNS based server selection at a finer level [53]. Such schemes are supposed to be effective in improving user QoE because they consider both the network proximity and the server load in server selection. Measurement studies in YouTube also reveal that network latency and server load are major factors in their server selection scheme [25]. However, recent studies reveal that these server selection schemes do not work as well as expected [10]. There remains many other factors affecting user QoE. These factors are neither considered completely nor be modeled accurately.

3.3 QMan System

The main idea of QMan is to use end users' QoE to control and manage the VoD system in the Cloud environment. The management tasks include monitoring user QoE in real time and tracking operational changes including content caching and resource provisioning. The control is to select servers for each user adaptively. As it is impossible to use a single manager to perform all these tasks in a large-scale VoD system, QMan distribute these tasks to agents on video players in the clients and cache servers in the Cloud, as shown in Figure 4. Client agents are responsible

for monitoring real-time QoE and communicating with their superior agents deployed on the "closest" server, also known as cache agents. In the following of this dissertation, "closest" denotes the "closest" network distance in terms of latency. Cache agents are responsible for management tasks of monitor-ing operational changes. These tasks include 1) discovering multiple neighboring servers that cache a requested video, hereafter refered to as Multi-candidate Content Discovery (MCCD); 2) maintaining a Candidate Server Table (CST) that records K neighboring servers for popular videos discovered from MCCD; 3) updating the CST of cache agents via sending the dynamic changes of the content caching/resource provisioning to neighboring cache agents. In order to discover neighboring cache agents, we deploy a centralized tracker in the Cloud to organize cache agents in a location aware overlay network. Thus, cache agents communicate with neighbors to discover changes in any part of the VoD system.

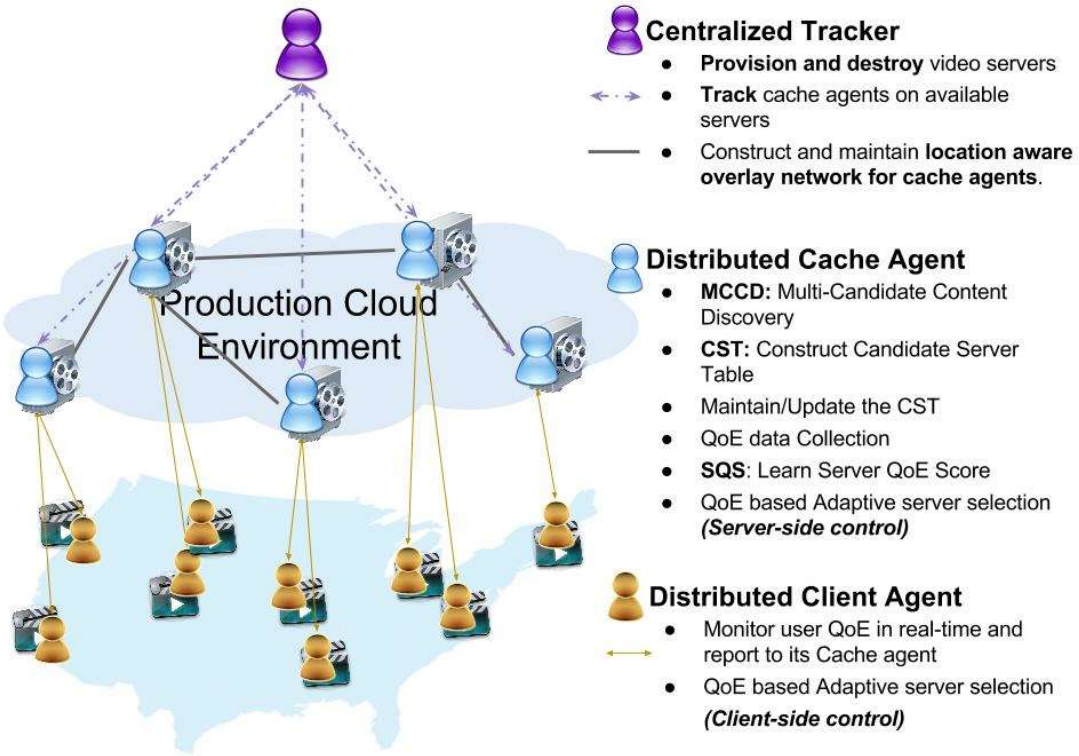


Figure 4: The agent-based design of QMan

QMan implements the control of server selection in two paradigms, a decentralized paradigm and a fully distributed paradigm. The decentralized paradigm controls on cache agents. Each cache agent controls the server selection for a group of users close by. The fully distributed paradigm controls on client agents. Each client agent only controls the server selection for itself. Control tasks include 1) evaluating how servers perform in terms of QoE, later refered to as Server QoE Score (SQS); 2) controlling the server selection adaptively according to SQS, namely the QoE based Adaptive Server Selection.

3.4 Management tasks in QMan

In this section, we explain the management tasks of QMan. These tasks include how the centralized tracker organized the location aware overlay network of cache agents; how cache agents communicate with each other to discover candidate servers; and how cache agents maintain their CSTs. Then, we explain control part of QMan in two paradigms.

3.4.1 Location Aware Overlay Network

The centralized tracker connects cache agents on all video servers in an overlay network. Operational changes on one video server spread to cache agents on all other servers in the overlay. Intuitively, in order to discover videos cached in one video server, the cache agent on the server floods its list of cached videos to neighboring cache agents in the overlay. To run the flooding like algorithm efficiently, the overlay network should connect cache agents in a way that each agent only forwards messages to closest agents with lowest network latencies. We construct such location-aware overlay network by the centralized tracker according to the algorithm shown in. It builds the overlay network in a tree graph, denoted as G. $A = \{A_1, A_2, \cdots, A_M\}$ are M cache

agents to be connected. The algorithm initializes graph G with nodes that are two agents with the minimum pairwise network latency. The average Round Trip Time (RTT) from ICMP pings measures the latency between two nodes. Next, the centralized tracker searches a new node not in G that has the minimum RTT to the closest node in G. The algorithm runs iteratively from line 4 to line 6 until all nodes are connected to G. The overlay network updates as nodes are provisioned and deleted. When deleting an existing node A_d, all child nodes of A_d need to be reconnected to nodes that are not in A_d 's branch. Though the overlay construction introduces ICMP Ping traffic between all pairs of agents, we believe it is acceptable as it is a one-time cost at the overlay constructing stage.

Data: All cache agents, $A = \{A_1, A_2, \cdots, A_M\}$

Pairwise Round Trip Time (RTTs) between agents, $T = \{t_{ij}, 1 \le i \le M\}$

1: Initialize overlay graph:

$G = \{V, E\}$

$V = \{A_x, A_y\}, \mathrm{E} = \{(A_x, A_y)\}$ where $t_{xy} = \min(T)$

2: Denote $B = A - \{A_x, A_y\}$

3: **while** $B \neq \varnothing$ **do**

4: **Denote** $T_{BV} = \{t_{bv} \mid A_b \in B, A_v \in V\}$

5: **Find** $t_{xy} = \min(T_{BV})$

6: **Update** G **with**

$V = V \cup \{A_y\}$

$E = E \cup \{(A_x, A_y)\}$

$B = B - \{A_y\}$

Figure 5: Construction of Location Aware Overlay

3.4.2 Multi-Candidate Content Discovery (MCCD)

Unlike unstructured P2P network, we run the content discovery once at bootstrapping stage and maintain the Candidate Server Table (CST) for popular videos in cache agents. We propose a distributed Multi-Candidate Content Discovery algorithm (MCCD), to build CST on each agent as described in the MCCD algorithm showin in Figure 6.

For each cache agent A_i, $V_i = \{v_t\} \subset V$ is the list of videos cached locally in A_i, where $V = \{v_t \mid t = 1, \cdots, T\}$ denotes all available videos in VoD system. In MCCD, each agent builds its own CST at the bootstrapping stage by flooding its locally cached video list to neighbors. The agents receiving the list will add the server into their own CSTs and iteratively forward the newly added items to their neighbors until each agent's CST is completed. Different from popular flooding algorithms where updates are flooded to all nodes in the overlay, the cache agent in MCCD stops forwarding messages once K candidate servers are discovered. We prove in theorem 1 that the amount of MCCD traffic can be bounded by K and does not increase as the overlay network grows.

Data: Local cached contents on agent A_i , $V_i = \{v_t\} \subset V$

1: Initialize **CST** for A_i as $CST_{A_i} = \{L(j) \mid 1 \le j \le T\}$

2: **for** $v_j \in V_i$ **do**

3: $L(j) = \{\langle A_i, 0 \rangle\}$

4: Initialize **content update** message for A_i as $U_i \longleftarrow\!\!\!- \varnothing$

5: $n_{forward} = 1$

6: **for** $v_j \in V_i$ **do**

7: Add $\langle v_j, A_i, n_{forward} \rangle$ into U_i

8: Send U_i to all neighbors of A_i

9: **while** *Receive* U_n *from node* A_n **do**

10: $U_i \longleftarrow\!\!\!- \varnothing$

11: **for** $\langle v_j, A, n \rangle \in U$ **do**

12: Sort items in $L(j)$ by their values ascendingly

13: **if** $\|L(j)\|_0 \le K$ **then**

14: $L(j) = L(j) \cup \{\langle A, n \rangle\}$

15: $U_i = U_i \cup \{\langle v_j, A, n+1 \rangle\}$

16: **else**

17: $k = \arg\max_{0 \le i < \|L(j)\|_0} (L(j)[i][1])$

18: **if** $L(j)[k][1] > n$ **then**

19: Delete $L(j)[k]$

20: $L(j) = L(j) \cup \{\langle A, n \rangle\}$

21: $U_i = U_i \cup \{\langle v_j, A, n+1 \rangle\}$

22: **if** $U_i \neq \varnothing$ **then**

23: Send U_i to all A_i's neighbors except A_n

Figure 6: Multi-Candidate Content Discovery (MCCD)

Theorem 1. The total outbound traffic for an agent A to build CST_A is proportional to $T \cdot K$, where T is the total number of videos and K is the number of candidate servers.

Theorem 1 shows that the outbound traffic sent by one cache agent in MCCD is independent of the size of the overlay network. It increases linearly with the number of videos and the number of candidate servers. Therefore, it is bounded by K as the number of popular videos is finite. Once the CST on an agent is completed, the cache agent can look up candidate servers for a video request and reponds directly.

3.4.3 Candidate Server Table (CST) Maintainance

Cache agents need a mechanism to update CST in case there are changes in the content placement and the overlay network. We develop following mechanisms to update CST.

- Video Deletion: When agent A deletes a video, the agent A should notify all others who denote A as the candidate server in their CSTs. A floods a message "DELETE v_d on A " to all neighbors. Agent receiving the message checks its CSTs and deletes A accordingly if is a candidate server for v_d in its CST. The agent then forwards the message to its neighbors iteratively.
- Video Addition: Agent A caching a new video v_n does not need to notify all other agents. Only agents whose CST has candidate servers further than A for v_n should be notified, so A sends a message "Add A as candidate for v_n " to neighbors iteratively and the message stops being forwarded at nodes whose CST hold K candidate servers of v_n that are closer than A .
- Cache Agent Leaving: Before performing overlay changes as Algorithm in Figure 5 describes, the leaving agent sends out video deletion messages for all its cached videos.

- Cache Agent Joining: When an agent joins to the cache agent overlay, the agent adds all its cached items to an update message and floods the update message in the updated overlay network.
- Periodic Maintenances: An agent's CST can be corrupted or miss items. In order to maintain the agent's CST, each cache agent periodically runs MCCD to fix corrupted CSTs. The period can be set to a relatively long period (i.e. several hours or one day).

3.5 QoE based adaptive server selection

In QMan, each cache agent maintains a list of servers that cache a particular video. Client agents obtain the list of candidate servers from cache agents. QMan implements the QoE based adaptive server selection in two paradigms. One controls on cache agents and the other controls on client agents. We then explain step-by-step how QMan operates in those two paradigms.

3.5.1 Decentralized paradigm

We then explain how QMan operates in the decentralized paradigm. As shown in the right part of Figure 7, each cache agent performs the QoE based adaptive server selection for a group of users near by. We have a VoD client B requesting a video v_i. B's client agent sends a query message with the requested video name v_i to its cloest cache agent S_M (Step ①). Upon receiving the request query from B, B looks up its CST and finds candidate servers that cache the requested video v_i. Each cache agent maintains a table of server QoE Scores for all servers. The server QS represents how a server performs and is learnt by the QoE of end users streaming from the server. The computation of server SQS is detailed in Section 2.4.2. B then looks up the SQS for candidate servers, selects S_k according to the QoE based adaptive server selection algorithm denoted as the

blue box (detailed in section 3.5.3), and responds to B 's query (Step ②). Client B downloads the initial segment of the video from S_k (Step ③). After receiving the initial segment, the video player on B starts playing the video. The client agent gets the the freezing time and the bit-rate of the segment, to compute its QoE according to the chunk QoE models (Step ④) (described in section 2.3). The client agent B then reports the QoE to its cache agent S_M . When the cache agent S_M receives B 's QoE, it updates the SQS for S_k according to the SQS learning algorithm denoted in the purple box (Step ⑤), described in section 2.4.2. Cache agent S_M then looks up its latest SQS table and runs the QoE based adaptive server selection algorithm to pick up S_x as the new server (Step ⑥). Client B then switches to server S_x and downloads the next segment (Step ⑦). The client agent and its cache agent then runs iteratively from ④ to ⑦ for all segments of the video in the streaming session. The segment is sufficiently sized to avoid unstable behavior.

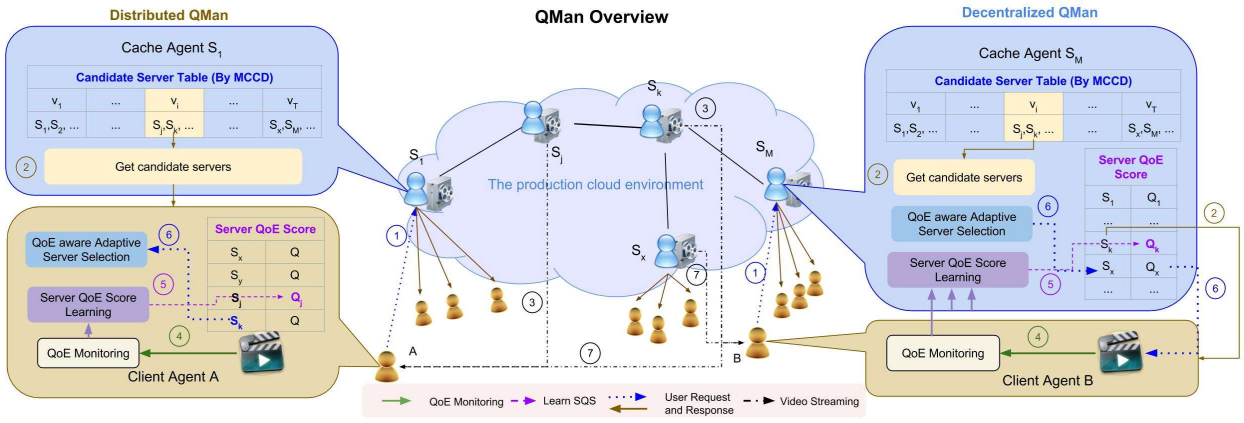


Figure 7: QMan Overview

3.5.2 Distributed paradigm

In the fully distributed paradigm, QMan controls on client agents. The client agent only needs to communicate with its cache agent once to get the list of candidate servers for each video request. Then it performs the QoE monitoring, the SQS learning, and the adaptive server selection all on itself. The fully distributed implementation does not involve any communications between the client agent and the cache agent in the process of server selection. Compared to the decentralized paradigm, the distributed paradigm gives more flexibility in the switching servers. The client agent can switch servers at any time without any limitation from periodical communications. However, the SQS of candidate servers can only be learnt from the client's own QoE, which may not be enough to explore all candidate server's performance in a timely manner.

The left part of Figure 7 shows the detailed steps of how QMan operates in the distributed paradigm. We assume that there is a client A requesting video v_i to its cache agent on S_i (Step ①). The cache agent S_i responds the request with a list of candidate servers (Step ②). The client agent A then initializes the SQS table for all candidate servers with an default QoE Q. Then the client first randomly chooses a server S_j to download the first segment of the video and computes the QoE (Step ③ to ④). Similarly as the cache agent in the decentralized paradigm, the client agent updates the SQS (Step ⑤) of server S_j with its own QoE and runs the adaptive server selection algorithm (Step ⑥) to select a new server S_k according to the updated SQS table. Then the client starts downloading the next segment (Step ⑦). The client agent runs step ④ to ⑦ iteratively until all segments of the video are downloaded. The smallest segment size in the distributed QMan is a video chunk.

3.5.3 QoE based adaptive server selection

The SQS of a server evaluates the QoE that a server can provide when it is selected. In order to maximize the QoE of next chunk, the server with the best SQS could be selected as denoted in Equation (2.7).

Greedy action:

$$s_t^* = \arg\max_{s \in A} Q^t(s) \tag{2.7}$$

This is the greedy action in reinforcement learning. This method always exploits current knowledge to maximize immediate reward. It spends no time at exploring servers with poor QoE in the past, so it cannot be aware of the performance recovery of servers.

An alternative is to behave greedily most of time but explore other servers once in a while with small probability ε, which is known as ε-greedy action described in (2.8). s' is randomly chosen from all candidate servers. p is a random number in $[0,1]$.

ε-greedy action:

$$s_t = \begin{cases} \arg\max_{s \in A} Q^t(s) & p \geq \varepsilon \\ s' & p < \varepsilon \end{cases} \tag{2.8}$$

3.5.4 QoE based failover control

Failures can happen in various components of the VoD system. Some failures are hard to detect and identify. Administrators can accidentally delete videos. Video server can hang due to software errors. A virtual machine in the Cloud can terminate due to physical machine failures. From the user's perspective, all these failures end up with a chunk request timeout or a HTTP request error. We designate these errors perceived by the user as unacceptable QoE ($q(s) = 0$).

When the cache agent receives $q(s)=0$, it sets $Q(s)\to 0$. The agent then selects another server. The client then resends the chunk request to the newly selected server to prevent the streaming session crash. When the cache agent itself fails, then we rely on DNS to connect to a new cache agent.

3.6 System Evaluation

3.6.1 Experimental Setup

We deploy cache agents and client agents in an experimental VoD system, which runs 12 video servers in 10 datacenters in Google Cloud [81] and emulates near 300 users in PlanetLab. PlanetLab [80] is a global research Cloud with hundreds of servers around the world. We deploy client agents in 284 PlanetLab nodes to emulate VoD clients. "f1-micro" instances are provisioned in Google Cloud to serve as video servers. The servers are provisioned according to the number of users shown in Table 1. The locations of clients and video servers are shown in Figure 8. To emulate a large number of videos, we rename the same video clip (ten-minute video) as 1000 distinct videos. The videos are encoded in 9 levels of bitrates for DASH streaming, varying from 300 kbps to 10 Mbps. We assume the popularity of these videos follow Zipf distribution [58]. A user demand based caching method is used to cache videos, so videos would be cached in more servers as more users request them. The least popular video is guaranteed to be cached in at least 3 servers. The following experiments all run in this VoD platform, yet the detailed setup may vary according to the purpose of evaluation.

Table 1: Resource Provisioning & Content Caching in Experimental VoD System

Regions	asia-east1	europe-west1	us-central1

Zones	a	b	c	b	c	d	a	b	c	d
# of clients	42			123			132			
# of servers	S_1		S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}
# of servers	1	0	1	2		2		1	1	2		1	1
# of videos	307		346	326		332		331	344	324	313	333	337

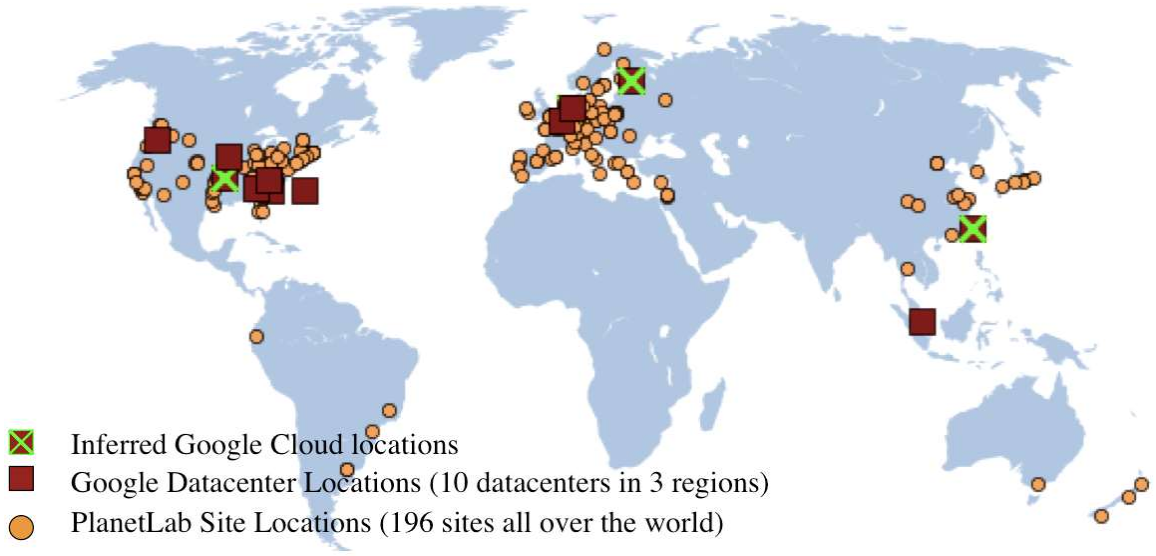


Figure 8: The Locations of Clients and Servers in Experimental VoD System

3.6.2 Study of QoE models in QMan

In order to show how QoE models impact our server selection system, we then deploy two clients in the same zone of Google Cloud. Both clients run adaptive server selection in the same fashion but use different QoE models. They request the same video from server A and adaptively select servers among the same candidate servers: A , B , C , which are deployed in the same zone of Google Cloud. To emulate QoE issues, we throttle the outbound bandwidth of A to 2Mbps in the middle of the streaming session and study how clients adapt to the issue. Figure 9 plots the candidate servers' SQS learned by two clients running the linear and the cascading QoE models,

repectively. We find that both clients change servers when the SQS of *A* drops below other candidates' SQS. This example shows that our QoE based server selection is operated via comparing the SQS among candidate servers. Only the relative SQS value matters and QoE models have little impact on the server selection logic. Our agents always optimize the predefined QoE value. Readers can apply their own QoE models in our agents to optimize corresponding factors at their own discretion.

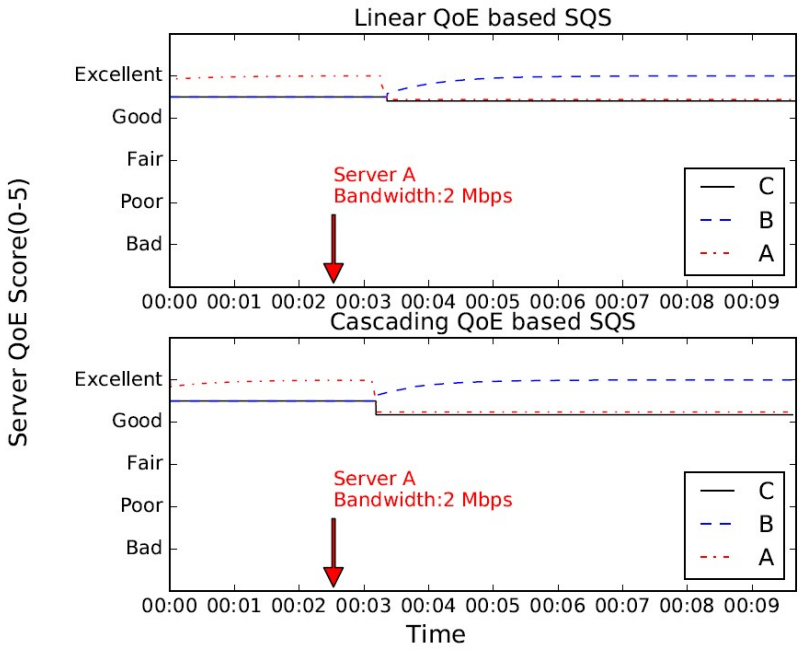
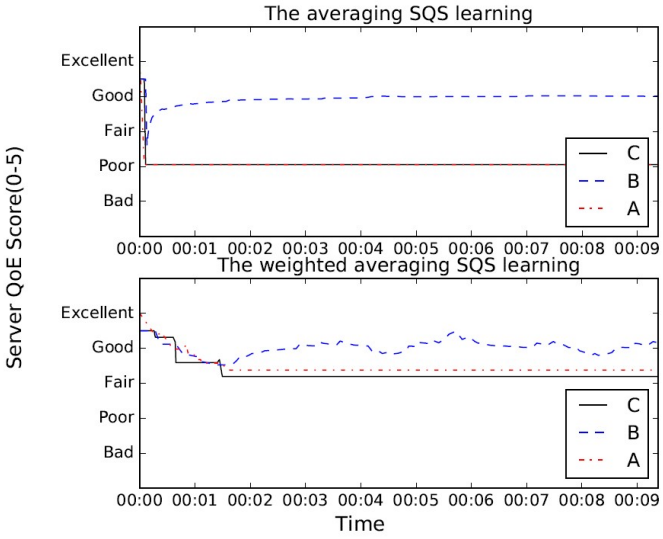


Figure 9: The linear QoE model vs. the cascading QoE model in QMan

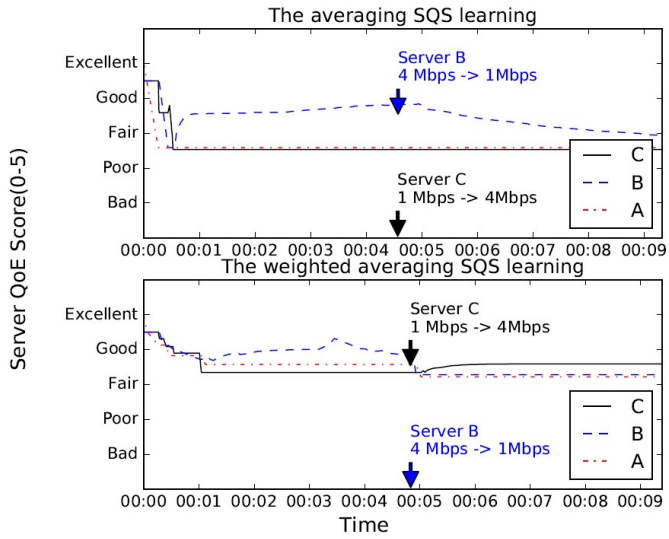
3.6.3 SQS learning in QMan

In QMan, we use commonly used methods to learn SQS for candidate servers. One is the averaging method that use the average of all QoE on a server as its SQS. The other is the weighted averaging method that always assign higher weights to more recent QoEs. In order to show how the above two methods learn the SQS over time, we test clients running both SQS learning methods

on three candidate servers. We compare the SQS learning methods in two cases. First, we emulate a stationary scenario where the performance of candidate servers remains constant over time. Second, we emulate a non-stationary scenario where the server performance changes during the streaming. Candidate servers' outbound bandwidth are throttled differently as if these servers' performance vary. In the non-stationary scenario, we randomly change two servers' outbound bandwidth in the middle of testing. In both cases, the clients use the cascading QoE model and select servers using greedy action. We run all testing clients and their candidate servers in the same datacenter in Google Cloud to exclude the impact of Internet condition.



(a) Stationary Scenario. A: 2Mbps; B: 4Mbps; C: 1Mbps



(b) Non-stationary scenario. Intial: A: 2Mbps; B: 4Mbps; C: 1Mbps.
After change: A: 2Mbps; B: 1Mbps; C: 4Mbps

Figure 10: The averaging vs the weighted averaging in SQS learning in QMan

Figure 10 (a) compares how each agent learns SQS of candidate servers. The averaging SQS learning agent only explores candiate servers several times and then stays with server B for the complete streaming session. In the averaging method, the first observed QoE is important. Because if the first QoE value is bad, the agent will not choose the server any more. In the weighted averaging method, the SQS of all three servers are initialized as the best QoE level and the agent then update SQS with recent QoEs. Because the weighted averaging method change SQS gradually, the agent needs to explore servers with more times to make SQS converge. In the non-stationary case shown in Figure 10 (b), we can see that if the server performance change in the middle of the streaming session, the weighted averaging method can weigh recent QoE higher to track the changes. However, the averaging method weigh each QoE observation equally, so the recent QoEs do not change the SQS much. With greedy action, the averaging agent cannot even

select other servers to obtain better QoEs so the non-stationary changes in servers not being selected can never be discovered.

3.6.4 Greedy action vs ε-greedy action in adaptive server selection

As we noticed in the above experiments, the greedy action client always stays with the server with good SQS. Because the agent does not select the server with bad SQS again, the agent has no chance to learn if the server reverts to good performance later. The ε-greedy action can give small randomness in the action of server selection. In Figure 11, we compare the greedy action and ε-greedy action clients in a non-stationary scenario. Both the clients run the weighted averaging method for SQS learning. In the middle of the streaming session, we increase server C's outbound bandwidth as if its per-formance recovers. At that time, both clients already learn that C has the worst SQS. The greedy client does not select C anymore even after C's performance recovers. However, the ε-greedy client later randomly tries C and finds that C provides the best QoE, so the ε-greedy clients is able to switch to C after its performance recovers.

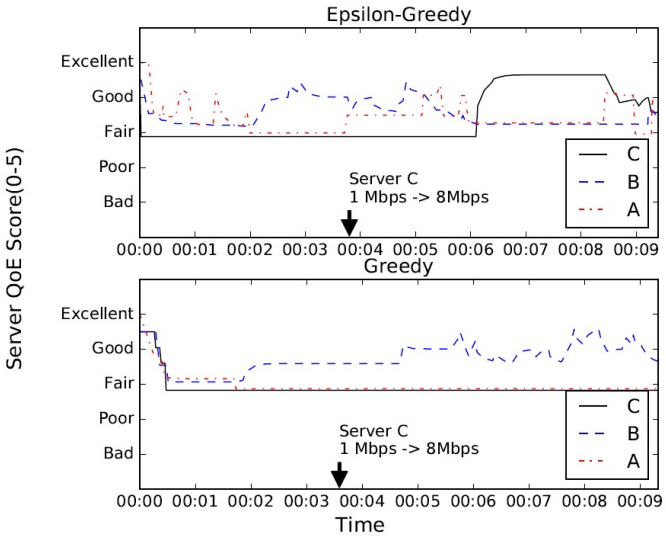


Figure 11: The greedy vs the ε-greedy actions in adaptive server selection

From the above comparison, we note that the weighted averaging method is effective in tracking non-stationary server performance and the ε-greedy action can provide some randomness to explore the perfor-mance changes in servers. Both are necessary for servers in the Cloud as their performance change dynamically.

3.6.5 QMan vs. Other server selection systems

3.6.5.1 Comparison Systems

We implement the following server selection schemes for comparison purpose.

- ***HOP***: Each request is redirected to the server with the minimum hop number among servers with the requested video.
- ***LOAD***: Each request is redirected to the server with the minimum load among servers with the requested video.
- ***RTT***: Each request is redirected to the server with the minimum RTT among servers with the requested video.
- ***RAND***: Each request is redirected to a randomly selected server from servers with the requested video.
- ***QoE***: Decentralized QMan system. We use the cascading QoE model. We apply the weighted averaging method in SQS learning with $\alpha = 0.1$. The greedy action is used for QoE based adaptive server selection. The cache agent selects servers for each user every 30 seconds.

In all existing systems, the client selects server only once at the beginning of a video streaming session. Theses systems represent server selection algorithms widely used in CDN [52]. In ***RTT*** and ***LOAD*** methods, each server periodically probes all other servers every 5 minutes. Smaller

probing period would incur large amount of probing traffic as the number of server increases. After redirection, the client remains with the selected server for the whole video session.

3.6.5.2 Experimental setting

We set up three scenarios to show the system performance under various levels of interference. First, we evaluate our QMan in a real production Cloud environment, namely in the Google Cloud without any additional stress on resources. Second, we evaluate QMan under severe interference. Because we do not have controls over the physical machines in Google Cloud, we emulated the severe interference as background traffic spike by throttling the outbound bandwidth to 4Mbps on two randomly selected servers. Third, we evaluate our system under dynamic interference by periodically throttling the outbound bandwidth to 4Mbps every other minute on two randomly selected servers. We let 284 PlanetLab nodes streaming videos at the same time as if they are watching online videos during a testing period of 10 minutes.

3.6.5.3 Experiment in production Cloud environment (Google Cloud)

Figure 12 shows the cumulative distribution of all users' session QoE in a real production cloud environment. The session QoE is the average QoE of all Chunks in a single video streaming session. The results show that our QoE method gets the best session QoE for most users. We have over 76% users with above QoE value 3 (3 in MOS corresponds to the user satisfaction level "fair"). The RTT has 73% and the HOP has only 49%. We have over 47% of users with above QoE level "good" (QoE value 4). The RTT has 38% and the HOP has only 17%. Our system has 9% and 30% more users with session QoE above "good" level than the RTT and the HOP respectively. It shows that, given the same amount of resources, "good" level QoE can be obtained for more users in our system. Our system has the 90th percentile QoE as 2.5708. The 90th

percentile QoE are 2.2393, 2.4187, 1.4445, 2.0423 for LOAD, RTT, HOP, and RANDOM respectively. Our system performs 6.29% better than the RTT. The poor performance of LOAD shows that the QoE degradation is not caused by the server overload but other factors .The HOP method is designed to select the closest server for users in terms of the network distance. However, it has the worst performance in Figure 12. We tested the same experiment in Google Cloud multiple times at different hours. The performance of the HOP varies a lot. We suspect that it is due to dynamic interference in Google Cloud or varying network conditions. The results show that our system always selects servers that serve better QoE without having to identify causes of performance degradation.

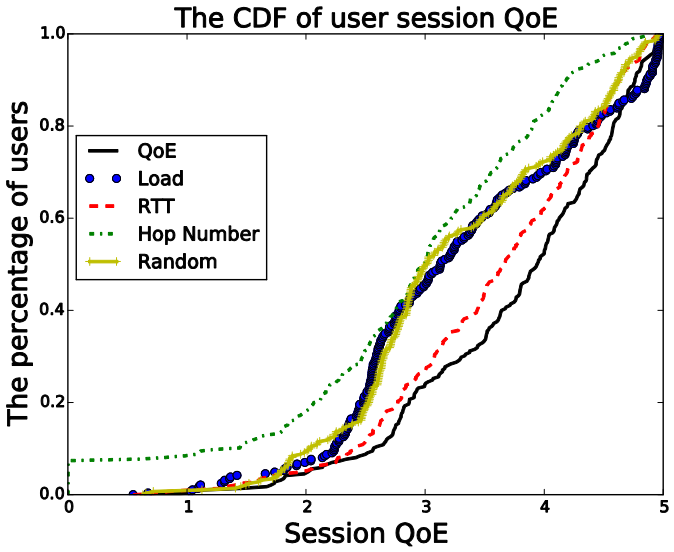


Figure 12: The CDF of session QoE for experiment in Google Cloud.

3.6.5.4 Experiment under severe interference

Figure 13 shows that the QoE method has significant advantages over other methods for those clients affected by the severe interference. We observe that around 10% to 20% users receive session QoE below 1 when using server measurements for server selection. These clients suffer

more from freezing events and are apparently impacted by the emulated interference on servers. Our results imply that the measuments (including load, network latency and hops) on servers with interference may not change much, so the clients choose to stay with those servers and their QoE degrade accordingly. In the real-world systems, issues similar to our emulated interference can affect user QoE but may not be reflected in any server measurements, thus the measurement based server selection would always fail in these cases. In contrast, QMan learns server performance from user QoE directly and adaptively selects servers providing good QoE, thus it can go around the challenges of selecting a measurement that correlates with QoE.

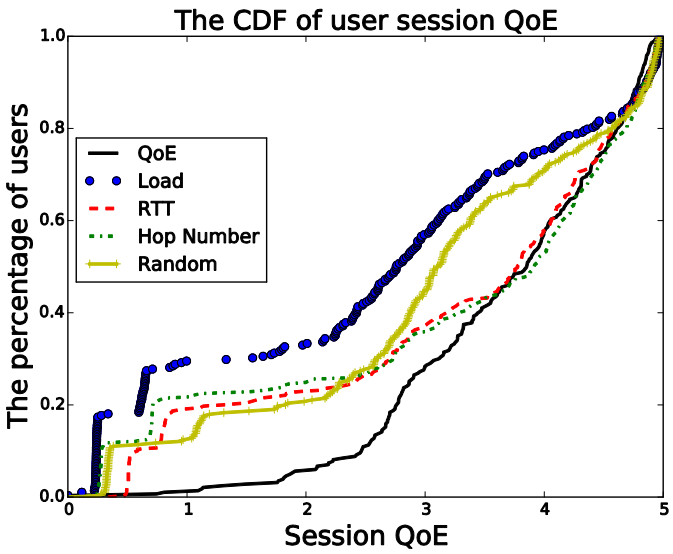


Figure 13: The CDF of session QoE for experiment under severe interference.

3.6.5.5 Experiment under dynamic interference

Figure 14 shows the QoE method has an absolute advantage over the RTT in providing users with better QoE. The RTT performs the worst. The RTT method probes servers every 5 minutes but the interference appears every other minute, so it misses the interferences. Periodic probing fails to capture the dynamic changes of background interference.

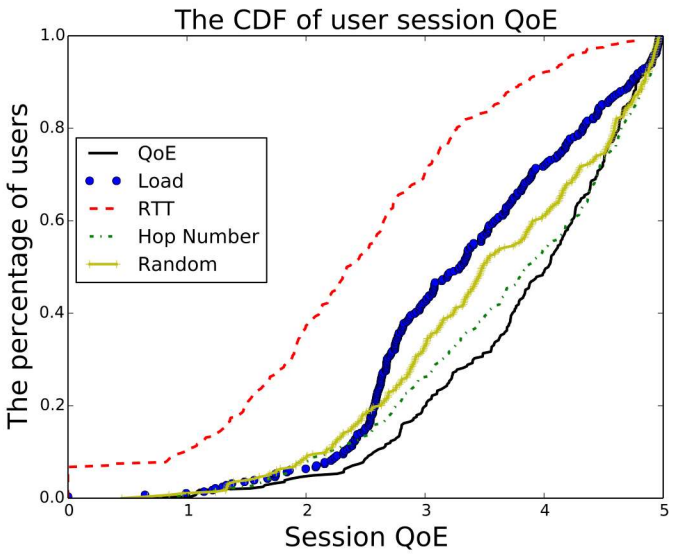


Figure 14: The CDF of session QoE for experiment under severe interference.

3.6.6 Interference & Failures

3.6.6.1 Evaluation under various types of interference

We also test our system under various types of interference in CPU, I/O, and memory. We emulate these interferences by stressing corresponding resources on two randomly selected servers. Our system outperforms other methods similarly. With regard to different types of interference, there is a slight difference on how much the interference impacts user QoE. I/O and bandwidth interference seem to have higher QoE impact than CPU and memory. Extensive experimental results show that our system can manage QoE better. There are more users obtaining QoE above a pre-defined level and better QoE guarantees.

3.6.6.2 Evaluation under failures

We study how our SQS reacts to two types of crash faults on servers in the Cloud. The first one emulates an unresponsive video server on a working VM due to various software errors and bursty user demand. This fault leads a server to hang or crash. The second one emulates an unresponsive VM. This is caused by PM to hang or crash due to various failures in other coexisting VMs.

In Figure 15 and Figure 16, we show an example on how our system reacts to the unresponsive server fault on a working VM. To emulate the fault, we stop the HTTP service on S_8 and stop the VM of S_3 in the middle of the streaming. Figure 15 shows the SQS of all servers learned on cache agent S_{10}. It shows that at time 0:15, the SQS of S_8 dropped to 0 and at time 0:30, the SQS of S_3 dropped to 0. Cache agent S_{10} successfully detected the server crash faults. Figure 16 shows the SQS curves of server S_8 learned from all cache agents through time. These SQS curves show that at time 0:15, the SQS of S_8 dropped to 0 on all cache agents. It shows that all cache agents have successfully detected crash fault on S_8 right after 0:15.

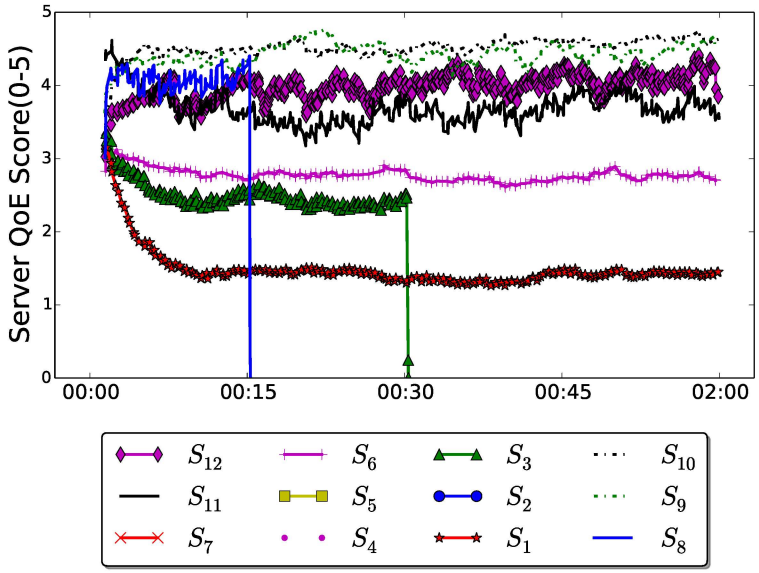


Figure 15: SQS for all servers learned on cache agent S_{10}

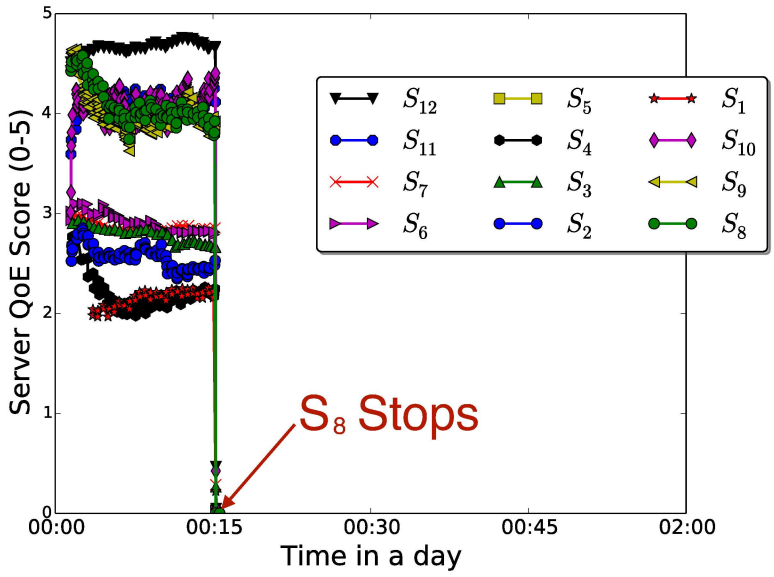


Figure 16: S_8 SQS learned on all cache agents through time.

3.6.7 Decentralized vs. Distributed QMan

As explained earlier, in the decentralized implementation, the cache agent can obtain QoE from many clients to learn the SQSs. In the distributed implementation, the client agent learns SQSs of candidate servers from its own QoE. At a certain time, the cache agent can collect QoE data from multiple users on different candidate servers so it learns the SQS of multiple servers at the same time. However, the decentralized paradigm assumes that all clients in its group have similar QoE on the same server. This assumption may not always hold when clients sharing the same closest cache agent connect to the server through different networks. As a result, the cache agent may select a wrong server for a client.

We then run the RTT based server selection, the decentralized QMan and the distributed QMan in our experiment VoD systems. Around 200 planetlab clients are tested in each run. The cascading QoE model, the weighted averaging SQS learning and the greedy action of server

selection are used in all runs. We stress the outbound capacity of two servers in the experimental system as if there are severe interference on those servers. We draw the cumulative distribution of user session QoE as shown in Figure 17.

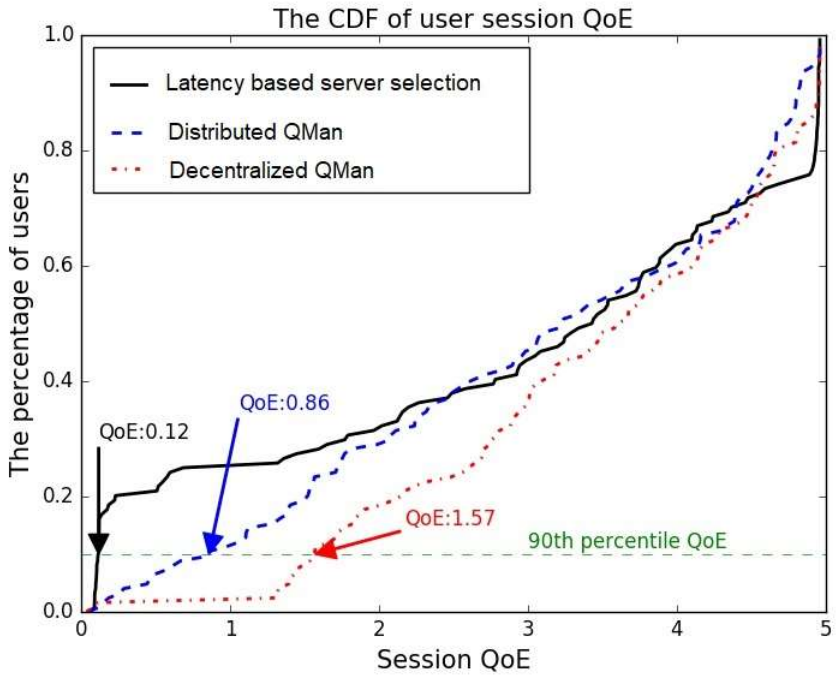


Figure 17: Comparison of session QoE between distributed and decentralized QMan

In RTT based system, we can see that almost 20% of users are heavily impacted by the server interference and they have low QoE (≈ 0.12) all the time. QMan systems have greatly improved the QoE for those users. If we look at the 90th percentile QoE, our distributed QMan can guarantee 90% of users having QoE greater than 0.86 and the decentralized QMan can guarantee their QoE greater than 1.57. Compared to the decentralized QMan, the distributed QMan only has its own QoE data to learn SQS so it would take longer time for the agent to explore the performance of servers. During the exploration period, the agent switches servers often to learn

how each candidate servers perform. We infer that the long exploration period in the distributed QMan lead to less optimal server selections.

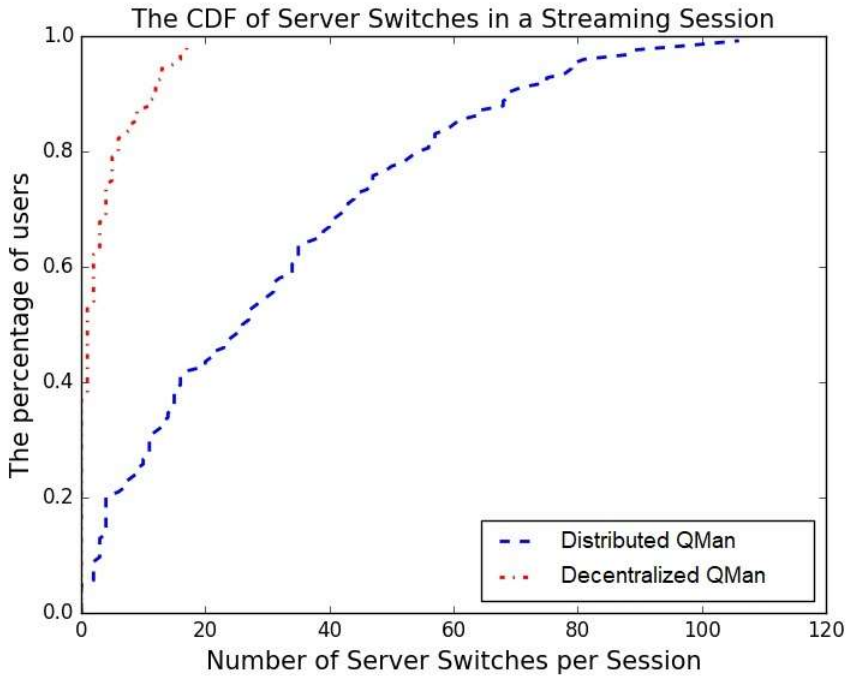


Figure 18: Comparison of server switches between distributed and decentralized QMan

We draw the cumulative distribution of server switches for all clients in Figure 18. It shows that the server selection changes more frequently in the distributed paradigm than in the decentralized paradigm. We pick up two clients at the same location that run the distributed QMan and the decentralized QMan respectively. We plot their QoE and server switches in Figure 19. It shows that the decentralized QMan only switches servers when QoE drops. The distributed QMan changes servers frequently in the beginning to explore candidate servers. The exploration period lasts up to 3 minutes. As in our experiment the testing videos are clipped in 10 minutes, the 3-minute exploration period can degrade user QoE severely. Meanwhile, we also note that in Figure

17, the decentralized QMan obtains better session QoE at almost all percentiles than the distributed QMan. It shows that our assumption about decentralizd QMan suffices for most testing clients.

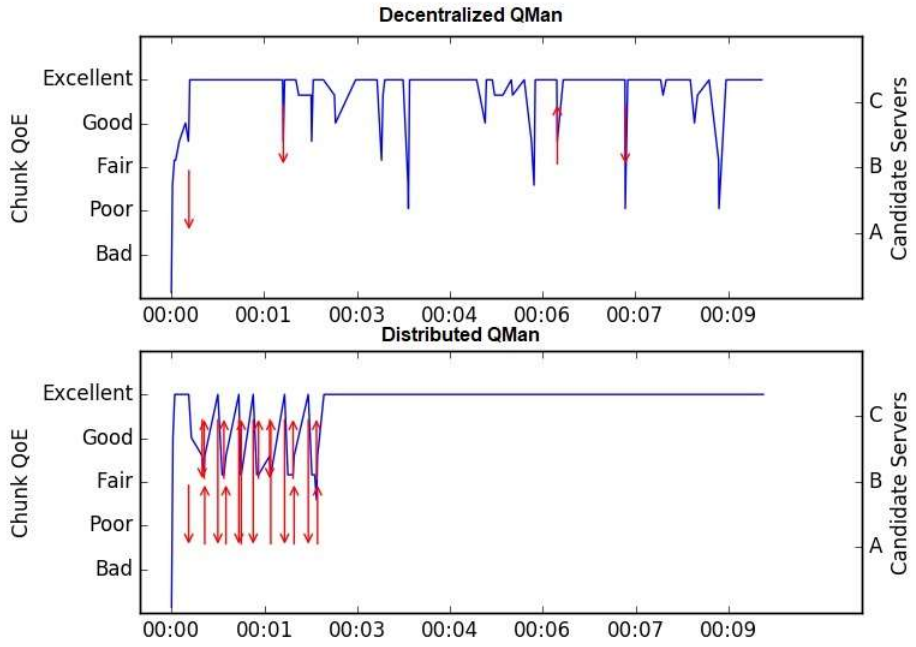


Figure 19: Comparison of server switches over time (distributed vs decentralized QMan)

3.7 Scalability Analysis

3.7.1 Communication cost for QoE based adaptive control

3.7.1.1 Decentralized QMan

In the experimental VoD system running QMan, there are $N = 284$ clients connecting to $M = 12$ cache agents to report their Chunk QoE periodically. On average, each cache agent receives N/M messages per period. Considering the value $N/M = c$ as a constant, the resource is provisioned according to the user demand. The average number of QoE update messages sent to each cache agent is determined by c , i.e., the average number of clients served per server.

3.7.1.2 Distributed QMan

In the distributed QMan, the client agent only queries the cache agent once to obtain a list of candidate servers at the beginning of the streaming session. The message size is fixed and is determined by the number of candidate servers configured.

3.7.2 Communication cost for management tasks

In QMan, cache agents run the MCCD algorithm to discover candidate servers and maintain CST updates according to dynamic operthational changes. There is communication cost involved in such management tasks.

3.7.2.1 Communication cost in MCCD

For MCCD algorithm, we have proved in Theorem 1 that the total outbound traffic to create a CST on each cache agent is proportional to $K \cdot T$, where K is a constant number of candidate servers to be chosen. The traffic is linearly increasing with the number of videos T. Considering that the MCCD only runs once at bootstrapping stage, we believe it is acceptable.

3.7.2.2 Communication cost in CST Maintainance

To maintain the CST on each cache agent, QMan considers both the agent joining/leaving and the video adding/deleting. For agent joining/leaving, one agent needs to notify all other agents to add/delete about its cached video items. It generates at most M messages in the whole system. M is the total number of cache agents. However, because each agent only forwards the message to its neighbors in the overlay network, the per-agent communication is bounded by the maximum node degree in the overlay graph G. Similarly, for the video addition/deletion, our algorithm only notifies the agent's neighbors and the message is forwarded in the overlay network.

3.8 Summary

We propose QMan, a QoE based Management system for VoD in the Cloud. We run QMan in an experimental VoD system deployed in Google Cloud. From extensive evaluations with hundreds of users emulated around the world, we show that by using QoE as a principle to select servers, overall user QoE can improved over common measurement based server selection. The improvement of QoE management lies in the following aspects. First, QMan provides 9% to 30% more users with QoE above the MOS "Good" level than the existing measurement based server selection systems. Second, for users who are impacted by dynamic and severe interference in the Cloud, QMan can improve the QoE from the "bad" to "fair" in MOS level. Third, QMan discovers operational failures by QoE based server monitoring and prevents streaming session crashes. Evaluations also show that the reinforcement learning used in QMan achieves a tradeoff between exploration and exploitation in the server selection. The exploration and exploitation are both necessary in the highly dynimic Cloud environment. By comparing two QMan implementation, we show that the decentralized QMan achieves better overall QoE with less server switches than the fully distributed paradigm. Though the decentralized QMan may introduce more communication cost, we also prove that the per-agent cost is acceptable and the system can adapt to large-scale systems consisting of thousands of servers.

4. QWATCH: A QOE ANOMALY DETECTION AND LOCALIZATION SYSTEM FOR VOD IN THE CLOUD

4.1 Introduction

Video on Demand (VoD) systems are complex. VoD providers, such as Netflix and Hulu, rely heavily on third-party systems including Cloud providers and Content Delivery Networks (CDNs) [26]. CDN, such as Akamai [83], Level 3 Communications and Limelight Networks, provide the content delivery [84]. Cloud providers, such as Microsoft [86] and Amazon [87], manage and provision resource for VoD systems. As there are multiple entities involved in the end-to-end video delivery, it is quite challenging to detect and locate performance problems.

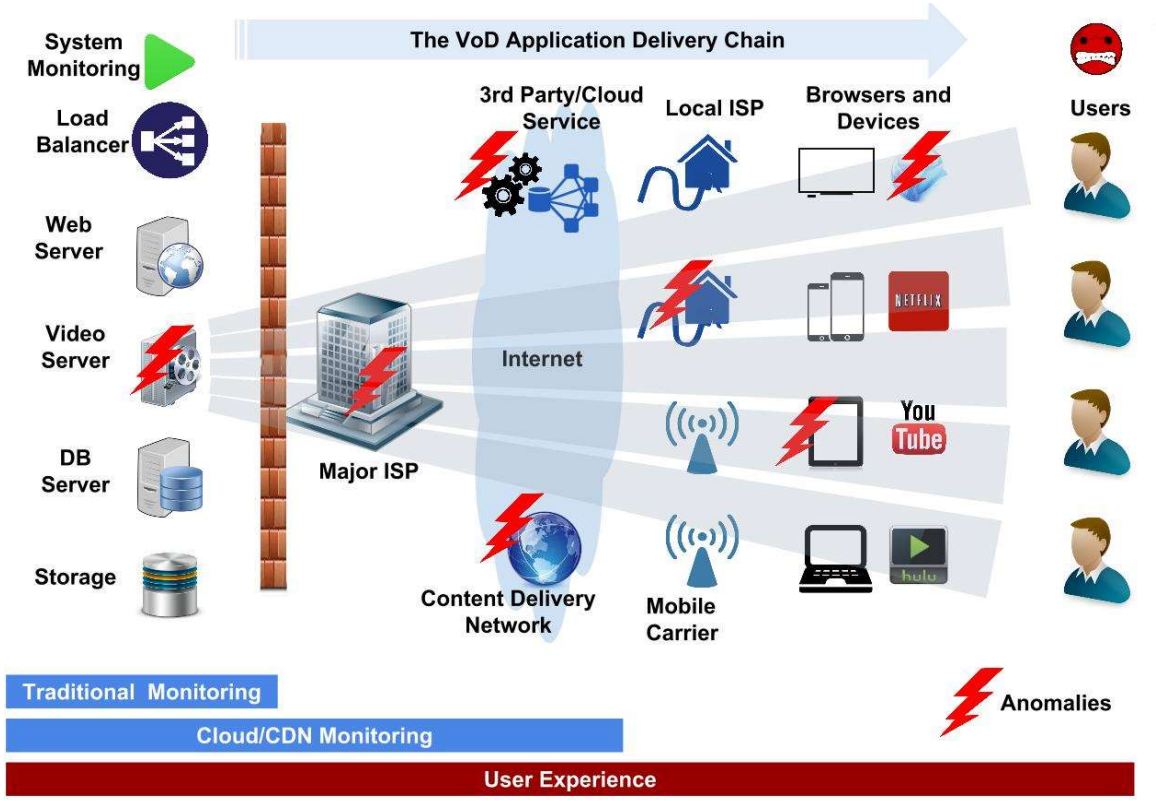


Figure 20: The delivery chain of a VoD application

In Cloud, anomalies arise when Service Level Agreements (SLAs), such as virtual machine (VM) uptime and availability, are violated. We find that Cloud SLA violations sometimes do not influence user QoE. SLA is not sufficient to ensure QoE. System metrics, such as CPU speed, CPU/disk utilizations, disk/memory throughput and network throughput do not fully reflect the user experience of video streaming in the Cloud. There are many other factors in Cloud including transit and client networks affecting the user QoE. The end-user device also plays a significant role in QoE.

Figure 20 shows all components in the VoD delivery chain. Any anomaly in any one of these components can degrade user experience. Each system in the VoD delivery chain only has a partial view of the VoD system. Different entities monitor anomalies independently. Thus, they fail to give a full picture of the VoD delivery chain. Detection and localization of anomalies are very challenging without a clear view of end-to-end VoD delivery chain.

We propose QWatch, a scalable framework, which detects and locates anomalies based on the end-user QoE in real time. The end-user QoE clearly gives meaningful performance of VoD systems. We assume that the user satisfaction is the ultimate performance measure of any complex systems. Regardless of what traditional performance parameters indicate, if the end user QoE is satisfactory, the system is deemed to be operating properly. The end-user QoE masks the complexity of understanding numerous system parameters in various entities in the VoD delivery chain. In QWatch, the end user devices cooperate and share their QoE and path information to detect the location of anomalies or narrow down the areas of possible problems.

We validate QWatch through extensive experiments in a controlled VoD system in production Cloud (Microsoft Azure [86]) and CDN (Amazon CloudFront [87]). Our experiments show that QWatch correctly detects QoE anomalies that cannot be detected using various

network/system metrics. QWatch also avoids false positives in anomaly detection methods based on system metrics. QWatch successfully locates QoE anomalies. We also share several insights obtained from running VoD system with 200 geographically separated users in production Cloud.

4.2 Related Work

Earlier works of anomaly detection in VoD services use different system parameters to infer QoE issues. Ajay et al [92] collect various system metrics in a large IPTV network and apply supervised learning algorithm to learn how anomalies detected in these metrics are related to customer call records. In Cloud and CDN, studies focus on detecting anomalies [30][15][10] based on critical network/server metrics. These metrics are believed to impact end-user QoE. They tend to have many false positives and false negatives. The selection of these metrics is difficult in end-to-end video delivery with many different entities. End-user QoS metric is also used to detect anomaly in [14]. Its detection requires off line computation. There are several works on identifying, locating and diagnosing QoE issues. Junchen et al analyze end-user data by unsupervised learning to find the root cause of QoE problems [14]. Giorgos et al diagnose QoE issues by supervised learning on various network and system metrics from different vantage points [93]. There are also commercial data analysis programs that statistically infer possible root causes of QoE issues (YouTube) [65][17].

4.3 System Overview

4.3.1 Background

VoD systems mainly use third-party CDNs for the content caching and content delivery. CDNs cache popular videos in their edge servers distributed in different geographical locations.

Video contents are delivered to users from the closest edge server. CDN could reduce network latencies from servers to users to improve end user QoE.

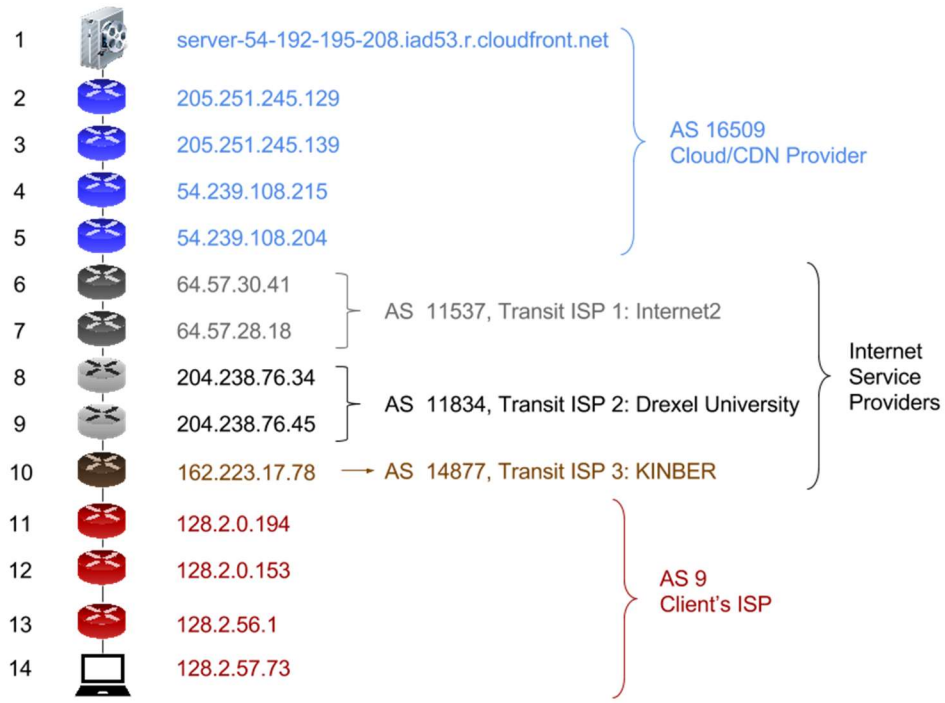


Figure 21: An example video delivery path from AWS CloudFront to Carnegie Mellon University campus network

Figure 21 illustrates CDN operations. We use a device in Carnegie Mellon University network to stream a video from a VoD website cached in Amazon CloudFront [87]. The video in the CDN goes through several networks. These networks are managed by the Cloud/CDN provider and multiple transit Internet Service Providers (ISPs) and a local ISP. Anomalies can occur in any part of this delivery path shown in Figure 2. In this particular experiment, there are five ISPs involved in the end-to-end video delivery path. When the user experiences a poor QoE, it would be very difficult to locate the problem, as there is no viable way to access information from these independent entities in the path.

4.3.2 System Design

QWatch deploys an agent in the video player of the client, referred to as client agent. It evaluates user QoE in real-time. We determine that the VoD has an anomaly when the user QoE drops below pre-determined QoE value q_0.

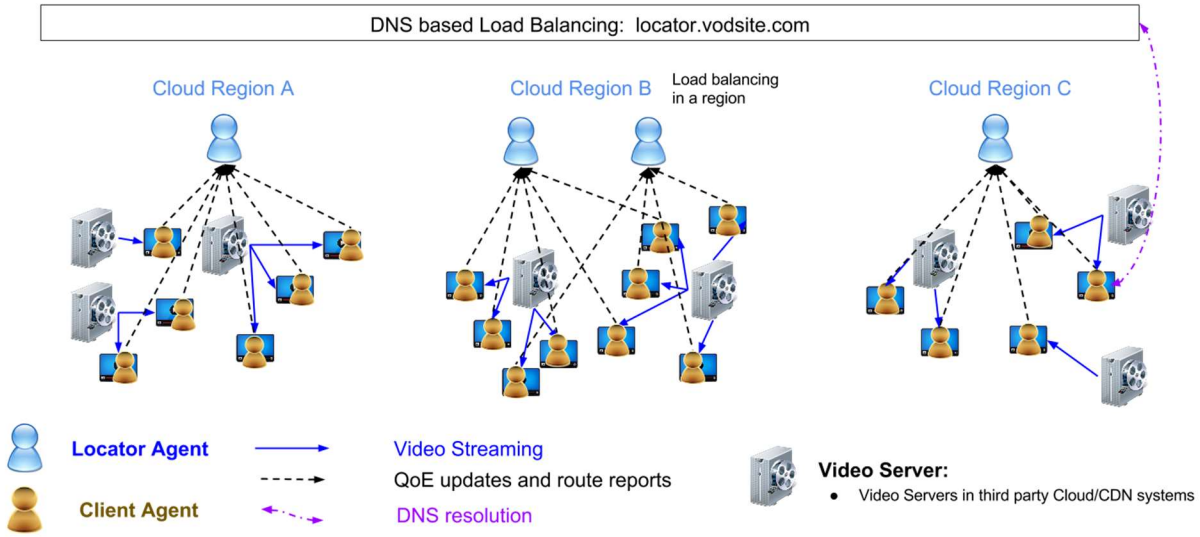


Figure 22: QWatch design with horizontal scaling

When an anomaly is detected, locating the source of anomaly can be challenging, as there are multiple entities involved in the end-to-end video delivery. QWatch reconstructs the underlying network topology using traceroute from users to their CDNs. QWatch then correlates multiple users' QoEs with their network paths to locate the source of QoE anomalies. Correlating QoE data from multiple users allows us to infer normal operating nodes and abnormal nodes. If a user has an acceptable QoE, we assume that all nodes in its video delivery path are functioning properly. If any of these nodes intersect with other video delivery paths, they are excluded from the possible set of anomalous nodes. We develop locator agent to collect traceroute data from users

periodically. The locator agent also collects the end-user QoEs from client agents in real time for the localization of QoE anomalies. Figure 22 illustrates the operation of QWatch.

4.3.3 Scalability

The commercial Cloud allows us to scale QWatch to accommodate the increasing number of users in the VoD system. QWatch clusters users by regions in the Cloud and applies horizontal scaling for locator agents within each Cloud region. Specifically, DNS based load balancing is used to direct users to locator agents in the closest Cloud region. The commercial Clouds, such as Google Cloud, Amazon Web Service [85] and Microsoft Azure [86], provide DNS load balancing services. Within a Cloud region, QWatch provisions one locator agent for clients. For clients, locator agents are provisioned in one Cloud region. Within a Cloud region, simple load balancing mechanisms [88][89] in commercial Clouds can be configured to schedule localization requests among locator agents. The topology is maintained in a database that are shared among all locator agents in one region.

4.4 QoE anomaly detection

The client agent runs the QoE anomaly detection algorithm (QADA) as shown in . The client agent traces its path to the video server and reports to its locator agent. For each video chunk, the client agent evaluates the chunk QoE $q(i)$ and compares with q_0 . If the chunk QoE is greater than q_0 , there is no QoE anomaly. The client agent then reports acceptable QoE to its locator agent periodically. If the chunk QoE $q(i)$ is lower than q_0 , QoE anomaly is detected and the client agent sends unacceptable QoE update to the locator agent immediately. The client agent runs QADA until the streaming session ends.

Data: Reporting period T
Video chunk length: T_0
Minimum acceptable QoE: q_0
Current client agent: C

1: Connect to the closest locator L_k by domain name
2: Get the DASH description file (MPD) from a CDN host by URL
3: Obtain the cache server address S from the MPD file response
4: Probe S by *traceroute* and report the route $R=(C,S)$ to the locator L_k .
5: The reporting period in the number of chunks: $N_T = T/T_0$
6: **while** *video streaming not ends* **do**
7: Download chunk i
8: Compute the QoE for current chunk $q(i)$
9: Obtain current server S_i from the chunk response
10: **if** $S_i \neq S$ **then**
11: Get the route to the current server: $R_i = (C, S_i)$
12: Report the route R_i to the locator L_k
13: **end**
14: Get the receiving time of current chunk t_i
15: **if** $q(i) > q_0$ **then**
16: Acceptable QoE status: $Q_i = True$
17: **if** $(i > 0)$ **and** $((i \bmod N_T) == 0)$ **then**
18: Report update: $U_i = (t_i, Q_i, R)$ to the locator L_k
19: **end**
20: **else**
21: Unacceptable QoE status: $Q_i = False$
22: Report update: $U_i = (t_i, Q_i, R)$ to the locator L_k
23: **end**

Figure 23: QoE Anomaly Detection Algorithm (QADA)

4.5 Topology discovery by *traceroute*

A locator agent receives the route data from client agents and constructs the network topology for the video sessions on those clients. Client agents in QWatch probe their video servers

at the start of each video streaming session and at the time when they change servers. Client agents report *traceroute* data to their locator agents. Clients in the same geographical region could possibly stream from the same video server. We use traceroute data from client agents to construct a server rooted tree graph for the underlying topology. If one client streams videos from multiple servers, then the locator constructs a client-rooted tree graph from *traceroute* to multiple video servers. Upon receiving a route data, the locator agent updates the regional topology accordingly. Route data do not reveal all routers along the path when the router disables the ICMP echo replies. Some routers return private IP addresses. The locator agent eliminates private IP addresses and hidden addresses when it is constructing the topology. The locator agent treats adjoining nodes as connected by a link. QWatch maintains the topology graphs per Cloud region. The locator agent updates if it discovers new nodes and links. The locator agent obtains the ISP name and the AS number of a valid IP node from a commercial API [94]. Router level topology discovery has been well studied in [95][96][97][98] and these methods can be applied in QWatch.

4.6 QoE anomaly localization

4.6.1 Prototypes

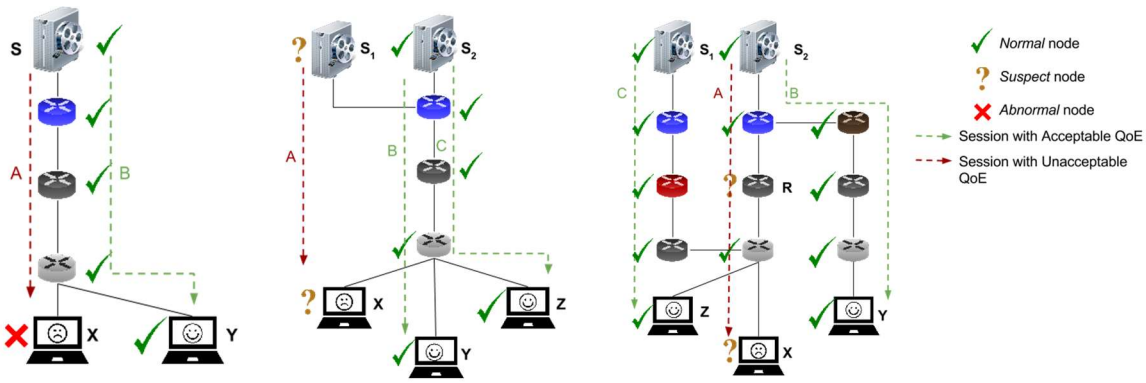
If a streaming session has an acceptable QoE, all nodes in its path are assumed to be functioning well. We assume that if a node has an anomaly, all video sessions going through that node would have unacceptable QoE.

We show examples of how anomalies can be located by analyzing path information of video sessions affected by anomalies. There are three types of nodes.

- Normal node: All nodes on a session's delivery path with acceptable QoE.

- Suspect node: Node on a session's delivery path with unacceptable QoE but does not belong to other delivery paths with good QoE.
- Abnormal node: Node is the only suspect node on a session's delivery path with unacceptable QoE. Rest of the nodes in this delivery path are normal.

If there are multiple suspect nodes in a streaming path, any one or more of these nodes could be the cause of QoE anomaly. When there are not enough clients to resolve the exact location of the anomaly, we classify these nodes as suspect nodes.



(a) QoE anomaly on client (b) QoE anomaly on server S_1 (c) QoE anomaly on router R

Figure 24: QoE Anomaly Localization Prototypes

Figure 24 show prototypes on how anomalies in server, router and client can be located. These prototypes assume all users' QoE can be observed at the same time. In Figure 24 (a), there are two video sessions A and B sharing the same path to server S. Client X perceives QoE anomaly. Client Y has an acceptable QoE and all the nodes through its path are labeled normal. The session A then labels node X suspect. Session A only has one Suspect node in its path. It is clear that the client itself has the anomaly and is labeled abnormal. Figure 24 (b) shows three sessions A, B and C sharing the same path to two servers S_1 and S_2. There is one anomaly

server S_1 and A is connected to S_1. Sessions B and C are connected to S_2 and have acceptable QoE. All nodes in their paths are labeled normal. Then nodes X and S_1 are labeled suspect. If session A does not change its server, we cannot exclude client X from suspect nodes. If session A changes its server to S_2, client X would have acceptable QoE and can be excluded from suspect nodes thus the server anomaly can be located. Figure 24 (c) shows three sessions A, B and C going through different paths to two servers S_1 and S_2. Session A and B connect to server S_2. Session C connects to server S_1. There is one anomaly router R. Sessions B and C have acceptable QoE and all nodes in their path are normal. Session A has two nodes labeled suspect. The router R is then located as suspect. In Figure 24 (a), X is the only suspect node so it can be determined anomaly. In (b) and (c), X is not on the paths of other sessions with acceptable QoE so X cannot be excluded from anomalies. QWatch labels both the anomaly node and the client as suspect nodes. QWatch can provide better resolution if there are more sessions sharing the particular path in question.

4.6.2 Implementation

When the locator agent receives a QoE update, it processes the updates according to the QoE anomaly localization algorithm (QALA) as shown in Figure 25 to label the nodes. It locates the suspect nodes. When the locator agent receives an acceptable QoE update, it retrieves all nodes in the path of the session and labels all nodes normal. If there are no sessions reporting QoE, the node labels expire in Δt seconds. If there is only one suspect node in the path, the node is labeled abnormal. The locator then logs the localization results and listens for the next update messages.

Data: Time window Δt
Node status labeled within Δt is assumed as the present status

1: **while** receiving update $U = (t, Q, R)$ from a client **do**
2: Get all nodes $\{N_i\}$ in R
3: **if** $Q == True$ **then**
4: **for** $N_i \in R$ **do**
5: Update node status: $S_{N_i} = Normal$
6: Label the node: $L_{N_i} = (t, S_{N_i})$
7: **end**
8: **else**
9: Initialize the number of suspect nodes $n^S = 0$
10: **for** $N_i \in R$ **do**
11: Get the latest label on N_i , $L_{N_i} = (t_{N_i}, S_{N_i})$
12: **if** $t - t_{N_i} < \Delta t$ **and** $S_{N_i} == Normal$ **then**
13: **continue**
14: **else**
15: Determine current node status as $S_{N_i} = Suspect$
16: Label N_i with $L_{N_i} = Suspect$
17: $n^s += 1$
18: **end**
19: **end**
20: **if** $n^s == 1$ **then**
21: Find the node N_s with the latest label
$L_{N_s} = (t_{N_s}, S_{N_s})$ where $S_{N_s} == Suspect$
22: Update the label for N_s with the latest label as
$L_{N_s} = (t_{N_s}, Abnormal)$
23: **end**
21: Find all nodes $N_A = \{N_a \mid S_{N_a} == Suspect \text{ or } Abnomal\}$
22: Log QoE anomaly event $E_t = (t, Q, R, N_A)$
23: **end**

Figure 25: QoE Anomaly Localization Algorithm (QALA)

4.7 Experimental Setup

We evaluate QWatch in two environments. The first one is a controlled environment that emulates anomalies at different locations in a small-scale VoD system. The second one is a production environment that deploys the VoD system in Microsoft Azure CDN and AWS CloudFront.

4.7.1 Controlled Environment Setup

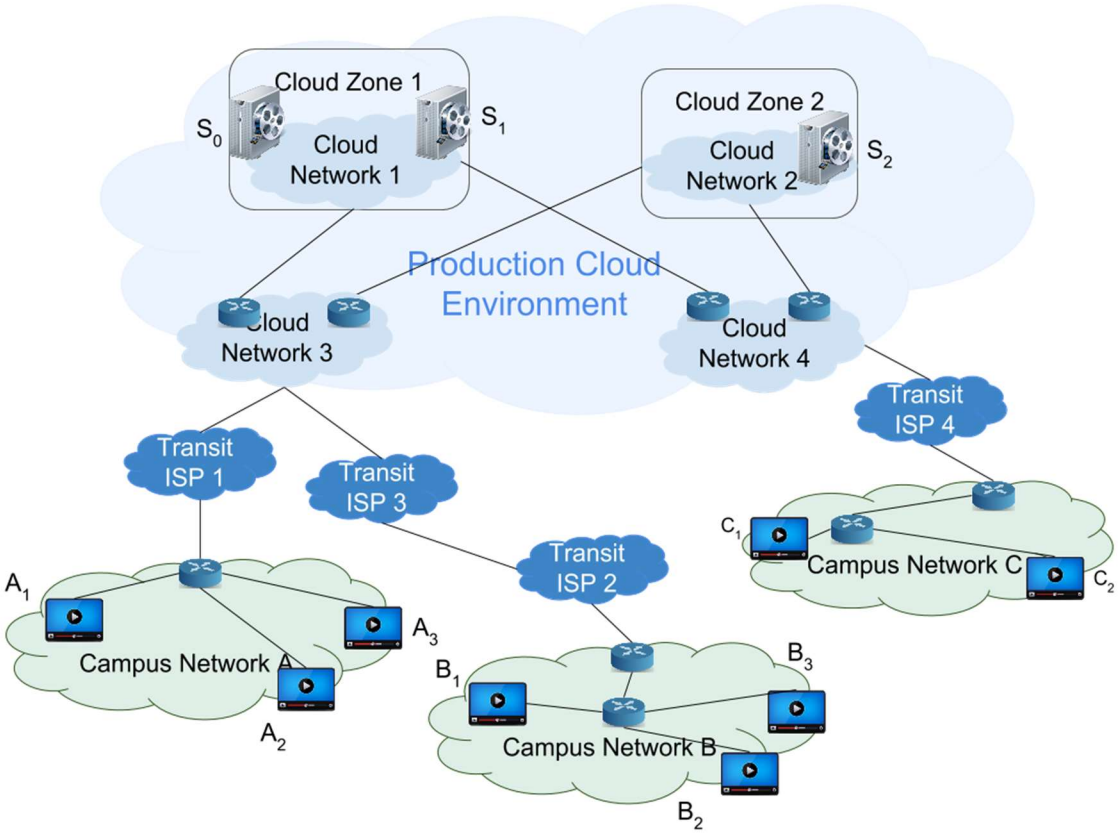


Figure 26: The topology of the controlled VoD

The VoD system runs in 3 servers and 8 clients. The network topology of the VoD system is shown in Figure 6. Three servers are deployed in two regions of Microsoft Azure Cloud [86].

Eight clients are deployed in three campus networks in PlanetLab [80]. A is the network in Rutgers University. B is the network in University of South Florida. C is the network in Emory University. Each campus network connects to the Cloud via different transit ISP networks. There are 4 transit ISPs, 1 Cloud provider, and 3 campus network providers in the experimental VoD system. Clients A_1, B_1, C_1 stream from S_0. Clients A_2, B_2, C_2 stream from S_1. Clients A_3, B_3 stream from S_2.

4.7.2 Production Environment Setup

Figure 27: The locations of QWatch locators and client agents for experiment in production environment

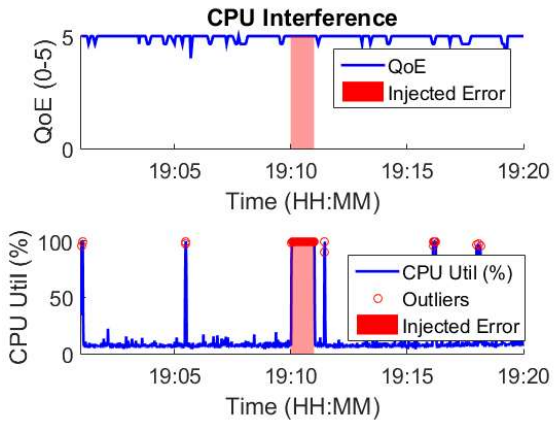
We deploy QWatch in production CDNs (Azure CDN and AWS CloudFront) and analyze QoE anomalies. We configure the caching of CDN to use all edge locations that provide the best performance. We run 200 clients in PlanetLab to emulate users around the world. We provision 5 locator agents in different regions in Azure Cloud to serve 200 clients at different geographical

locations as shown in Figure 27. We choose $K = 100$ and provision locator agents in 5 available zones in Azure.

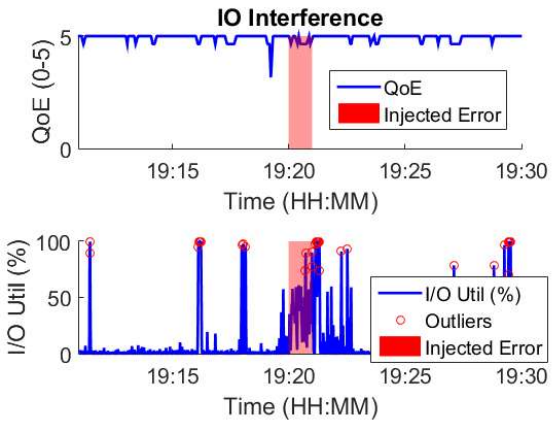
4.8 Evaluation of QoE anomaly detection

4.8.1 Evaluation in controlled environment

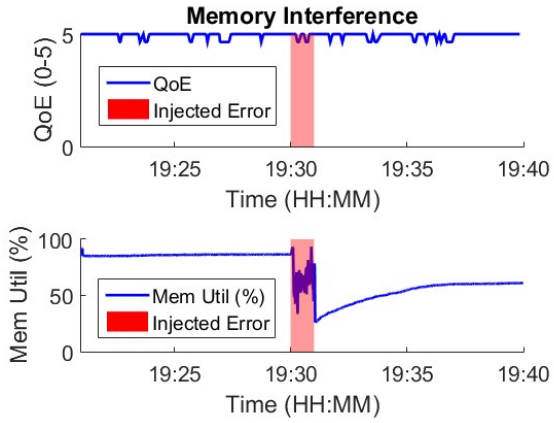
We first consider the effectiveness of system metrics, such as CPU/I/O/memory utilizations, network latency and throughput for anomaly detection in VoD in the Cloud. These system metrics can be obtained in commercial Clouds, such as AWS CloudWatch [90] and Azure Cloud monitor [91]. We show several examples of false positives and false negatives resulting from anomaly detection systems based on system metrics. We then compare QWatch with existing anomaly detection methods. Existing anomaly detection methods find outliers in system metrics [99]. The statistical outlier is defined as data outside the range of 3 standard deviation [100]. We use the statistical outlier detection for a comparison. We let client A_3 stream videos from S_2 and collect various server and network metrics on S_2 . These metrics include CPU, I/O utilization, memory utilization, server outbound traffic throughput (Net Out), network latency between the server and a vantage point (RTT from VP), and the number of TCP retransmissions (TCP retrains #). All the metrics are collected by Performance Co-Pilot [101]. The ICMP ping is probed from S_0 . We inject several faults that appear often in Cloud and networks. These faults include CPU, I/O and memory interferences, network congestion in Cloud/client networks, and packet drops in client network. VM interference are emulated by Stress tool [102] and various network errors are emulated by the Linux network emulator [103].



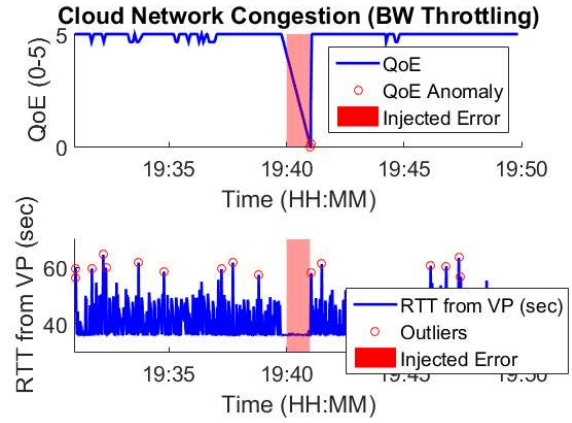
(a) CPU interference



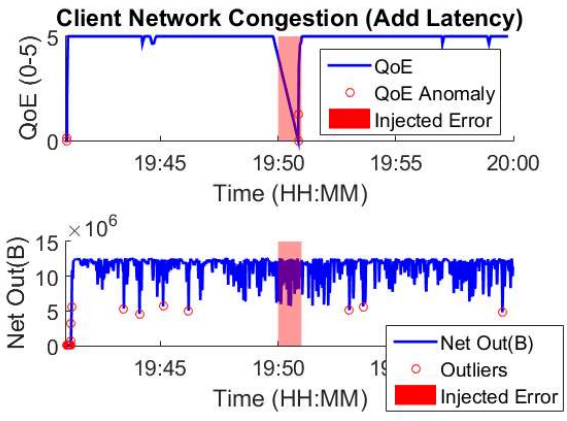
(b) I/O interference



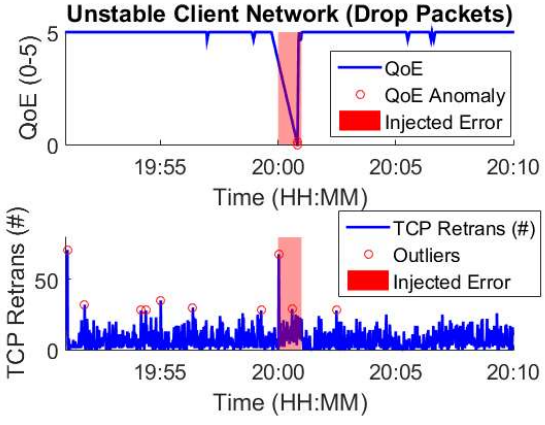
(c) Memory interference



(d) Cloud network congestion (Outbound bandwidth throttled on S_2)



(e) Client network congestion (Long latency to clients in A)



(f) Unstable client network (Packet drops in A)

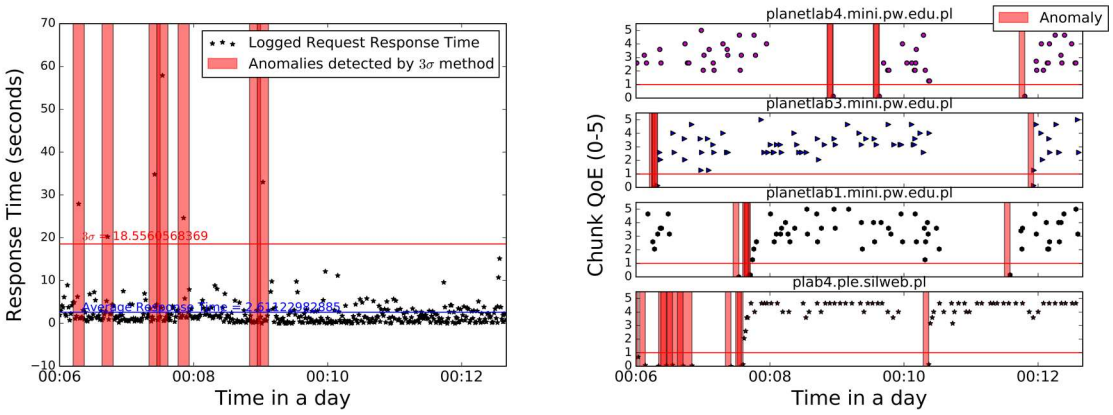
Figure 28: Anomalies in measurements vs. QoE anomalies

Figure 28 shows numerous false positives and false negatives when system metrics are used to detect QoE anomalies. Figure 28 (a) compares CPU utilization metric in the Cloud with the end user QoE. Although the CPU metric triggers an anomaly alarm when the CPU interference is injected, it is not sufficient to create QoE degradation thus resulting in false positives. Figure 28 (b) considers I/O utilization metric with QoE anomalies. Many false positive alarms result from I/O interferences. However, I/O interferences do not influence end user QoE. Similarly, in Figure 28 (c), we show a false positive case where memory interferences impact memory utilization metric but have little impact on user QoE. Figure 28 (d) and (e) compare QoE anomalies with network metrics on server S_2 with network errors. Figure 28 (d) shows that network congestions in the Cloud greatly impact end user QoE. However, the metric based system fails to trigger an anomaly alarm as the vantage point do not capture such QoE degradation. Figure 28 (e) shows that the client network congestions generate QoE anomalies in the client. The metric based system again fails to trigger an alarm in the network throughput of S_2 . These represent false negative cases. The metric based system sometimes correctly detects QoE anomalies when there are numerous TCP retransmissions, namely when the network is unstable. Figure 28 (e) further shows that many other anomalies detected by the TCP retransmission metric do not indicate QoE anomalies.

Cloud monitoring systems use metrics such as CPU speed, CPU/disk utilizations, disk/memory throughput and network throughput [104][105]. These metrics poorly reflect the user experience of video streaming in the Cloud. They fail to account for many other factors that impact user QoE, such as faults in Cloud/transit/client networks and user devices.

4.8.2 Evaluation in production environment

Commercial CDN providers offer their own monitoring systems. It logs errors in the cache server that could impact end user QoE. Common metrics are the HTTP response time, the cache request status (cache/miss), and the HTTP response code. We show how these errors logged in CDN are correlated to QoE anomalies. We run QWatch with a VoD site deployed in Amazon CloudFront on Jan. 9, 2016 from 00:00 am to 01:00 am. CDN logs are compared with several user QoEs shown in Figure 29.



(a) Cloud CDN measurement anomalies (b) QoE anomalies on end users

Figure 29: Anomalies in Cloud CDN measurements vs. QoE anomalies

Figure 29 (a) shows the logged HTTP response time and detected anomalies. There are numerous anomalies detected before 00:10. Figure 29 (b) shows QoE anomalies detected by QWatch. Anomalies in Figure 29 (a) correlate with some QoE anomalies but not all users are affected. Errors logged in the cache server do not cause all QoE anomalies as shown. Video players usually have failover schemes on error responses and maintain a buffer to tolerate temporary cache misses. Other QoE anomalies are shown in red bars after 00:10 in Figure 29 (b). These anomalies

are not captured by measurement in CDN as shown in Figure 29 (a). These experiments in production environment demonstrate that there are numerous false positive and negatives in existing measurement based anomaly detection methods.

4.9 Evaluation of QoE anomaly localization

4.9.1 Evaluation in controlled environment

We inject anomalies at various components including server S1, Cloud Network 1, Campus Network A and Client A1 to evaluate QWatch's QoE anomaly localization. We use the network emulator to throttle the bandwidth capacity for all packets going through different locations. Clients A_1, B_1, C_1 stream from S_0. Clients A_2, B_2, C_2 stream from S_1. Clients A_3, B_3 stream from S_2. Figure 30 shows the entire nodes involved in the experimental VoD system. Later figures only show affected components.

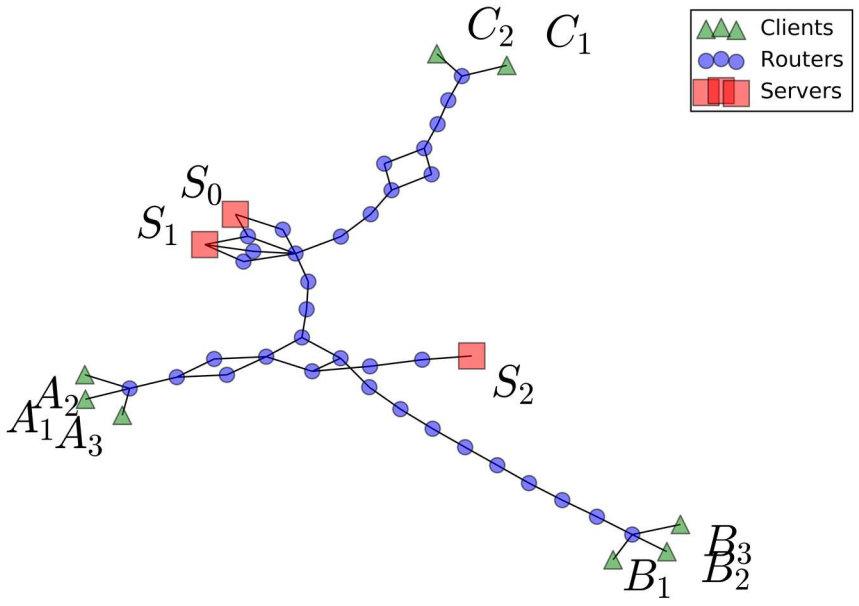


Figure 30: Topology of experimental VoD with entire nodes

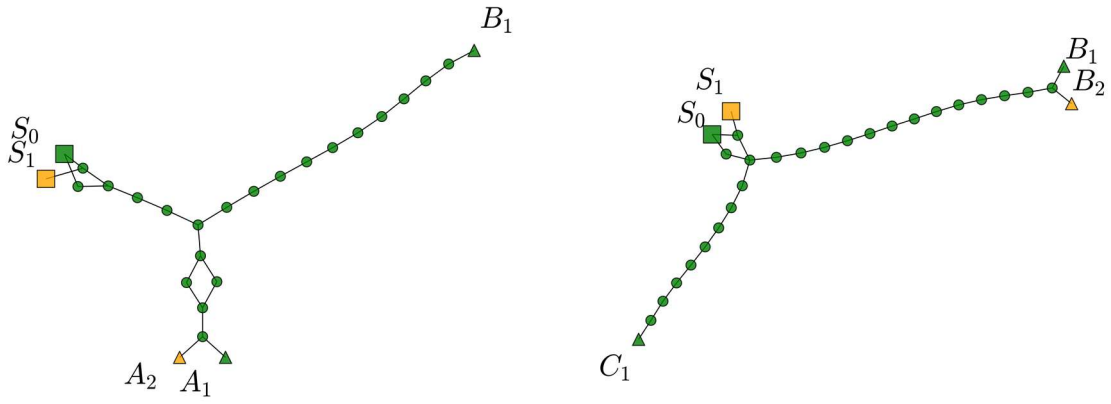


Figure 31: Localization of QoE anomalies at S_1

Figure 31 shows the localization results for two QoE anomalies caused by S_1. Client A_2 and B_2 are affected and their QoEs degrade. A_2 and B_2 have neighbors A_1 and B_1 streaming from another server S_0. They share the same path and shared nodes on their paths are labeled normal. Client A_2, B_2 and server S_1 are labeled Suspect. S_1 is then correctly found as the cause of the anomaly.

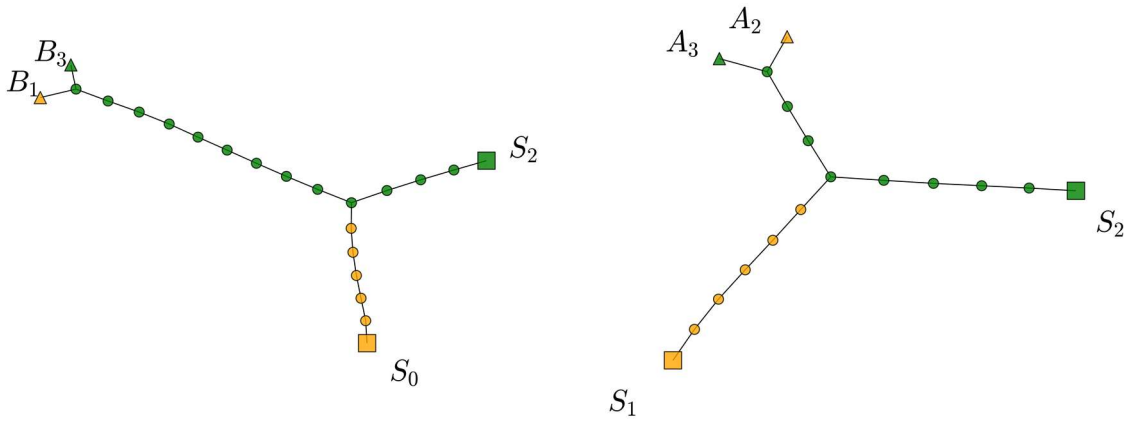


Figure 32: Localization of QoE anomalies at Cloud Network 1

Figure 32 shows two QoE anomalies caused by the Cloud network 1. Client A_2 and B_1 stream videos from S_1 and S_0 through Cloud network 1. Their neighbors A_3 and B_3 both stream from S_2 in Cloud network 2 and they have acceptable QoE. All common nodes shared in client networks and transit ISPs are labeled normal. Nodes in Cloud network 1 are correctly labeled as suspects. Servers connecting to the Cloud network 1 have no anomalies. However, these servers do not provide good QoE and they are labeled Suspect. In this example, Cloud network 1 and servers connecting to Cloud network 1 are both located as suspects. Further troubleshooting is required to obtain localization with higher resolution.

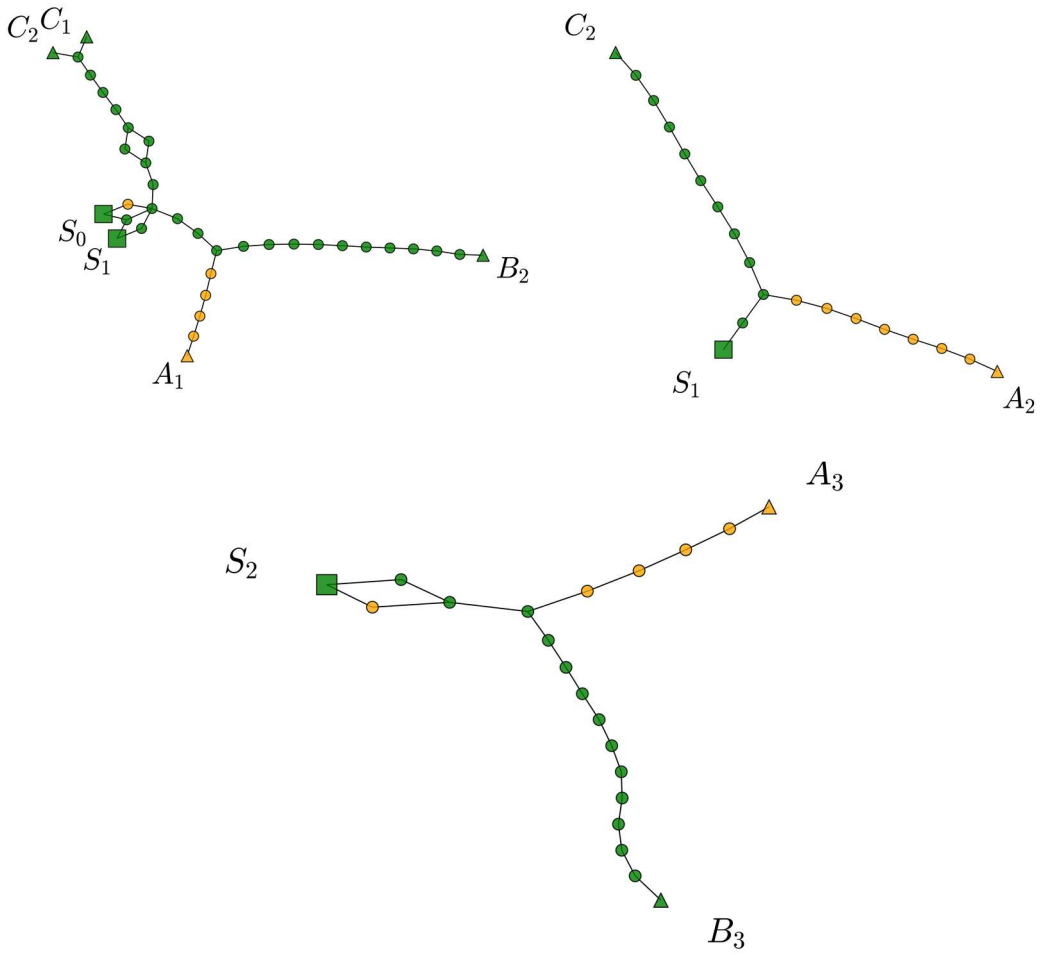


Figure 33: Localization of QoE anomalies at Campus Network A

An anomaly is injected in the campus network A in Figure 33. Clients A_1, A_2 and A_3 connect to campus network A. QWatch correctly labels all nodes in the client network as suspects.

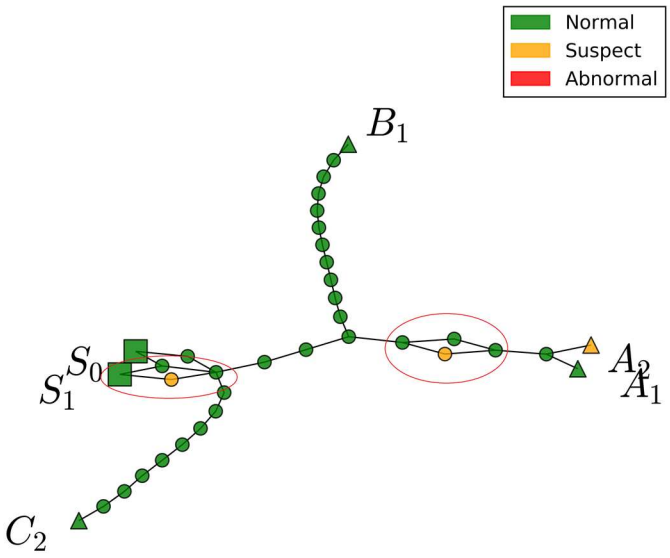


Figure 34: Localization of QoE anomalies at client A_2

An anomaly is injected at client A_2 in Figure 34. QWatch correctly locates the cause of QoE anomaly by labeling client A_2 as suspect. Two nodes that are exclusively on A_2's streaming path are also labeled as suspects. These nodes can be excluded from anomalous nodes if further analysis of topology is performed. Six nodes in the red circle in Figure 34 connect to the same set of nodes that are both on the path (A_1 to S_1) and on the path (A_2 to S_1). We infer that these nodes belong to load balancing networks. These nodes should be excluded from Suspects as a whole because these load-balancing networks are on the path of client A_1 with acceptable QoE.

4.9.2 Evaluation in production environment

We run QWatch on Windows Azure CDN on April 14, 2016 from 15:30 to 16:30. Results of QoE anomaly localization are similar to those shown in Figure 31 to Figure 34. Figure 35 shows the count of QoE anomalies located in different components. The data are collected from the

locator agent in the east US region. There are 219 QoE anomalies detected during 1 hour in the region. Figure 35 shows that all QoE anomalies label clients as Suspects. Interestingly, we do not observe any adaptive server selection strategies in Azure CDN. There are 35 clients in east US region and they all stay with the same video server during the period of experiment. Therefore, when a client has QoE anomalies, there is no other video delivery path providing better QoEs. Thus, the client itself remains as a Suspect node. We find that Azure CDN assigns users in a very broad area (i.e. from Ottawa to Florida) to the same server. In a large geographical area (i.e. US east, US west, Europe), users are assigned to servers that are relatively close to them in terms of network or geographical distances. We do not know the details of Azure's server allocation algorithm. Surprisingly, Azure's algorithm is not as dynamic as we would expect.

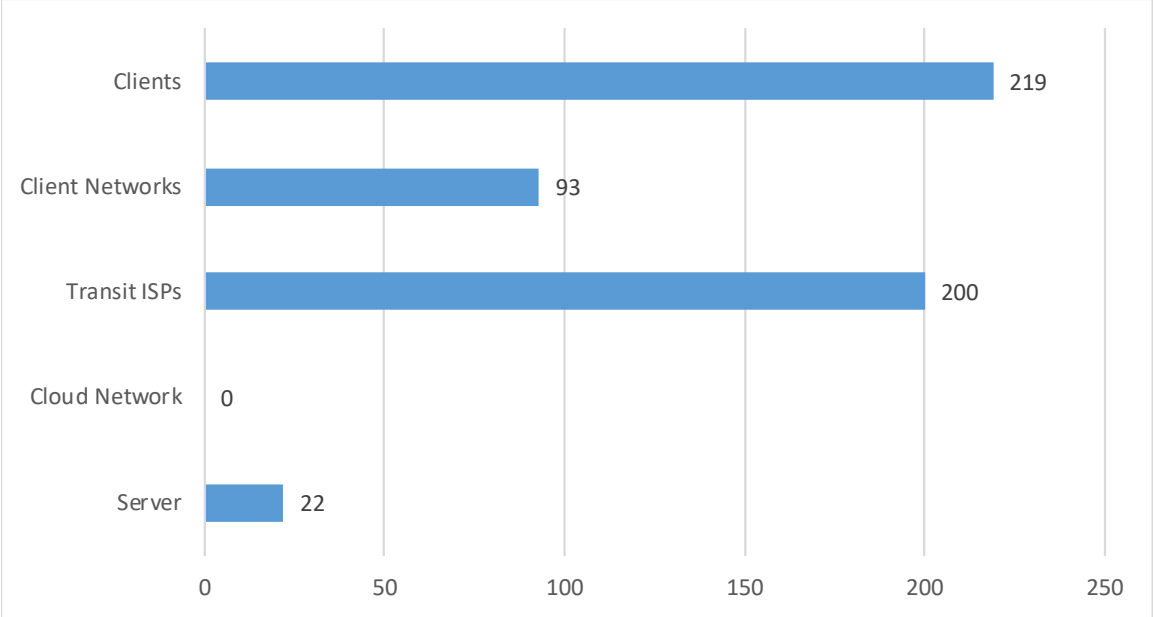


Figure 35: QoE anomalies located in different components in production envrionment

A large number of QoE anomalies are also located in transit ISPs. Our 200 clients around the world connect to Azure CDN through different ISPs. Results show that a majority of QoE

anomalies are located in transit ISPs and clients as Suspects. We notice that there are few anomalies located in servers and there is no anomaly located in the Cloud network. The localization graphs for QoE anomalies in servers show that most of these anomalies label server and other components as Suspects at the same time due to the limited number of video sessions. The number of QoE anomalies located in the servers is relatively small compared to QoE anomalies located in clients and transit ISPs. QWatch would have better resolution identifying server and transit ISP anomalies if our experiments had larger number of users.

4.10 Scalability Analysis

The locator agents are deployed on Basic A1 VMs in Microsoft Azure. The average time to locate a QoE anomaly is 200 ms. We have only one locator agent per region and the topology database is deployed in the locator agent. All client agents in one region report QoE updates to the locator agent every minute. The processing time per update depends on the number of hops in the video delivery path. $K = 100$ does not result in request failures in locator agents. As the number of users increases, QWatch can horizontally scale the locator agents. The network size per region is bounded by K and the length of the path. The length of the path is usually below 50 hops. As the number of users in one-region increases, the distributed database can adapt to maintain the underlying topology.

4.11 Summary

QWatch uses end-user QoE to detect QoE anomalies and correlate users' data to locate QoE anomalies. We run extensive experiments in a controlled VoD system and production Cloud (Azure Cloud and CDN) to validate QWatch's accuracy in detection and localization. We find numerous false positives and false negatives in production Cloud when system metric based

anomaly detection methods are used. QWatch correctly detects and locates anomalies in controlled experiments. In production Cloud, we validate QWatch with 200 users. Results show that a major of QoE anomalies are located in clients and in transit ISPs. No QoE anomalies are found in Cloud networks. Compared to clients and transit ISPs, servers and Cloud networks are less likely to cause QoE anomalies. Interestingly, Azure CDN's server allocation algorithm may not be as dynamic as we would expect.

5. QRANK: A QOE ANOMALY IDENTIFICATION SYSTEM FOR VOD IN THE CLOUD

5.1 Introduction

Modern commercial VoD systems, such as Netflix, HBO, and Amazon Prime Video, are complex. Their videos are produced and stored in Clouds [106], such as Amazon Web Service, Microsoft Azure, Google Cloud. The video contents are distributed and cached in Content Clouds [83], such as Netflix open connect, Akamai, Level 3, and Limelight. The video traffic is delivered to users via Internet. Multiple stakeholders, including the VoD provider, the Cloud providers, the CDN providers, and the Internet Service Providers are involved in the video streaming. When users have poor Quality of Experience (QoE), it is important for the VoD provider to identify faulty or overloaded systems for QoE degradations.

VoD systems deliver videos via many heterogeneous systems including servers, Cloud/CDN networks, transit networks, access networks, and user devices. Any of these systems could have anomalies degrading user QoE. For example, resource exhaustion on servers can increase the servers' response time thus degrading user QoE. Similarly, the resource exhaustion on user devices may freeze the video playback or crash the video player. Low capacity and traffic congestion in access and transit networks degrade the quality of streaming videos. Different stakeholders, namely the Cloud provider, the CDN provider, the transit ISPs, the access ISPs and the users, manage the systems involved in the video delivery. These providers have their own monitoring systems and target different Service Level Agreements (SLAs). As their objectives vary, maintaining QoE would be challenging. For example, YouTube monitors user QoE per ISP as they assume that access ISPs to be the most likely capacity bottlenecks for high-definition (HD)

video streaming [13]. Studies in a large European ISP show that Google's CDN server selection policy might be the cause of QoE degradation [17]. To identify the anomalies, VoD providers need a unified metric that reflects user QoE. VoD providers do not have enough visibility over other systems. The Cloud/CDN systems select servers for users. Depending on the selected server, video traffic could go through different networks to get to the access network. The Cloud network, multiple transit networks and the access network together determine the route to deliver videos. Load balancing in these networks further complicates the video routes. Video from the same server could go through different routes to the user depending on load balancing policies. Without knowing the underlying network topology for a particular video delivery, it is very difficult to find which system causes QoE anomalies.

In this chapter, we propose QRank, an anomaly identification system that identifies the bottleneck system causing QoE anomalies. QRank detects QoE anomalies based on QoEs monitored by users at run time. QRank discovers the underlying network topology and all systems for video streaming by traceroute measurements. We assume that the system with users that experience more QoE anomalies or lower QoEs is more likely to be the system causing QoE anomalies. QRank identifies the anomalous system by ranking the QoE scores in different systems. We validate the effectiveness of QRank through extensive experiments in a controlled VoD. In our experiments, we inject QoE anomalies in user device, access network, transit network, cloud network, and server to degrade user QoE. QRank correctly identifies the anomalous system in these cases. QWatch locates anomalies in a wide range of suspect nodes in different systems. QRank successfully pinpoint the anomalous system, which can be a server, a user device or a network managed by a specific provider at a specific location. We run QRank in a production VoD deployed in Azure Cloud with 100 users emulated in PlanetLab and 24 users emulated in Azure

Cloud. The results show that access, transit networks and user devices contribute mostly for QoE degradations. 61.97% QoE anomalies identify access or transit networks as anomaly systems and 38.14% identify the user devices as anomaly systems. Cloud networks and servers seldom cause QoE anomalies.

5.2 Related Work

5.2.1 Analysis of QoE degradations

Existing studies collect and analyze the QoE measurement from YouTube [11][17], large-scale video streaming events [109], and Internet streaming services [14]. Casas et al study the correlation between the server changes and the QoE relevant degradations [11][17]. They find that the root causes of QoE degradations are Google CDN's server selection strategies. The measurement is done in one ISP and their conclusion may not be true for users worldwide. Conviva [14] collects QoE measurement data worldwide from 379 video service providers. They cluster QoE anomalies over the space of client/session attributes, including the CDN, the client AS and the connectivity type. They ignore many other systems involved in the video delivery, e.g. transit networks. Adnan et al analyze QoE for a live streaming event in North America and find lower engagements for users with low QoE [109].

5.2.2 QoE anomaly localization and diagnosis

QWatch locates nodes that are exclusively on the routes of users with QoE anomalies as suspect nodes [108]. However, node-level localization provides little insights about the systems and had low accuracy when there is a dynamic routing. Dimopoulos et al diagnose QoE anomalies

for video streaming on mobile devices by correlating QoE anomalies with anomalies detected in network/device system measurements. Intrusive measurements are hard to obtain from commercial VoDs [93].

5.2.3 QoE based learning of system performance

QoE driven control systems [55][107] learn dynamic server performance via end user QoE at run time and perform adaptive server selections to optimize overall user QoE. Pytheas [110] correlates user QoE over session attributes, including CDN and bitrates, and adaptively control the combination of CDN and bitrate selections by reinforcement learning techniques. They learn server or CDN performance via QoE measurements. QRank uses end user QoEs to learn the performance of all systems involved in video delivery, including the Cloud/transit/access networks and user devices.

5.3 Background

5.3.1 Root causes of QoE anomalies

Various studies analyze QoE anomalies through collecting QoE measurements from video streaming service on mobile devices [93], YouTube streaming service [11][17] and various anonymous Internet streaming services [14]. They apply statistical analysis and machine learning techniques on collected QoE data. They highlighted following systems as anomalous systems causing QoE degradations.

- Cloud/CDN servers [11][17]: Users can be directed to congested, disconnected or faulty CDN servers.

- Internet [17]: Inter-AS routing changes and inter-AS path congestion reduce the network capacity from end user to server, consequently degrading end user QoE.
- Transit network [17]: Within a transit network, the latency increases due to burst traffic, equipment failures or routing misconfigurations.
- Access network [14] [17] [93]: Overloading at logical access aggregation points, misconfigurations in access networks, and equipment failures are noted. Limited capacity in access networks also slow down users' connection impacting user QoE.
- User devices [17] [93]: Various devices including tablets, TVs, mobile devices, home routers, and set-top boxes have issues such as resource exhaustion, misconfiguration, device memory leaks and software/hardware failures.

5.3.2 Systems incurring QoE anomalies

VoD systems mostly use third-party CDNs for content caching and delivery. CDNs usually cache popular videos in the edge servers that are closer to users. When a user requests a video, the "closest"' server[1] responds with the video. We use a Planetlab server in a campus network to stream a video cached in Microsoft Azure CDN. We probe the CDN server that caches the video and discover the underlying network topology as shown in Figure 36. The video traffic is delivered from the CDN server to the user server through multiple networks: the Cloud network, several different transit networks, and the campus network. VoD providers do not have control and direct access to routers. As routers belong to different ISPs, VoD providers need to work with ISPs when

[1] CDN providers usually determine the criterion of choosing the "closest" server for a user. DNS based server selection [29] is generally used to approximate selecting the "closest"' server in terms of network latencies.

routers in their networks incur QoE anomalies. We therefore group routers into networks. Each network is composed of routers managed by a specific ISP. We further classify those networks into Cloud networks, transit networks and access networks. Cloud/CDN providers manage servers and users manage devices. The VoD providers need to identify if these systems cause QoE anomalies and work with the Cloud/CDN providers and users to improve user QoE. In summary, servers, user devices, and networks are possible anomalous systems causing QoE anomalies. We see that the possible anomalous systems are the server (72.21.71.200), the cloud network (AS 15133), several transit networks (AS 11537, AS 11164, AS 11834, AS 14877), the access network (AS 9) and the user device (planetlab-3.cmcl.cs.cmu.edu).

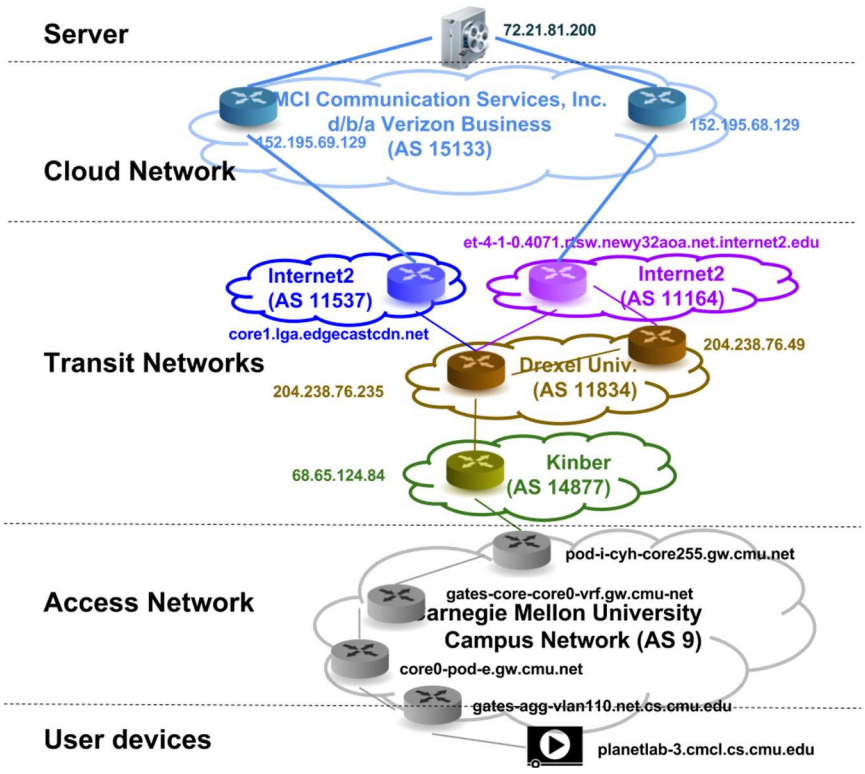


Figure 36: The underlying topology involved in the video streaming from a cache server in Microsoft Azure CDN to a user in Carnegie Mellon University

5.4 System overview and design

5.4.1 System overview

QRank has 3 main operations: 1) select suspect systems that may cause QoE anomalies, 2) evaluate the performance of suspect systems with user QoEs, 3) identify the true anomalous system among suspect systems by ranking the QoE scores in suspect systems. QRank needs to determine whether a server/device/network could be a suspect system. According QWatch, nodes including server and routers on well-experienced user's route have no anomalies. Therefore, the server with high user QoE cannot be a possible anomalous system. The network consisting of routers only on well-experienced users' routes is less likely to be an anomalous system. QRank extends the QWatch system by clustering the suspect nodes into suspect systems including servers, networks and devices, and identify the anomalous system by ranking the aggregated QoEs among suspect systems.

QRank system operates in following steps. 1) **QoE anomaly detection**: QRank detects QoE anomalies for all users. 2) **QoE anomaly localization**: QRank uses anomaly localization algorithm QALA in QWatch to locate suspect nodes that may cause QoE anomalies. 3) **Detection of QoE anomaly systems**: QRank groups suspect nodes into suspect systems, namely servers, networks and devices. 4) **System QoE score learning**: QRank computes the QoE scores in suspect systems during the anomaly period, to represent the system performance. 4) **QoE anomaly identification**: QRank ranks all suspect systems according to their QoE scores (from low to high) and identify the one with the lowest QoE score as the true anomalous system.

5.4.2 System design

We implement QRank as a decentralized agent based system. QRank deploys client agents to run on all user devices to monitor chunk QoEs at run time. Client agents probe cache servers periodically. QRank also deploys decentralized cloud agents in Microsoft Azure Cloud to collect QoE and *traceroute* measurements from client agents in each region. The client agent measures QoE for every chunk and reports QoEs to the closest cloud agent every N chunks. The client agent probes cache servers every 10 minutes and reports the *traceroute* data. The client agents only monitor and report QoE and traceroute measurements. The cloud agents analyze collected data and identify QoE anomalies in run time.

5.5 QRank System

5.5.1 QoE anomaly detection on cloud agent

Client agent on a user device reports QoE measurements to its cloud agent every N chunks. If a chunk of video is a T second video segment, the client agent reports N chunk QoE values every $T \times N$ seconds when there is no freeze. The length of a video chunk is commonly $T = 5$ seconds. In QRank implementation, we choose $N = 12$ and when there is no freezing the client agent reports QoE values every 1 minute. Smaller N would incur more QoE reporting traffic and larger N would delay the detection of QoE anomalies.

The cloud agent then detects QoE anomalies based on N chunk QoE measurements. If there is any chunk with QoE value less than q_0, the cloud agent alarms that there is a QoE anomaly. Sometimes, the cloud agent alarms QoE anomalies in every N chunk period for a user. These alarms are regarded alarms for the same QoE anomaly that last more than $T \times N$ period, as shown

in Figure 37. In such cases, the cloud agent determines the start and the end of the anomaly by the time-stamps of receiving the first and the last chunk with QoE below q_0 .

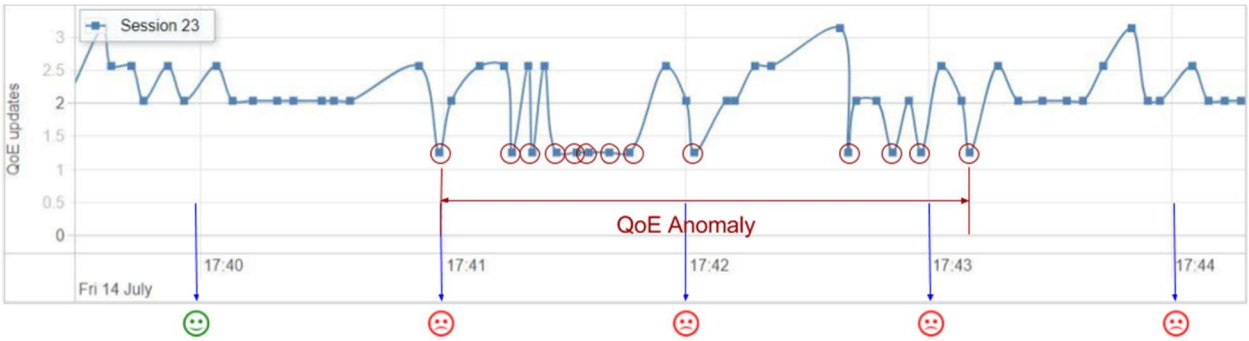


Figure 37: Example of QoE anomaly detection

5.5.2 Detection of anomalous systems

By running QALA, the cloud agent obtains a list of suspect nodes for QoE anomaly. These nodes can be routers, servers or user devices. Suspect routers can belong to different networks. QoE anomalies are usually incurred by issues in those network systems instead of faults on routers. A network with limited capacity can cause QoE anomalies but all routers in the network may function well. In QRank, a network consisting of suspect routers is determined as a suspect system. If a suspect node is a server, the server is also a suspect system. Servers can have various issues causing QoE anomalies, such as insufficient capacity, resource exhaustion, etc. The server as a suspect system impacts QoE for all users streaming from it. If a suspect node is a client node, the QoE anomaly can be caused by various faults on the user device. Thus, user devices are also suspect systems.

QRank determines suspect network systems by all suspect routers. Each suspect router is associated with a public IP address. Given an IP address, QRank uses public databases including

ipinfo [94], iplocation [111], and bgpview [112] to find the ISP the router belongs to. The Autonomous System (AS) number and the geographical coordinates of the router can also be discovered. As information among those 3 databases may not be consistent, QRank queries all three databases and chooses the information that is agreed by more databases as an accurate information of an IP. QRank also maintains a database to cache information for all discovered IPs as shown in Table 2. Given the geographical location, the AS number, the ISP name of routers, etc., QRank discovers suspect network systems.

Table 2: The ISP and location information of an IP

IP	ISP Name	AS #	Geo- Location	City, Region, Country
205.213.119.30	WiscNet	2381	(43.1184, -89.5207)	Middleton, Wisconsin, USA

As QoE anomalies incurred by devices are usually software bugs or hardware issues, those anomalies can reoccur on other users who use the same type of devices. Thus, we define a suspect device system as a type of devices that have same attributes leading to similar faults. In a production VoD system, these attributes can include but not limited to the device type (e.g. Tablet, Laptop, TV, mobile phone, etc.), the device Operating System (e.g. android, iOS, MacOS, Windows 10), the mobile application version or the browser version (FireFox, Chrome, IE, etc.), and the software version of the video player (dash.js, Adobe Flash Player, Azure media player). VoD providers can study common device faults affecting QoEs to determine what attributes to choose. In all experiments, we emulate users in PlanetLab nodes and Azure VMs. The device type can be PlanetLab server (abbreviated as PL) or Azure VM (abbreviated as AZ). The PlanetLab servers are installed with Fedora and CentOS and Azure VMs are installed with Ubuntu, so the OS

version can be Fedora/CentOS/Ubuntu. All emulated users ran an emulation code of DASH player we wrote in Python. Thus, the software framework version is the Python framework version and the video player is the emulated DASH player, denoted as EM-DASH. We later injected faults in the emulated DASH player denoted as EM-DASH-ERR.

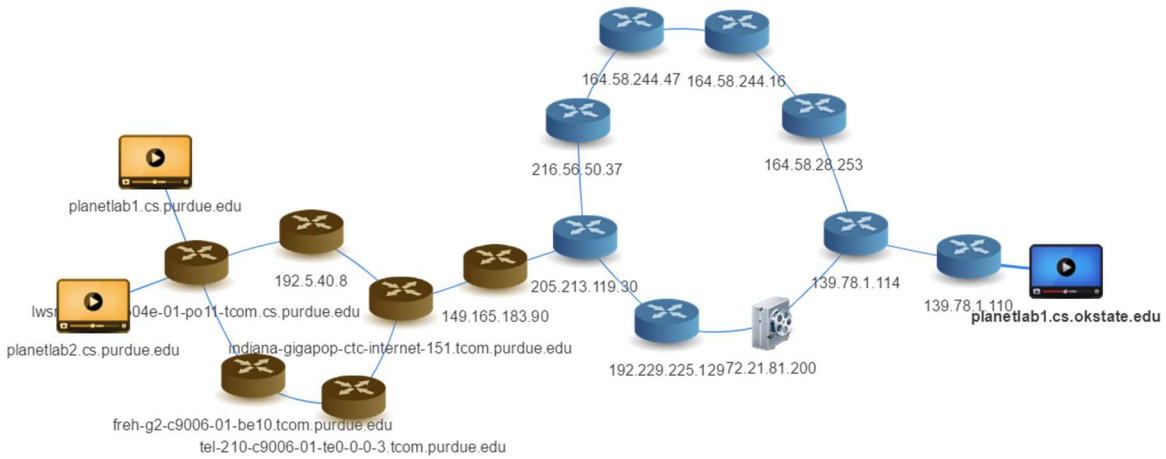


Figure 38: Localization result for an example QoE anomaly

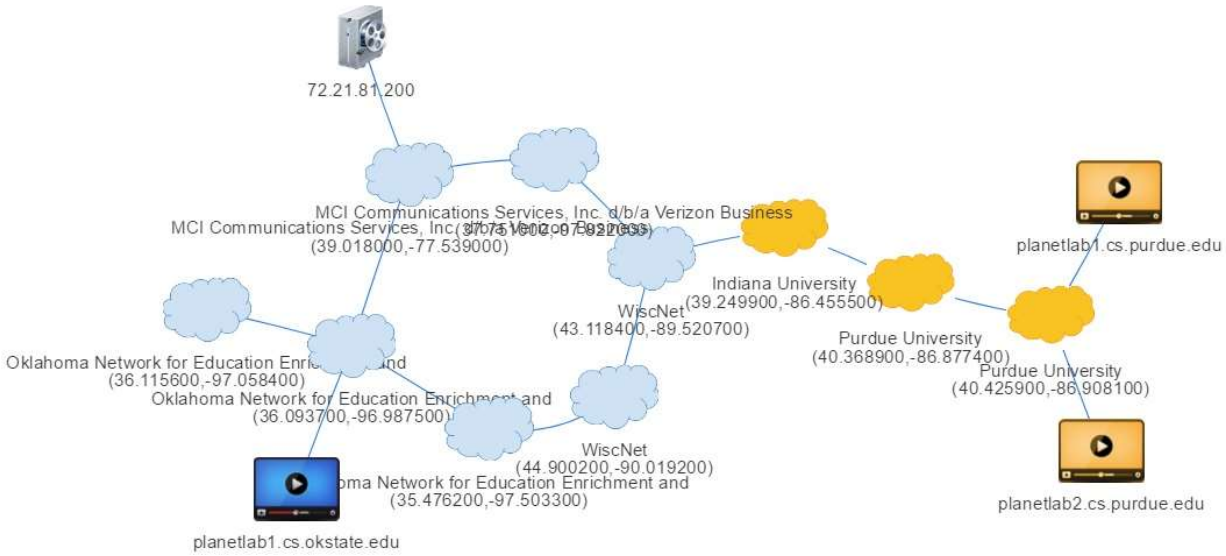


Figure 39: Suspect systems for the example QoE anomaly

We use an example QoE anomaly to show how the suspect systems are detected. Figure 38 shows the localization result for an example QoE anomaly detected on "planetlab2.cs.purdue.edu" at 7:11 am on June 4, 2017. Given suspect nodes located in Figure 38, QRank then detects suspect systems in brown as shown in Figure 39. The suspect systems detected for the example QoE anomaly include the followings. A transit network managed by Indiana University at (39.2499, -86.4555), a transit network managed by Purdue University at (40.3689, -86.8774), an access network managed by Purdue University at (40.4259, -86.9081) and a device.

The device is a Planetlab server installed with Fedora 14 OS running our emulated DASH player in Python 2.7 framework. Such type of devices is denoted as "<PL, Fedora14, Python2.7, EM-DASH>".

5.5.3 System QoE Score learning

QRank infers the performance of these systems via end user QoEs. We compute a QoE score based on QoEs of users using the system (also refer to as related users). If the suspect system is a network, QoE of related users whose video traffic deliver through the network are counted. If the suspect system is a server, QoE of related users who stream videos from the server are counted. If the suspect system is a device, then QoE of related users using the same type of devices are counted. As users' QoEs change over time, we compute the QoE score for a system by the averaging of QoE in a shifted time window. The system QoE score is computed by Equation (2.9)

.

$$Q(o,t_1,t_2)=\frac{1}{|U_o|}\sum_{u\in U_o}\sum_{t_1\le t\le t_2}\frac{q_u(t)}{n_u(t_1,t_2)} \tag{2.9}$$

$U_o = \{u\}$ denotes the set of users using the system o. $q_u(t)$ denotes the chunk QoE value for user u at time t. $n(t_1, t_2)$ denotes the number of video chunks received on u during time $[t_1, t_2]$. QRank learns the QoE scores for all suspect systems during the anomaly period. The time range $[t_1, t_2]$ denotes the start and the end time of a QoE anomaly to be identified. In Table 3, we compute the QoE scores for all suspect systems retrieved for the example QoE anomaly. For the network managed by Indiana University at (39.2499, -86.4555), there are 3 users streaming through the network. Except two "Purdue" users, "pl2.ucs.indiana.edu" is also streaming videos through the network and has good QoEs during the anomaly period. Though "pl2.ucs.indiana.edu" is not detected to be active when QALA is running, its QoE indicates how the network performs in terms of QoE. For the Purdue networks, they both only have two "PURDUE'" users going through and they obtain the same QoE score.

Table 3: QoE scores of suspect systems for the example QoE anomaly

System	***Name***	***AS #***	***Geo- Location***	***% of anomalous users***	***QoE Score***
Network	Indiana Univ.	19782	(39.2499, -86.4555)	66.66% (2/3)	2.1161
Network	Purdue Univ.	17	(40.4259,-86.9081)	100% (2/2)	0.6742
Device	<PL, Fedora14, Python2.7,EM-DASH>	17	(40.4259,-86.9081)	18.18% (4/22)	3.9587

5.5.4 QoE anomaly identification

Given suspect systems that are likely to incur the anomaly, QRank identifies the QoE anomaly in the anomalous system that 1) has the maximum percentage of related users with QoE

anomalies and 2) has the lowest QoE score. In summary, QRank identifies QoE anomalies in systems following the below two rules.

RULE I: Given a QoE anomaly and suspect systems that may cause the anomaly, a system with larger percentage of related users with QoE anomalies during the anomaly period is more likely to cause the QoE anomaly.

RULE II: Given a QoE anomaly and suspect systems that may cause the anomaly, a system with lower QoE score is more likely to incur the QoE anomaly.

RULE I bases that a network/server/device system is more likely to cause QoE anomalies if there are higher percent of related users having QoE anomalies. For a suspect network system, related users are all users who stream through the network. If the percentage of related users with QoE anomalies is high, it indicates the network has insufficient capacity to provide good QoE for all users streaming through the network. For example, as shown in Table 3, both the network in AS 17 and the network in AS 19782 have two related users with QoE anomalies. However, the network in AS 17 has a higher percentage of related users with QoE anomalies. Considering one user streaming through the network in AS 19782 has good QoE, the network is less likely to have capacity issues. As all users streaming through the network in AS 17 have QoE anomalies, it is more likely that the network in AS 17 has limited capacity. Similar reasoning can be applied to a suspect server. For a suspect device, if the percentage of related users with QoE anomalies is high, it means there is a higher chance to have QoE anomalies when using this type of devices. For example, as shown in Table 3, during the time of the QoE anomaly, there were 22 users using the same type of device (PlanetLab server) installed with the same type of OS (Fedora 14), running the same type of software framework (Python 2.7) and the same emulation code of DASH streaming. There are only 4 of those having QoE anomalies, which indicates the anomaly is not a

reproducible anomaly caused by those device attributes. It is possible that the attributes of device we choose is not complete enough to cover all reproducible anomalies caused by devices. To choose a device attribute, VoD providers can collect various user device attributes and user QoE data in long term to study if one attribute is highly correlated with QoE anomalies.

RULE II identifies anomalous system when two suspect systems have that same percentage of related users with QoE anomalies. When two networks both have insufficient for all their related users and they both have the same percentage of related users with QoE anomalies, the one with lower QoE score has lower average QoEs for all its related users during the anomaly period, which indicates it is more likely to be a capacity bottleneck. To make a hypothetical example in Table 3, we let the user "pl2.ucs.indiana.edu" have one chunk QoE value just below the anomaly threshold, such as $q = 1.9999$, then it would be detected to have QoE anomaly during the example anomaly period. The network in AS 19782 would also have 100% of related users with QoE anomalies. However, if the capacity of network in AS 19782 is higher than the network in AS 17, its QoE score would still be higher than the QoE score of the network in AS 17. We then go through the suspect systems for the example QoE anomaly. As the network managed by Purdue University has the highest percent (100%) of related users with QoE anomalies and the lowest QoE score (0.6741), it is identified as the anomalous system for the example QoE anomaly.

5.6 Evaluation of QRank in Controlled VoD

5.6.1 Experiment Setup

We deploy a controlled VoD system in Azure Cloud to evaluate QRank. Figure 40 shows the network topology of the controlled VoD system. We deploy 3 servers S_1, S_2 and S_3 in two Cloud networks 1 and 2. We emulate 9 users in 3 campus networks A, B and C.

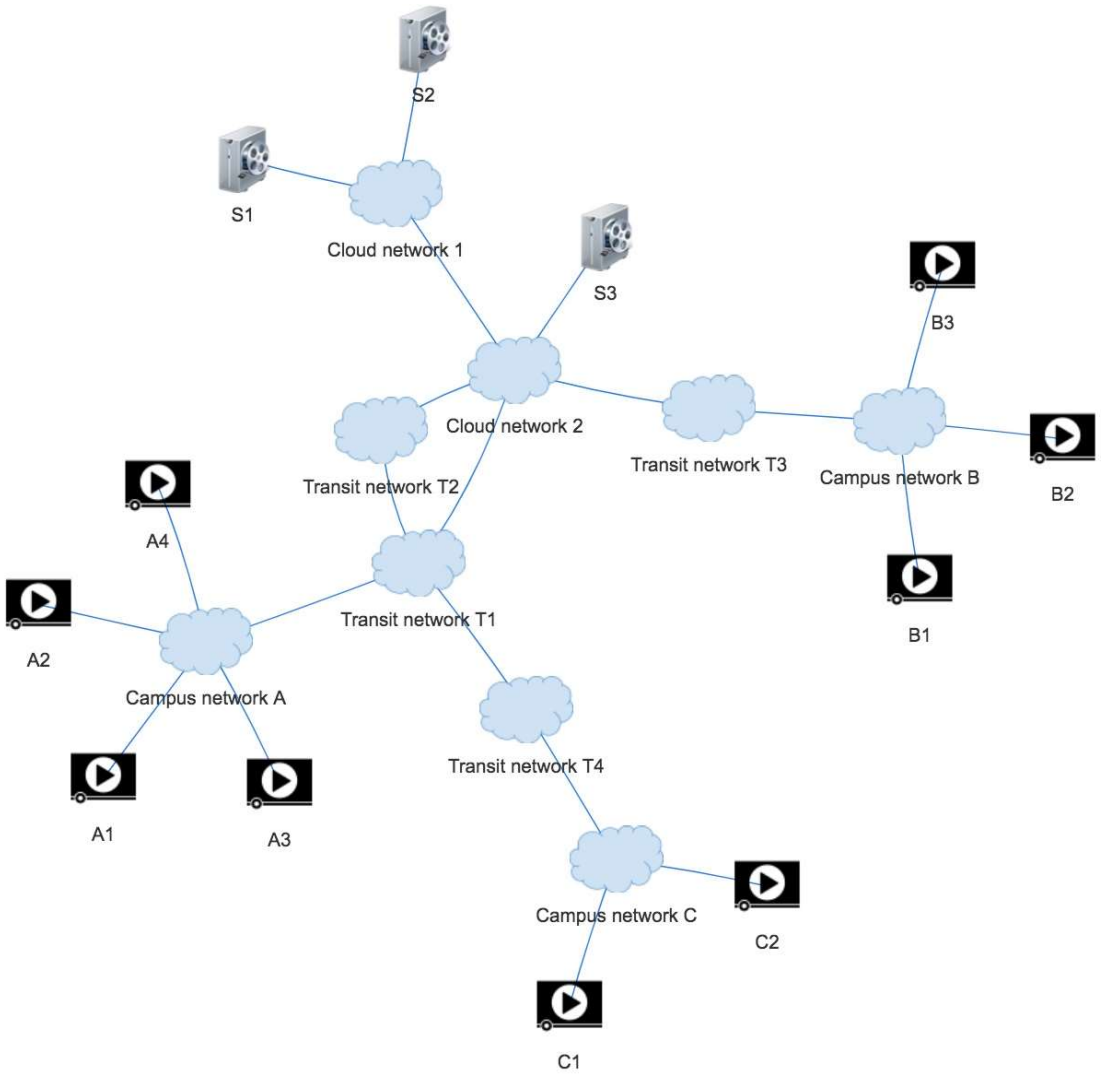


Figure 40: Network Topology of controlled VoD system

Table 4: Video streaming sessions in the controlled VoD

User	*Campus Network*	*Transit Networks*		*Cloud Network*	*Server*
A_1	A	T_1	T_2	1	S_1
A_2					S_2
A_3, A_4				2	S_3
B_1	B	T_3		1	S_1
B_2					S_2
B_3				2	S_3
C_1	C	T_4	T_1	1	S_1
C_2				2	S_3

As Table 4 shows, users A_1, B_1, C_1 stream videos from the server S_1, A_2 and B_2 stream videos from the server S_2. Users A_3, A_4, B_3 and C_2 stream videos from S_3. All users in campus network A connect to the Cloud via transit network T_1 and T_2. All users in campus network B connect to the Cloud via transit network T_3. All users in campus network C connect to the Cloud via transit network T_4 and T_1. We inject QoE anomalies on server S_2, the total capacity of Cloud network 1, the transit network T_1, the campus network B and the user device A_1. For servers and networks, we inject anomalies by limiting their capacity so users stream through the network or from the server will have QoE anomalies. To achieve the purpose of limiting capacity in servers and networks, we use the network emulator "netem tc" [103] to throttle the outbound bandwidth on all server hosts. If the capacity limit is on a server, "netem tc" can be applied directly on server host. If the capacity limit is on a network, we apply "netem tc" rules on all servers with destination ip prefexes denoting users who go through the network. We determine the capacity to throttle according to the number of users streaming through the component. For anomaly to inject on S_2, we limit the capacity on the server S_2 to 1 Mbps. There are 2 users streaming through S_2 so each one on average gets 500 kbps. For anomaly to inject in the Cloud network 1, we limit the capacity of the Cloud network 1 to 2.5Mbps. Specifically, we throttle the outbound capacity on S_1 to 1.5

Mbps for users A_1, B_1, C_1 and we throttle the outbound capacity on S_2 to 1 Mbps for users A_2, B_2. For anomaly to inject in the transit network T_1, we limit the total capacity in T_1 to 3.0 Mbps. There are 6 users streaming through T_1. Specifically, the capacity on S_1 is throttled to 1Mbps for IPs of A_1 and C_1. The capacity on S_2 is throttled to 0.5Mbps for IP of A_2. The capacity on S_3 is throttled to 1.5Mbps for IPs of A_3, A_4, and C_2. For anomaly to inject in the campus network B, we limit the total capacity in B to 1.5 Mbps as there are 3 users in campus network B. Specifically, S_1, S_2 and S_3 throttle their outbound network capacity to 0.5 Mbps for IP prefix of network B. When we inject anomalies on a specific user device, we inject faulty code in our emulated DASH player to add delays for all packets received, denoted as EM-DASH-ERR. All users start streaming videos at the same time for 10 minutes. Anomalies are injected throughout the streaming period.

5.6.2 QoE anomaly injected at server S_2

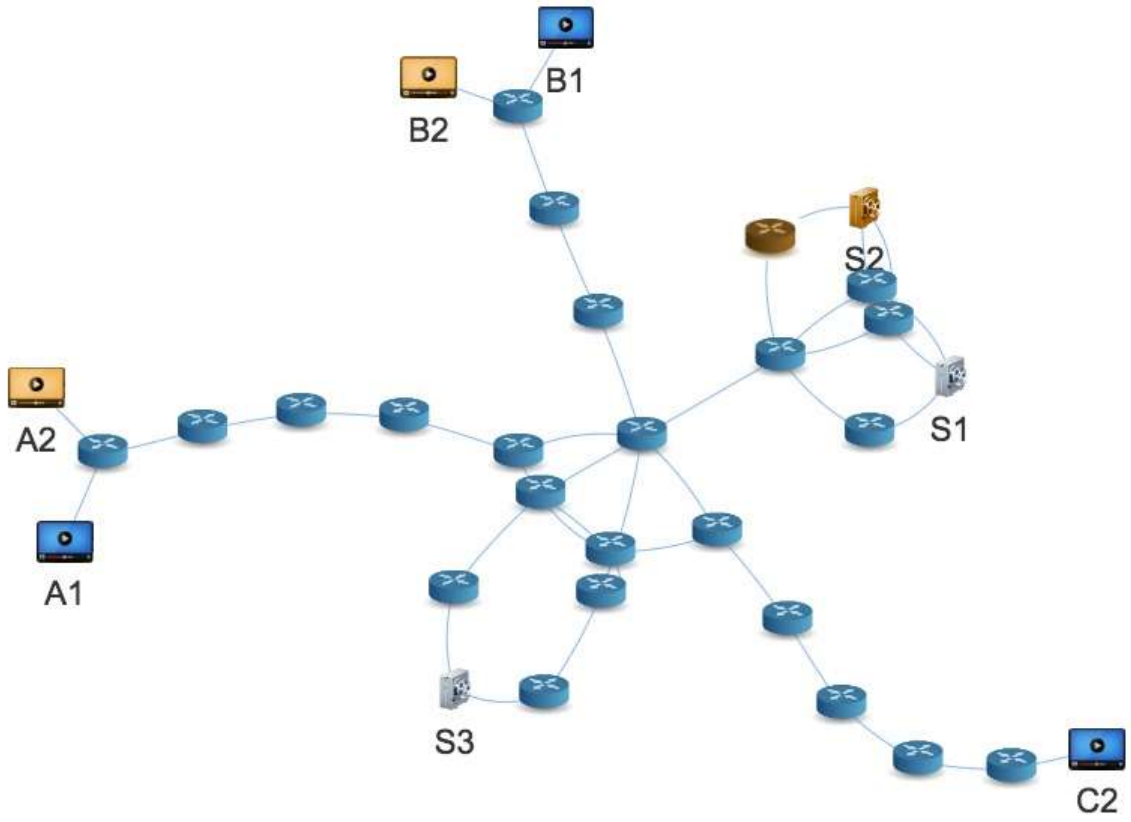


Figure 41: Localization of suspect nodes for QoE anomalies injected at S_2

QRank detects QoE anomalies spanning the whole streaming period on A_2 and B_2. In Figure 41, we show the localization of suspect nodes for the QoE anomaly on B_2. It shows that only the server S_2, a router connecting to S_2 and user devices on A_2 and B_2 are suspect nodes. Then in Table 5, QRank computes the percentage of related users with QoE anomalies for all suspect systems retrieved from suspect nodes. QRank then identifies the server S_2 as the anomalous system for the QoE anomaly. It has the highest percent (100%) of related users with QoE anomalies and the lowest QoE score (1.3049). It shows that the ranking rules further narrow down the number of

suspect systems from 3 to 1, namely successfully identify the anomalous system with injected anomaly.

Table 5: QoE scores for suspect systems of QoE anomaly injected at S_2

System Type	Name	% of related users with QoE anomalies	# of related users with QoE anomalies	QoE Score
Server	S_2	100%	2	1.3049
Network	Cloud Network 2	40%	5	3.4512
Device	<PL, …EM-DASH>	22.22%	9	4.1263

5.6.3 QoE anomaly injected in the Cloud network 1

As expected, it is observed that 5 users (A_1, A_2, B_1, B_2, and C_1) who were streaming from the Cloud network 1 had chunk QoE anomalies while users A_3, A_4, B_3, C_2 had good QoEs all the time. In Figure 42, we show the localization of suspect nodes for QoE anomaly detected on B_1. From suspect nodes colored in brown, QRank then detects the suspect systems and computes their QoE scores as shown in Table 6. QRank identifies the Cloud network 1 as the anomalous system for the QoE anomaly, because it has the highest (100%) percentage of anomalous related users and the lowest QoE score (1.2911).

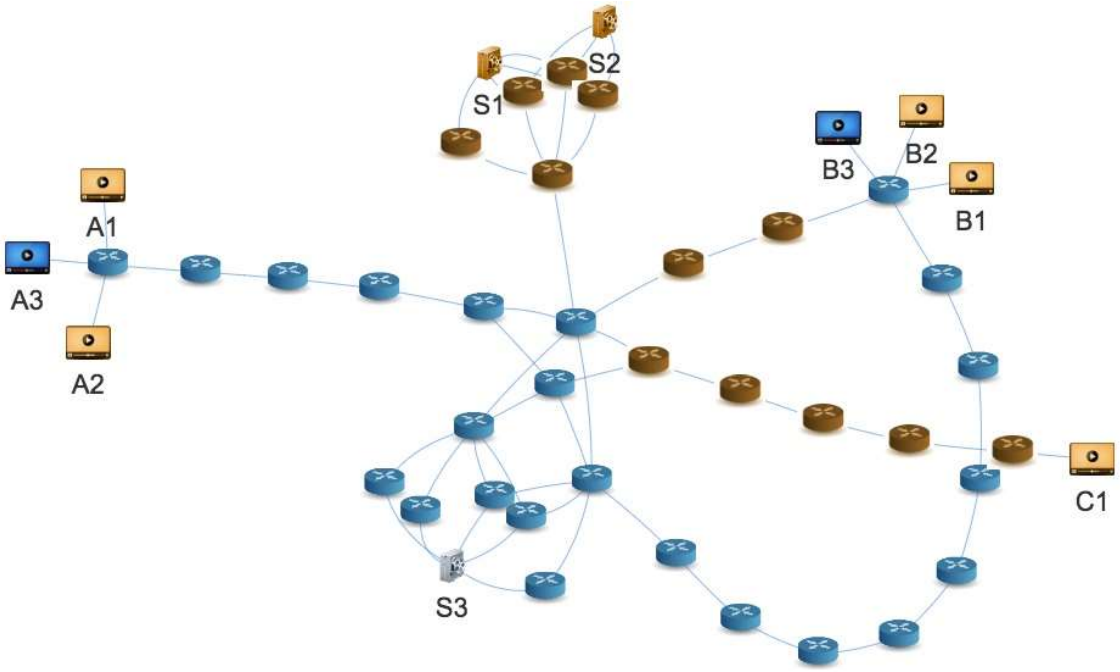


Figure 42: Localization of suspect nodes for QoE anomalies injected in Cloud network 1

Table 6: QoE scores for suspect systems of QoE anomaly injected in the Cloud network 1

System Type	*Name*	*% of related users* with QoE anomalies	*# of related users with QoE anomalies*	*QoE Score*
Network	Cloud Network 1	100%	5	1.2911
Server	S_1	100%	3	1.3090
Network	Transit Network T_3	66.7%	2	2.5638
Network	Campus Network B	66.7%	2	2.5638
Device	<PL, …EM-DASH>	55.56%	5	2.9764

5.6.4 QoE anomaly injected in the transit network T_1

As expected, QoE anomalies are detected on all T_1 related users throughout the experiment. In Figure 43, we show the localization of suspect nodes for QoE anomaly on A_4. It shows that all nodes on user A_4's path to the server S_3 are labeled as suspect nodes except the server S_3 as B_3 was having good QoE. QRank then retrieves all suspect systems for the anomaly in Table 7. By ranking

the percentage of related anomalous users, QRank narrows down the suspect systems to the transit network T_1, the campus network A, and the transit network T_2. Then, via comparing the QoE scores, QRank successfully identifies the transit network T_1 as the anomalous system.

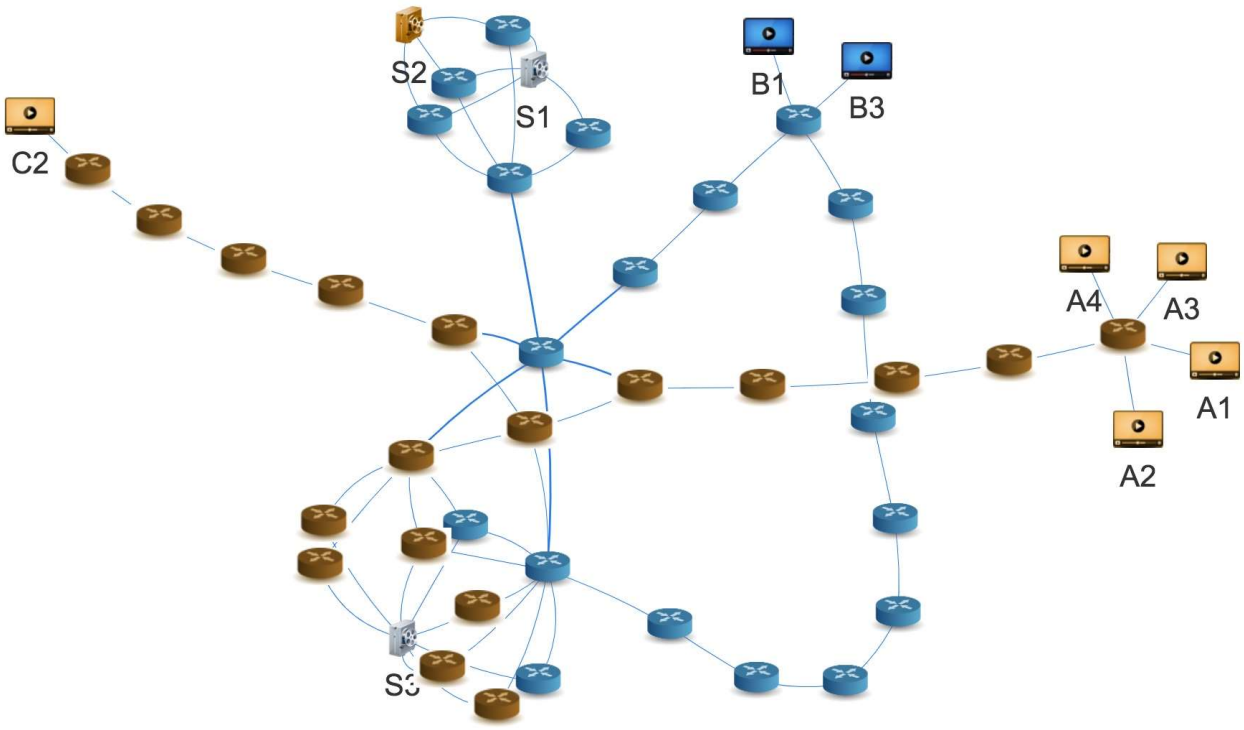


Figure 43: Localization of suspect nodes for QoE anomalies injected in Transit network T_1

Table 7: QoE scores for suspect systems of QoE anomaly injected in the Transit network T_1

System Type	*Name*	*% of related users with QoE anomalies*	*# of related users with QoE anomalies*	*QoE Score*
Network	Transit Network T_1	100%	6	0.4966
Network	Campus Network A	100%	4	0.5493
Network	Transit Network T_2	100%	4	0.5493
Device	<PL, …EM-DASH>	66.67%	6	1.9466
Network	Cloud Network 2	66.67%	6	1.9466

5.6.5 QoE anomaly injected in the campus network B

As expected, all users in campus network B have QoE anomalies. Figure 44 shows the localization of all suspect nodes that are exclusively on anomalous user B_1's path. As annotated, user B_1 was streaming videos from S_1 through routers in the campus network B and the routers in the transit network T_3. The suspect systems retrieved from suspect nodes are the user device on B_1, the campus network B and the transit network T_3, as shown in Table 8. It is shown that the campus network B and the transit network T_3 had the same highest percentage (100%) of users with QoE anomalies and the same lowest QoE score (1.2657). They are both identified as anomalous system causing the QoE anomaly. However, the campus network B is the true anomalous system. In this experiment, QRank is not accurate enough to pinpoint the system with injected anomaly but it successfully narrows down the anomalous systems from 3 suspect systems to 2. If there were some users in other access networks streaming through the transit network T_3, they would not have QoE anomalies as the anomaly is injected in campus network B. Then, the percentage of related anomalous users in T_3 would be less than 100% and the original QoE score on T_3 would be higher. This experiment shows that the accuracy of QRank depends on the number of users related to a system. If more related users are available for a suspect system, the more accurate QoE score can be learnt, which in turn improves anomaly identification accuracy.

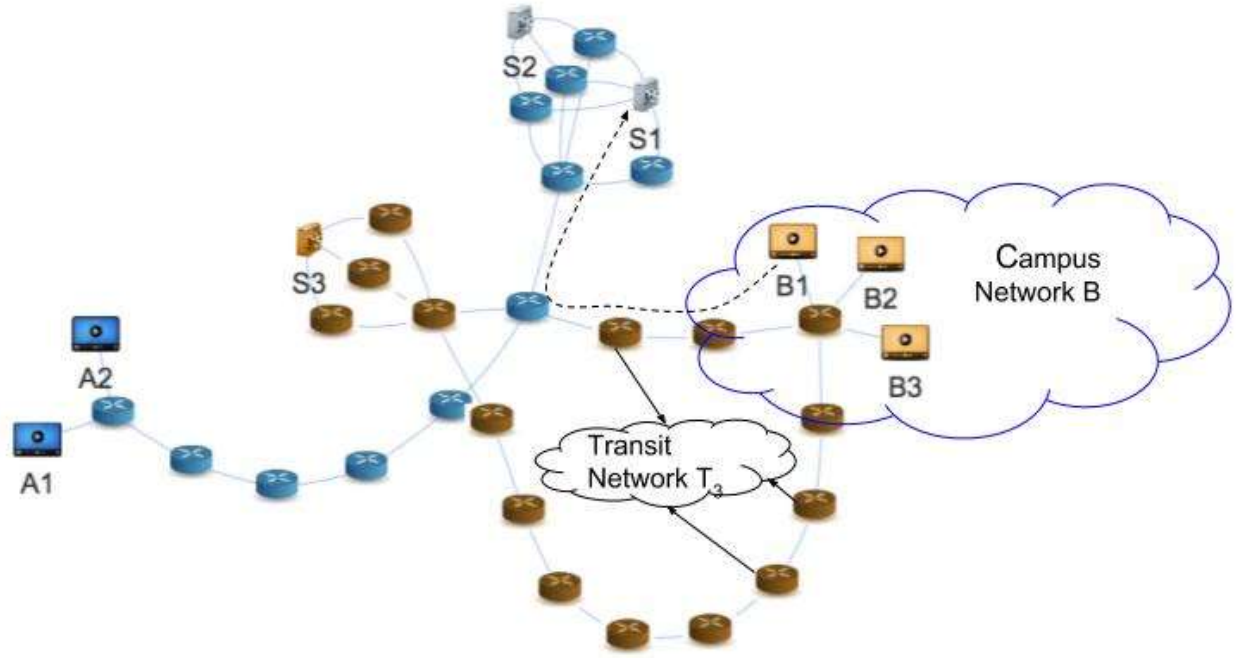


Figure 44: Localization of suspect nodes for QoE anomalies injected in campus network B

Table 8: QoE scores for suspect systems of QoE anomaly injected in the campus network B

System Type	*Name*	*% of related users with QoE anomalies*	*# related users with QoE anomalies*	*QoE Score*
Network	Campus Network B	100%	3	1.2657
Network	Transit Network T_3	100%	3	1.2657
Device	<PL, … EM-DASH>	33.33%	3	3.7230

5.6.6 QoE anomaly injected in a type of devices

In the previous experiments, all users ran our emulated DASH players (EM-DASH) written in Python 2.7 on PlanetLab (PL) servers installed with Fedora 14, thus all users' devices belonged to the same category, denoted as <PL, Fedora 14, Python 2.7, EM-DASH>. In order to test if QRank is able to identify anomalies caused by software issues on user devices. We inject faults in our emulated video player to add delay for all packets it receives, denoted as EM-DASH-ERR. We

let user A_1 and C_2 use EM-DASH-ERR and all others use EM-DASH for video streaming in this experiment. As expected, A_1 and C_2 have QoE anomalies while others have good QoE. Figure 45 shows the localization of suspect nodes for QoE anomaly on C_2. From the ranking of possible anomaly origins shown in Table 9, we notice that though the Cloud network 1 is detected as a suspect system as there are routers in the network exclusively on user C_2's streaming path, the Cloud network 1 has higher QoE score compared to the type of device C_2 is using. It shows that QRank successfully identifies QoE anomalies on user devices if the anomalies is a reproducible software issues.

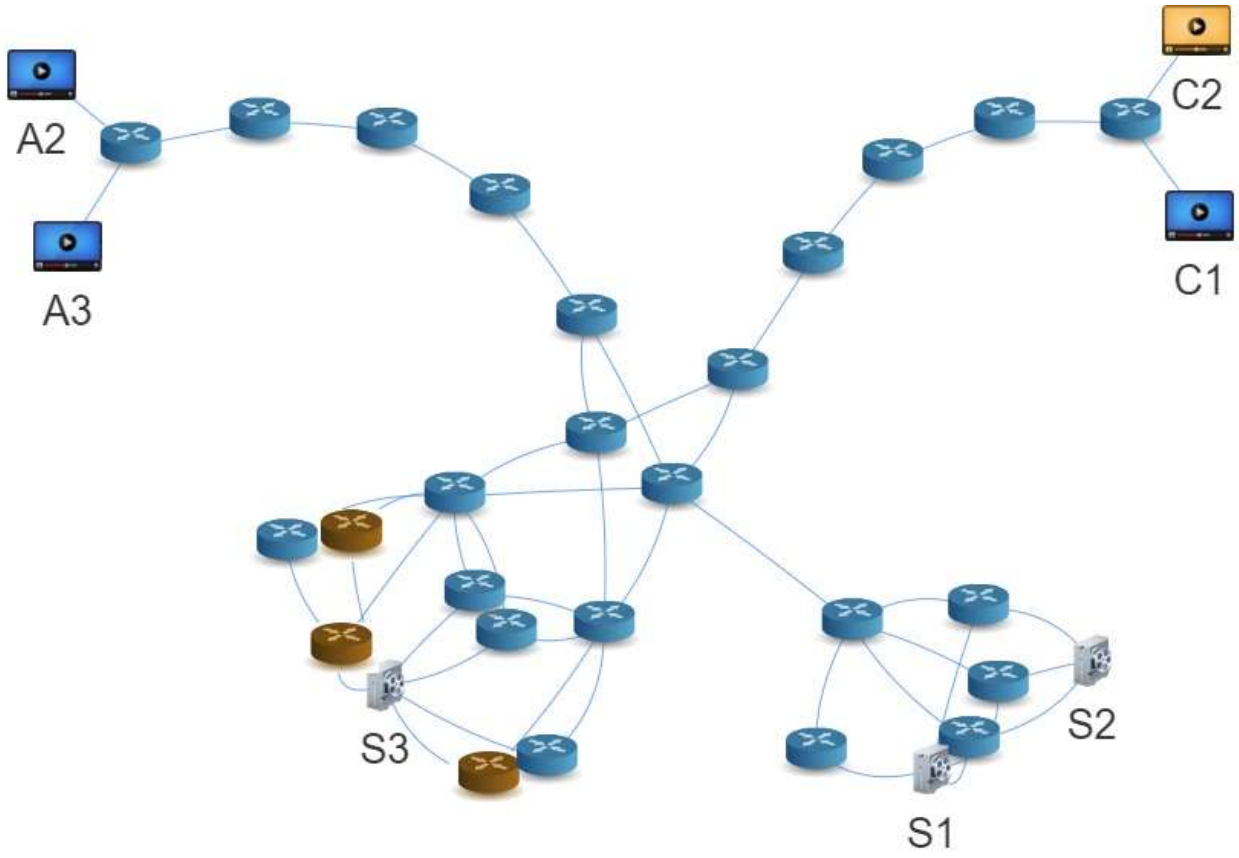


Figure 45: Localization of suspect nodes for QoE anomalies injected on devices running EM-DASH-ERR video player

Table 9: QoE scores for suspect systems of QoE anomaly injected on devices running EM-DASH-ERR video player

Origin Type	*Name*	*% of related users with QoE anomalies*	*# of related users with QoE anomalies*	*QoE Score*
Device	<PL, Fedora 14, Python 2.7, EM-DASH-ERR>	100%	2	1.2657
Network	Cloud Network 2	33.33%	2	3.8378

5.7 Evaluation of QRank for VoD in Azure Cloud

We run QRank in production Azure Cloud and find the access network, the transit network and user devices are major anomalous systems causing QoE anomalies in production Cloud environment. The Azure Cloud itself does not incur any QoE anomalies during 2 days. We deploy a VoD website in Microsoft Azure with video content cached in Azure CDN. We emulated 100 users worldwide in PlanetLab and 24 users in two types of VMs in all regions of Azure Cloud to stream videos. Two types of Azure VMs are the A0 and A2 instances installed with Ubuntu 16.04 and all Azure emulated users run EM-DASH player in Python 2.7. We deploy QRank cloud agents in 5 regions (east US, central US, west US, west Europe and Japan west) of Azure Cloud. The cloud agents collect QoE measurement for both Planetlab and Azure users in each region. All users request 55 minute-long videos at the beginning of every hourto run DASH streaming continuously for 48 hours. The chunk size is 5 seconds for all videos. QRank cloud agents collect QoE measurement every 12 chunks every minute if there is no freezing.

5.7.1 QoE anomalies in production Cloud

QRank detects 9,367 QoE anomalies in total. 65 users among 124 emulated users experienced QoE anomalies. We observe that 61.98% (5806 in 9367) QoE anomalies identify networks as the anomalous systems. Among those anomalies, 94.64% (5495 in 5806) QoE anomalies identify access networks as anomalous systems; 22.39% (1300 in 5806) QoE anomalies identify transit networks as anomaly systems; only one QoE anomaly lasting one chunk period was identified in Cloud network. As 97.23% QoE anomalies (5343 in 5495) incurred by access networks and 95.38% QoE anomalies (3408 in 3573) incurred by devices were experienced by PlanetLab users, it is reasonable to infer that the capacity of PlanetLab servers were limited intentionally. In production environment with real users, transit networks would be the major causes of QoE anomalies. For all QoE anomalies caused by devices, we notice that 95.38\% were caused by Palnetlab nodes with average duration of 89.97 seconds, 2.23% were caused by Azure A0 instances with mean duration of 46.212 seconds and only 2.37% were caused by Azure A2 instances with mean duration of 12.49 seconds. It can be inferred that a device used in multi-tenant environment can caused QoE anomalies more likely. The anomalies also last longer on devices with low hardware resource configurations.

5.7.2 Accuracy of QRank

QWatch only locates anomalies in suspect nodes. QWatch assumes these nodes are likely to cause anomalies. QRank system further retrieves suspect systems from suspect nodes and identifies anomalous system by ranking the percent of anomalous users and their QoE scores. The ranking algorithm can further narrow down the range of suspect systems and identify the true anomaly system. Figure Figure 46 compares the cumulative distribution of the number of suspect

systems obtained by QWatch only and the number of anomalous system obtained by QRank for all QoE anomalies detected. It shows that the ranking algorithm successfully identified the unique anomalous systems for 74.26% of QoE anomalies.

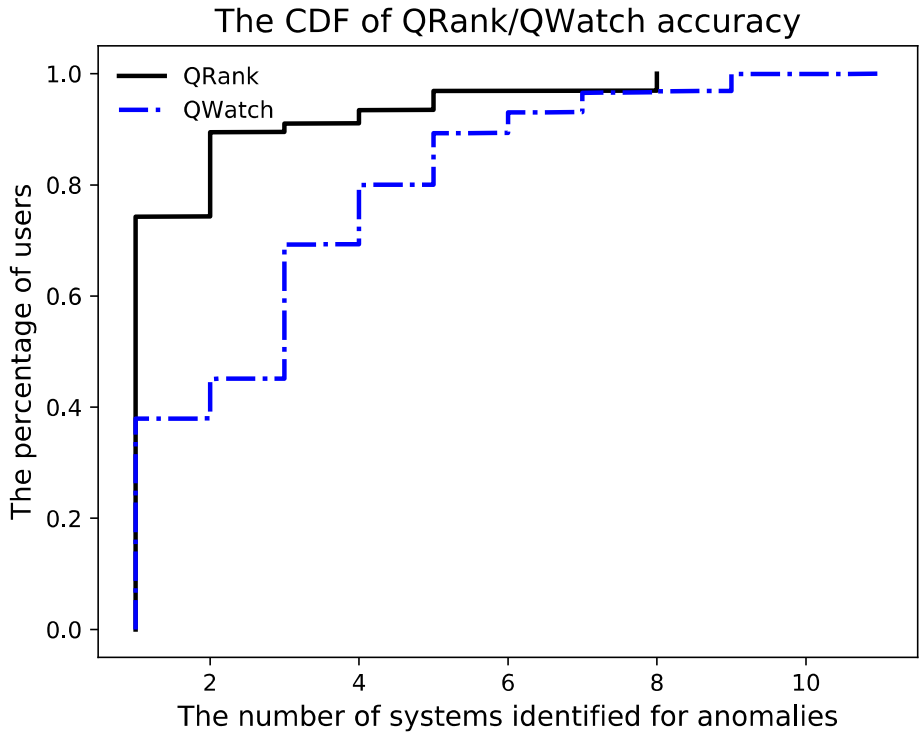


Figure 46: Comparison of accuracy of QWatch vs QRank

5.8 Summary

By injecting anomalies in different systems in a controlled VoD system, we verify that QRank can correctly identify the anomalous system causing the QoE anomaly. We test QRank in a VoD deployed in Azure Cloud with users emulated in Planetlab and Azure Cloud around the world for 2 days. The practice of QRank shows that QoE data is an effective measurement to evaluate how different systems in VoD perform in run time. With enough users sharing their QoE

dada and route information, QRank can identify the anomalous systems for QoE anomalies without detailed monitoring in involved systems.

6. INSIGHTS FROM PERSISTENT AND RECURRENT QOE ANOMALIES FOR DASH STREAMING IN THE CLOUD

6.1 Introduction

In this chapter, we analyze the QoE anomalies for DASH streaming from a production Cloud CDN. We run a QoE anomaly identification system, QRank. QRank uses real-time QoE measurements to identify the anomalous systems. QRank assumes that the system with users who experience lower QoEs is more likely cause QoE anomalies. QRank identifies anomalous system by ranking the QoEs in all systems in the video streaming. We consider Cloud CDN servers, Cloud CDN networks, transit networks, access networks and different types of user devices. We deploy 124 users worldwide in PlanetLab and Azure Cloud to run DASH video streaming sessions for 100 hours. QoE measurements from 124x100=12400 video sessions are collected. 9440 QoE anomalies with average length of 127.48 seconds are detected on 65 emulated users.

Our extensive experiments in production Cloud find the following insights.

- Users experience QoE anomalies very differently. a) Recurrency: 12.1% users experience QoE anomalies recurrently and experience 87.83% of QoE anomalies. b) Persistency: 8.87% of users experience persistent QoE anomalies with duration over 15 minutes. c) Sparcity: During 100 hours' video streaming, 41.1% of users experience only less than one QoE anomaly per hour and all QoE anomalies last less than 900 seconds. 47.58% of users do not experience QoE anomalies at all.
- According to QRank, access networks, transit networks and user devices incur more than 99.98% of QoE anomalies. Among those, 58.66% of QoE anomalies are identified in access networks, 38.14% in user devices, and 13.89% in transit networks.

- 95.38% of QoE anomalies in user devices and 97.23% in access networks are experienced by PlanetLab users. These users are emulated on 48 PlanetLab nodes that belong to 21 campus networks. We infer that PlanetLab nodes in those campus networks are capacity limited, causing QoE anomalies.
- QoE anomalies incurred in access networks and devices are due to PlanetLab conditions. We infer that transit networks could be the major cause of QoE anomalies for real world video application in the Cloud. For all QoE anomalies identified in transit networks, more than 95% of QoE anomalies are identified in only 10 transit ISPs.

These results have an important implication. In order to provide good user QoE, the Cloud provider should identify transit networks that may become bottlenecks for high quality video streaming and appropriate peering with Internet Service Providers (ISPs) to bypass these bottlenecks.

6.2 Descriptive statistics of QoE anomalies

6.2.1 Prevalence of QoE anomalies among users

During 100 hours of video streaming from 124 emulated users in PlanetLab and Azure, QRank detects totally 9440 QoE anomalies. 65 users out of 124 experience QoE anomalies. We count the number of QoE anomalies per user. Results show that a small number of users experience a huge number of QoE anomalies while most users have no QoE anomalies or very small number of QoE anomalies in two days. The top 10 users account for 8049 QoE anomalies in total of 9440 QoE anomalies (85.26%). Most QoE anomalies are severe (4485 out of 9440) and medium (4861 out of 9440) anomalies. Among all QoE anomalies, only less than 1% (94 out of 9440) are light QoE anomalies. More than 99% of QoE anomalies are severe and medium QoE anomalies. It

indicates that during anomaly period, users have more than 20% chunks with QoE values less than q_0 . They probably stream videos at two lowest bit-rate all the time.

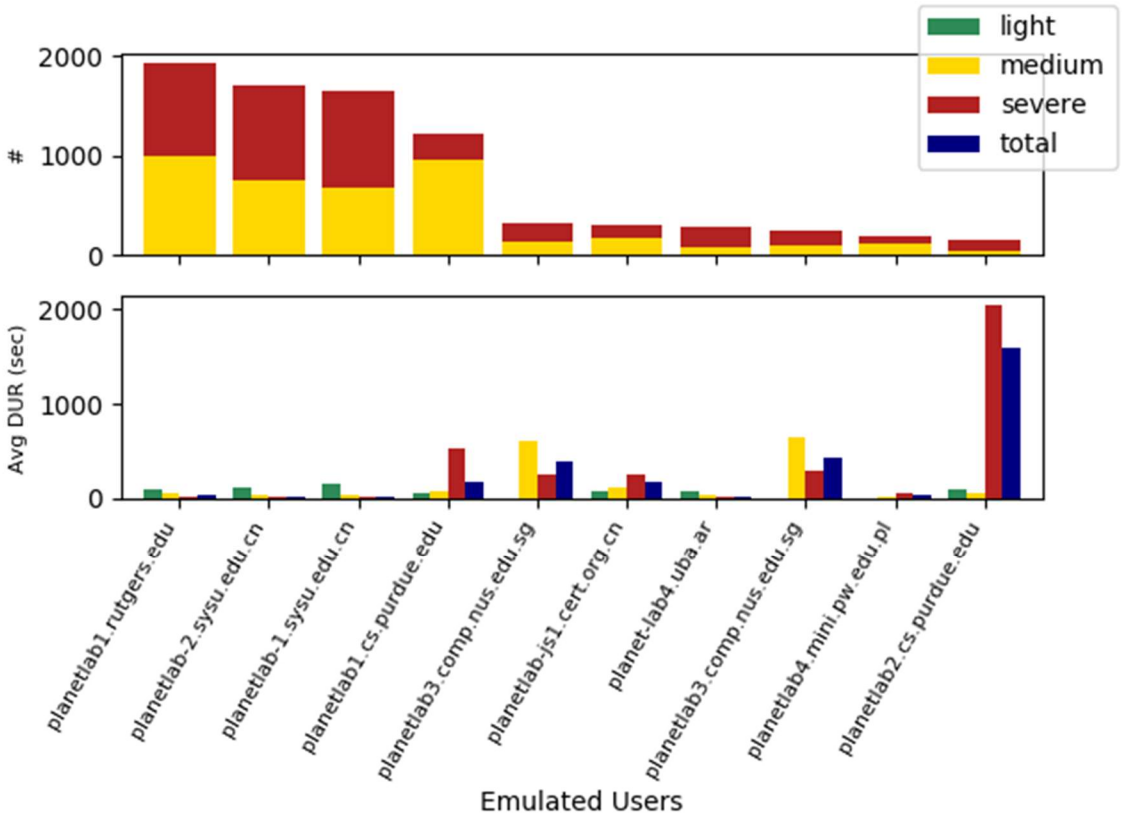


Figure 47: The count and the average duration of QoE anomalies per user (Top 10 shown)

We notice that some users experience QoE anomalies with an average anomaly period longer than 30 minutes. We denote QoE anomalies lasting longer than 900 seconds (i.e. 15 minutes) as persistent QoE anomalies. QRank detects that there are 11 users experiencing 332 persistent QoE anomalies. The top 6 users experience more than 97% (323 in 332) persistent QoE anomalies. All of them are PlanetLab users.

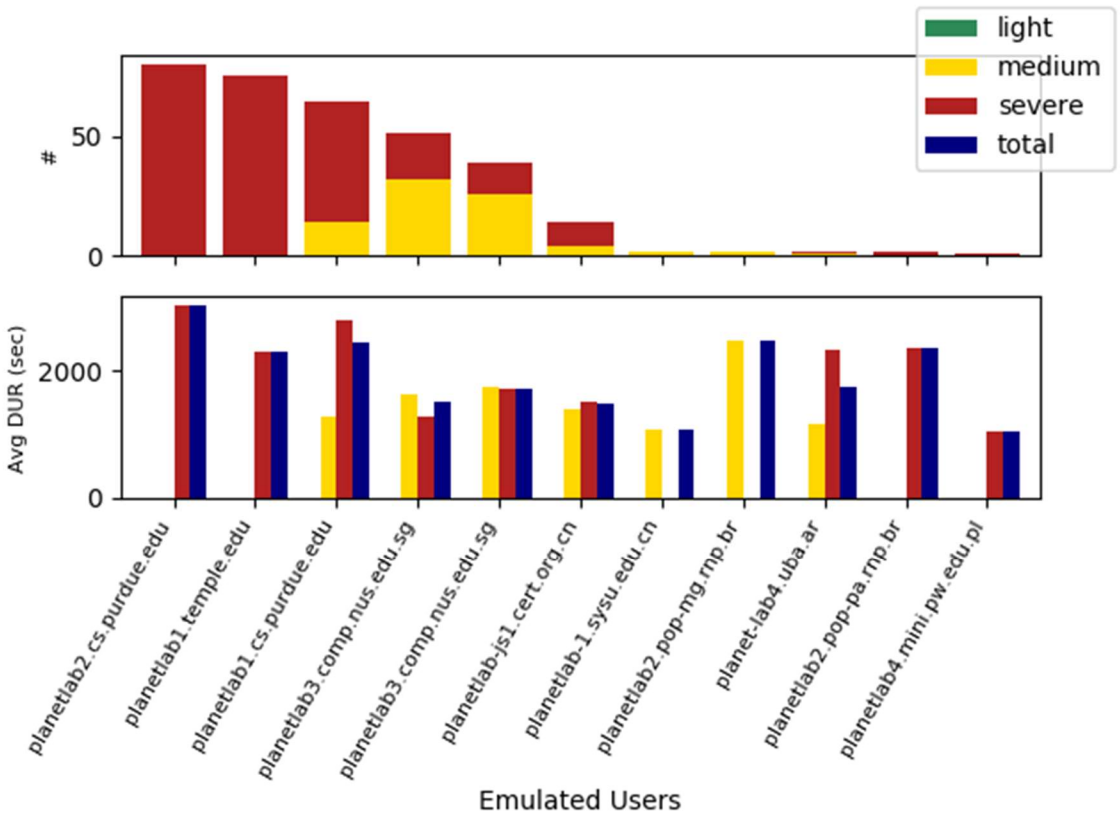


Figure 48: The count and the average duration of persistent QoE anomalies per user

We list the number and the average duration of all persistent QoE anomalies they experience in Figure 48. Almost all persistent QoE anomalies are severe and medium QoE anomalies. There are users experiencing short QoE anomalies that occur frequently. These QoE anomalies last less than 15 minutes but occur recurrently, on average occurring more than once per hour. We denote these QoE anomalies as recurrent QoE anomalies. All users with recurrent QoE anomalies are shown in Figure 49. Among 65 users with QoE anomalies, there are 14 users experiencing recurrent QoE anomalies. The top 4 users with most recurrent QoE anomalies experience 77.8% (6453 out of 8292) of recurrent QoE anomalies.

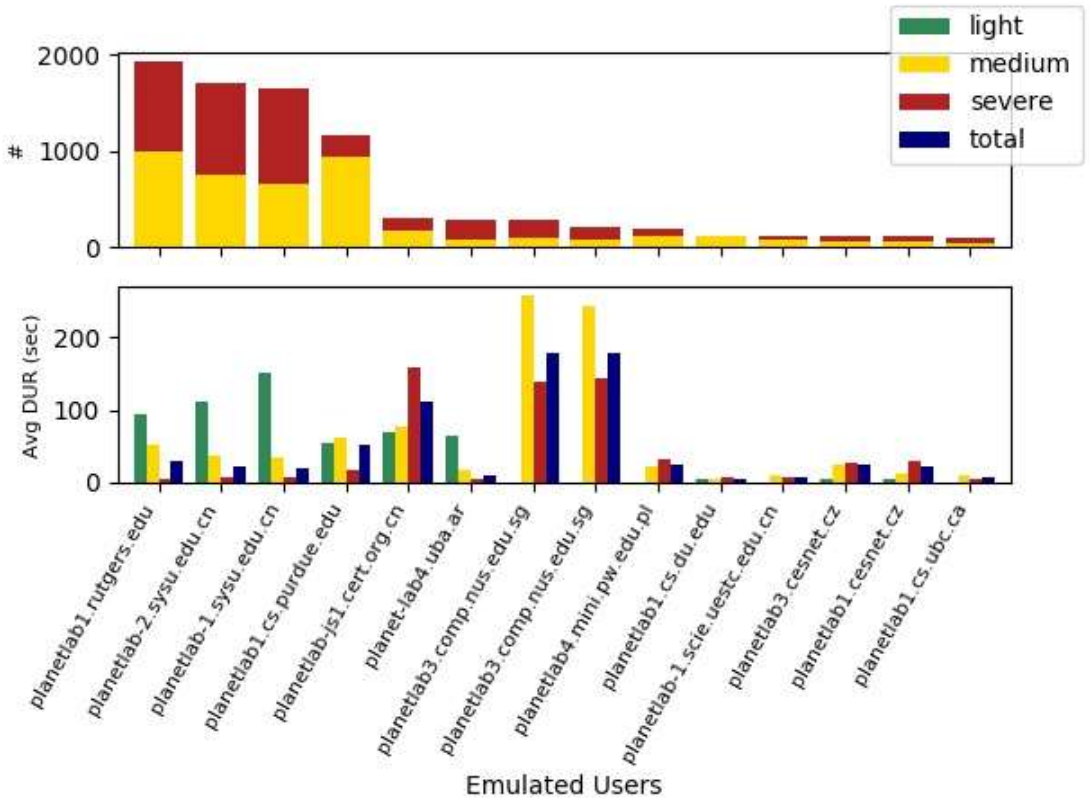


Figure 49: The count and the average duration of recurrent QoE anomalies per user

All other QoE anomalies are occasional QoE anomalies. Occasional QoE anomalies last less than 900 seconds and on average occur less than once per hour. Figure 50 shows that occasional QoE anomalies among users follows a long-tail distribution. 47 users only have occasional QoE anomalies and 4 users experience both occasional and persistent QoE anomalies. Among all 9440 QoE anomalies, there are 332 persistent QoE anomalies, 8292 recurrent QoE anomalies and 812 occasional QoE anomalies. Around 91.4% of QoE anomalies are persistent and recurrent QoE anomalies. These anomalies occur only on 19 users. (15 users with recurrent QoE anomalies plus 11 users with persistent QoE anomalies minus 7 users with both anomalies as shown in Table 10.) From above results, we show that persistent and recurrent QoE anomalies are

not prevalent and only occur on a small number of users. Occasional QoE anomalies are prevalent among users and its distribution over users seems like a long-tail distribution. We later show that these occasional QoE anomalies are mostly caused by occasional traffic congestion in different networks.

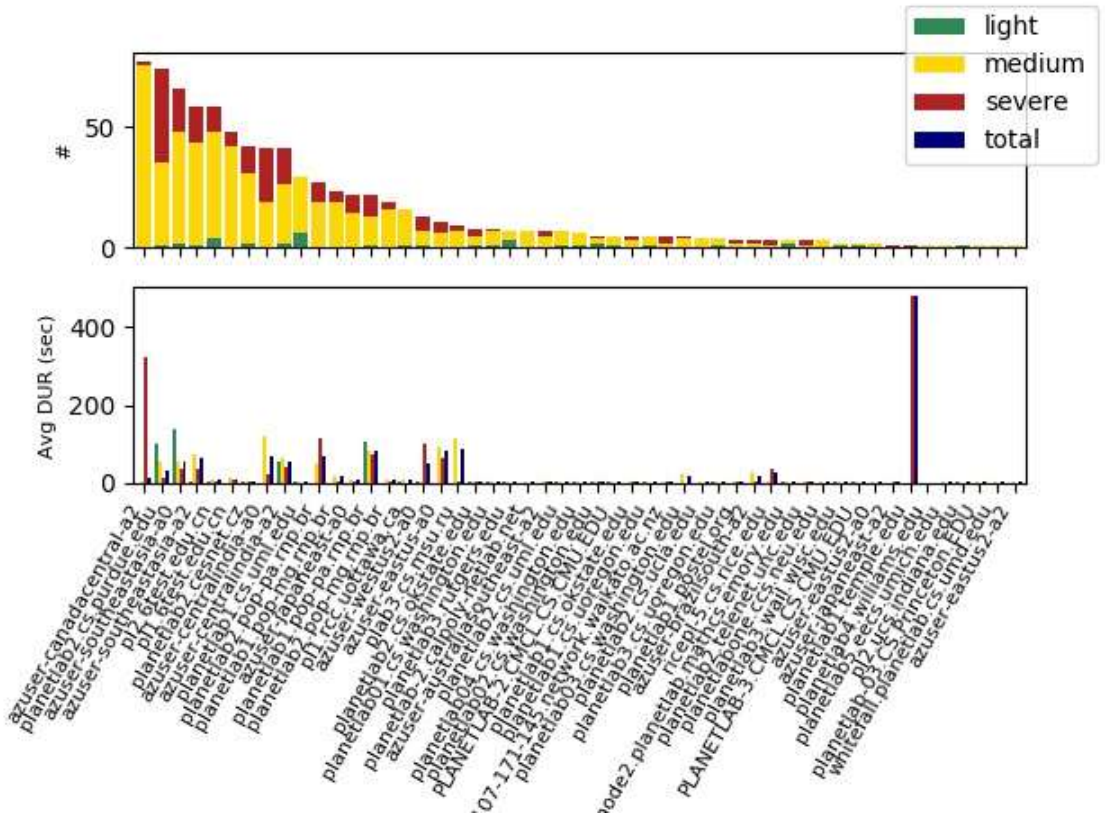


Figure 50: The count and the average duration of occasional QoE anomalies per user

Table 10: Number of users with different QoE anomalies

Types of QoE anomalies on emulated users	# of users	Emulated User Examples
Total # of emulated users	124	
Emulated users got QoE anomalies	65	
Emulated users get ***occasional*** QoE anomalies	51	azuser-canadacentral-a2, planetlab2.cs.okstate.edu

Emulated users get ***recurrent*** QoE anomalies	15	planetlab1.rutgers.edu, planetlab-2.sysu.edu.cn
Emulated users get ***persistent*** QoE anomalies	10	planetlab2.cs.purdue.edu, planetlab1.temple.edu

6.2.2 Types of anomalous systems

Table 11: QoE anomaly statistics in each anomalous system type

Types of Anomalous systems	**Networks**				**Client**	**Server**
Anomalous sytems	**Cloud Network**	**Transit Network**	**Access Network**	**All Networks**	**User devices/Home network**	**Servers**
# of QoE anomalies	1	1373	5568	5887	3573	0
Mean Anomaly Duration	5.0	185.992	139.22	133.97 secs	87.153	N/A

QRank identifies QoE anomalies in user devices/home networks, access networks, transit networks, cloud networks and CDN servers. When a QoE anomaly is detected on a video session, all systems in the video streaming are analyzed. QRank identify the system with the lowest average QoE value during the anomaly period. We count the number and the average duration of QoE anomalies identified in different types of anomalous systems in Table 11. We observe that 62.36% (5887 out of 9440) QoE anomalies identify network systems as the anomalous systems. Among those anomalies, 94.58% (5568 out of 5887) QoE anomalies identify access networks as their anomalous systems. 23.32% (1373 out of 5568) QoE anomalies identify transit networks as anomalous systems. Only 1 QoE anomaly identify Cloud networks as anomalous systems. QoE anomalies identified in access and transit networks usually last long time on average. QoE anomalies identified in access networks last 139.22 seconds on average. Anomalies identified in transit networks on average last 185.992 seconds. Client devices/home networks are identified as

anomalous systems for 3573 QoE anomalies (37.85%) and these anomalies on average last around 87.153 seconds. We now describe in detail anomalous systems in each category.

6.2.3 QoE anomalies identified in access networks

In Figure 51, we count the number and the average duration of QoE anomalies over access networks. The top three access networks totally incur 78.8% (4392 out of 5568) of all QoE anomalies in access networks. The top three access networks are AS4538 with Name "China Education and Research Network Center", AS17 with name "Purdue University" and AS4134 with name "No.31,Jin-rong Street". We study users connecting through these networks and we find that AS4538 is the access network of planetlab-1.sysu.edu.cn and planetlab-2.sysu.edu.cn. AS17 is the access network for planetlab1.cs.purdue.edu, planetlab2.cs.purdue.edu. AS4134 is the access network for user "planetlab-js1.cert.org.cn". These users experience a large number of QoE anomalies as shown in Figure 47.

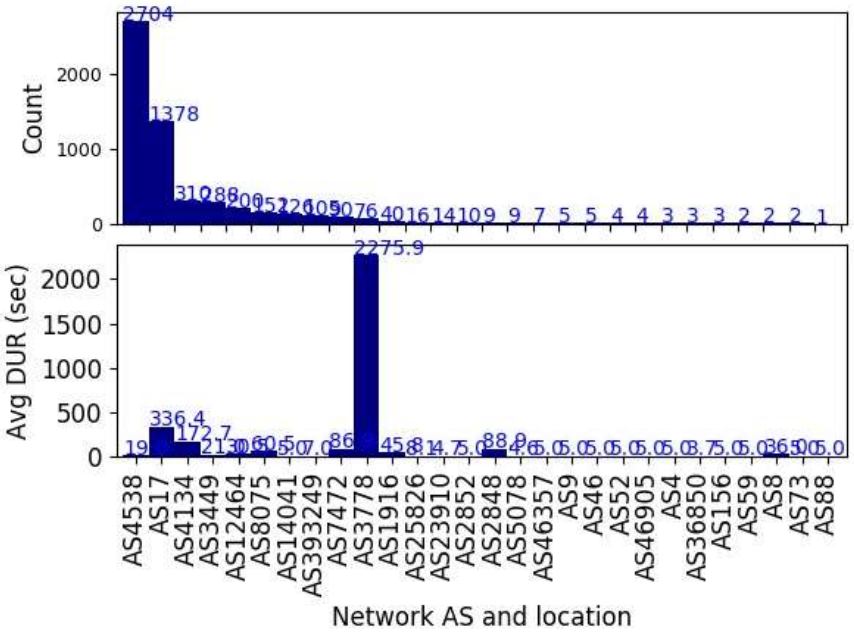


Figure 51: The count and the average duration of QoE anomalies in access networks

As shown in Figure 52, AS17 and AS4134 also incur persistent QoE anomalies. There is also another access networks, AS 3778 with ISP name “Temple University”, incurring persistent QoE anomalies that on average last more than 38 minutes (2299.9 seconds). Among 238 persistent QoE anomalies identified in access networks, these three access networks incur 233 persistent QoE anomalies.

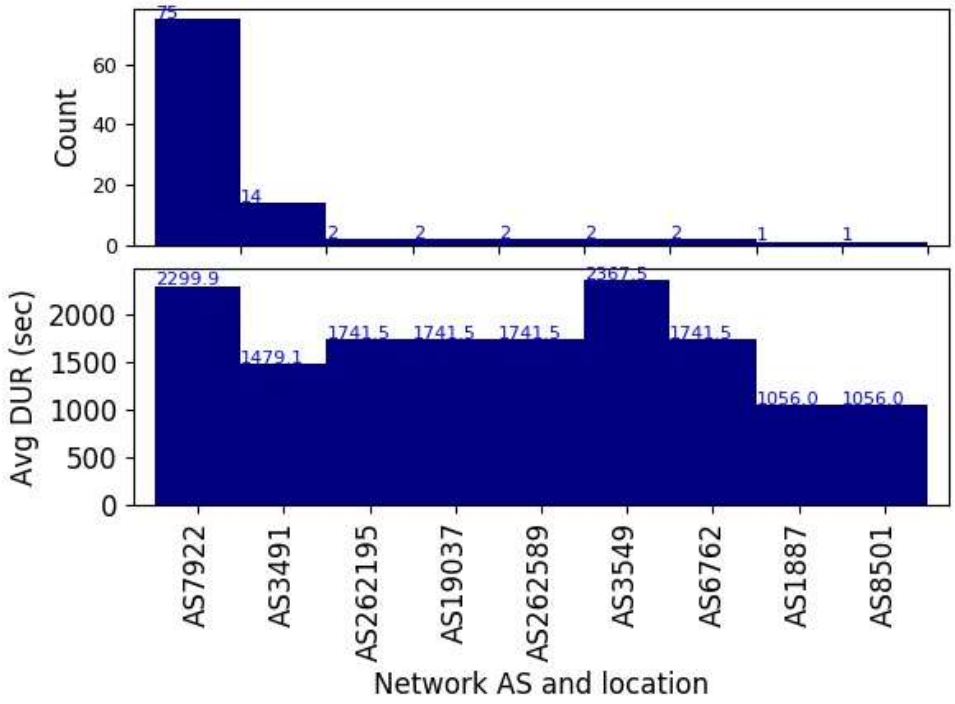


Figure 52: The count and the average duration of persistent QoE anomalies in access networks

As shown in Figure 53, most recurrent QoE anomalies occur in a small number of access networks. The top three access networks totally incur 83.36% (4067 out of) of QoE anomalies in access networks. The top six access networks incur total 95.88% (4678 out of 4879) recurrent QoE anomalies identified in access networks. The recurrent QoE anomalies they incur last from 9 seconds to 2 minutes (110.9 seconds) on average. There are also anomalous networks that only

cause QoE anomalies occasionally, such as AS46357 with ISP name of California Polytechnic State University and AS 88 with ISP name of Princeton University.

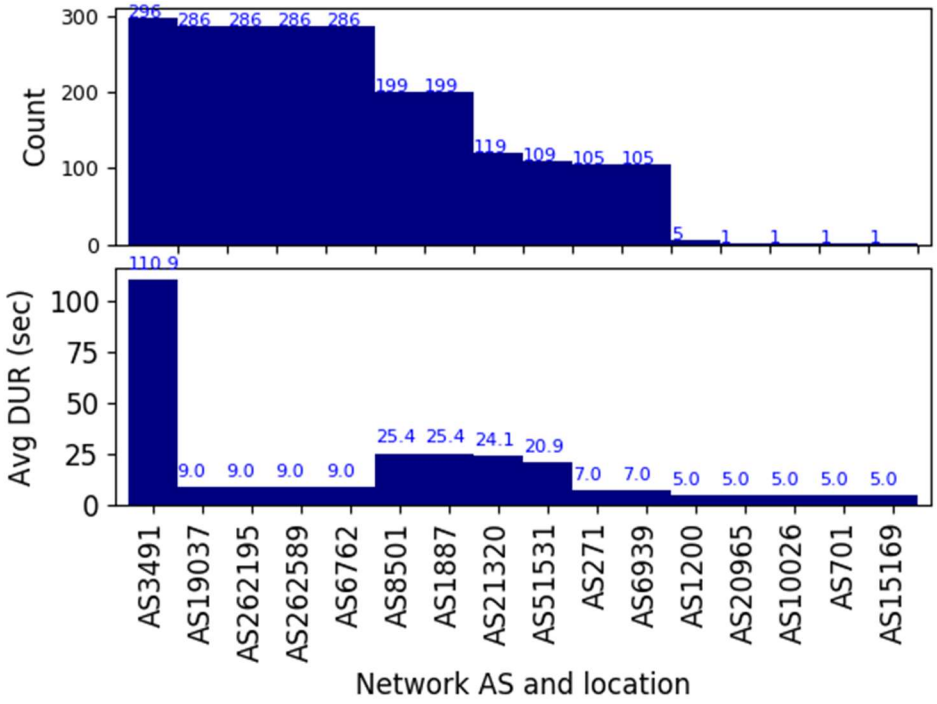


Figure 53: The count and the average duration of recurrent QoE anomalies in access networks

6.2.4 QoE anomalies identified in transit networks

In Figure 54, we count the number of QoE anomalies that are identified in transit networks. There are totally 38 anomalous transit networks and these networks totally cause 1373 QoE anomalies. AS262589 with ISP name "INTERNEXA Brasil Operadora de Telecomunicacoes S.A" is identified to cause the largest number of QoE anomalies. The distribution of QoE anomalies among transit networks has a long tail. The top 10 transit networks totally incur more than 82.45% (1132 out of 1373) of all QoE anomalies identified in transit networks.

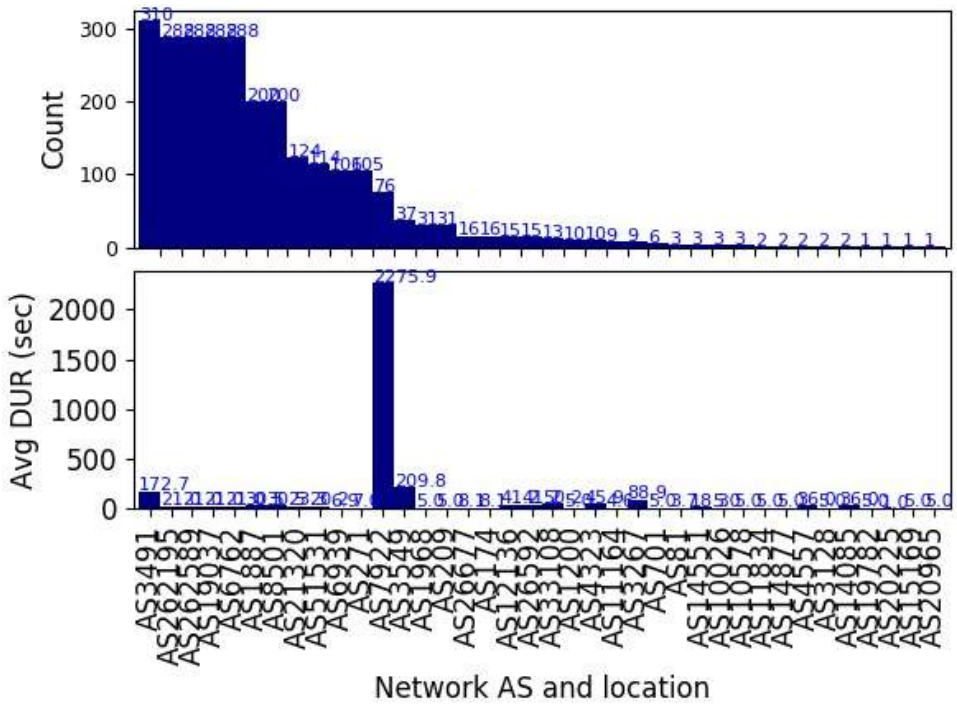


Figure 54: The count and the average duration of QoE anomalies in transit networks

From Figure 55, we notice that AS 7922 (Comcast Cable Communications, LLC) and AS 3491 (PCCW Global, Inc.) cause 94.68% of persistent QoE anomalies (89 out of 94).

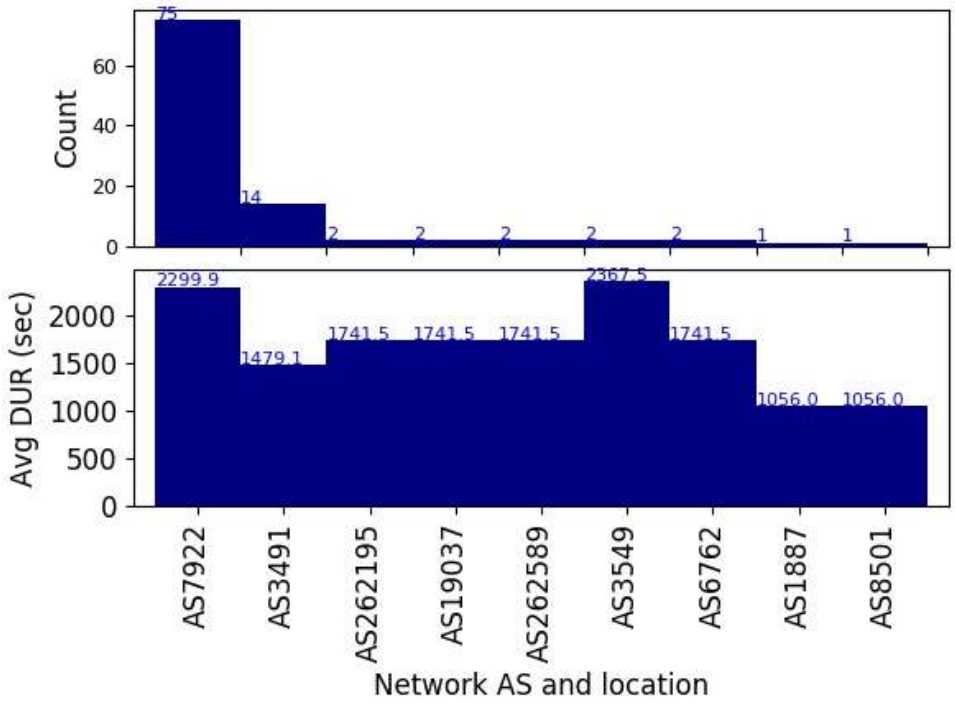


Figure 55: The count and the average duration of persistent QoE anomalies in transit networks

In addition to AS 3491, there are four other transit ISPs that incur many recurrent QoE anomalies. As shown in Figure 56, they are AS19037 (AMX Argentina S.A.), AS262195 (Transamerican Telecomunication S.A.), AS262589 (INTERNEXA Brasil Operadora de Telecomunicacoes S.A), and AS6762 (TELECOM ITALIA SPARKLE S.p.A). AS3491 incurs recurrent QoE anomalies lasting longer than 100 seconds on average. The other three transit networks incur recurrent QoE anomalies lasting less than 10 seconds on average.

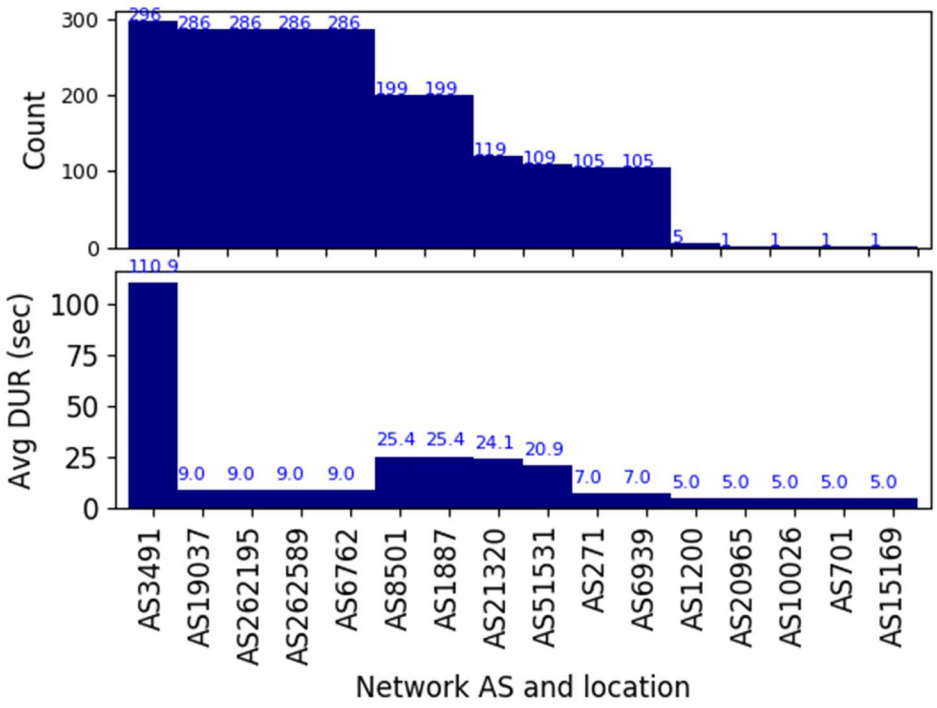


Figure 56: The count and the average duration of recurrent QoE anomalies in transit networks

6.2.5 QoE anomalies identified in devices

Around 37.85% (3573 out of 9440) QoE anomalies identify devices as anomalous systems. In Figure 57, we show the count and the average duration of all QoE anomalies identified in different types of devices. In our experiments, we use PlanetLab nodes to emulate users so the users' devices are PlanetLab servers.

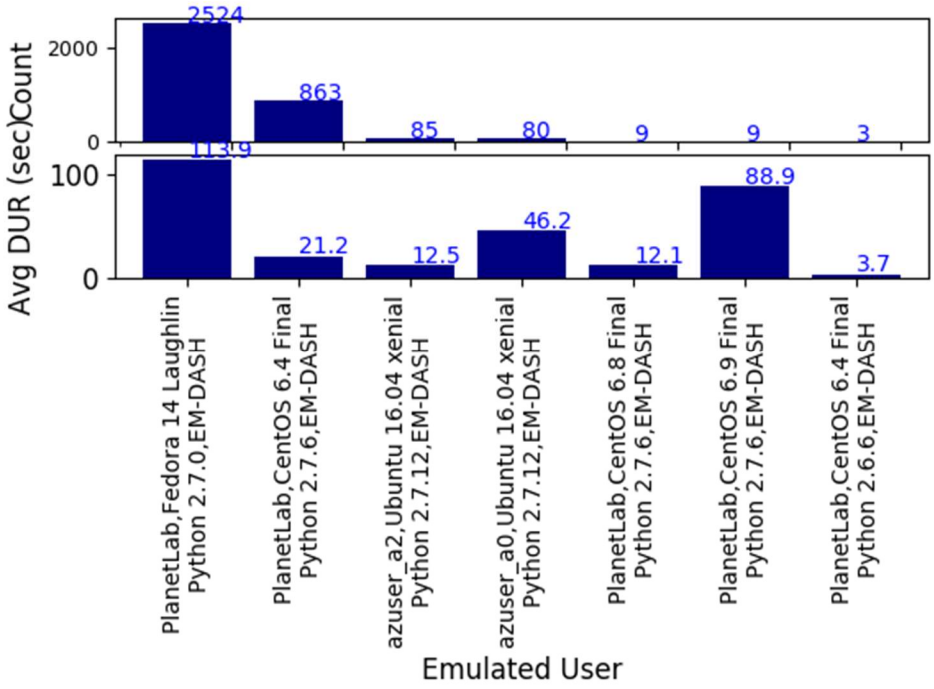


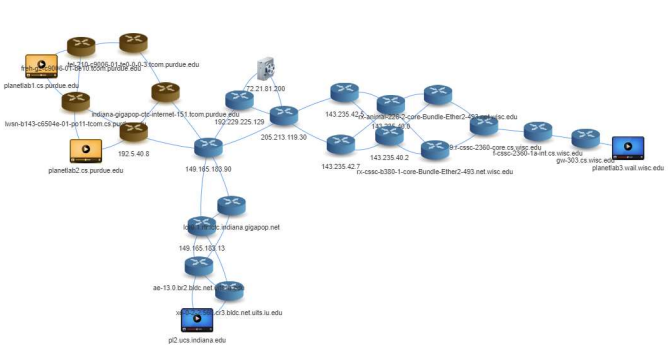
Figure 57: The count and the ave duration of QoE anomalies in various types of devices

6.3 Root cause analysis for QoE anomalies

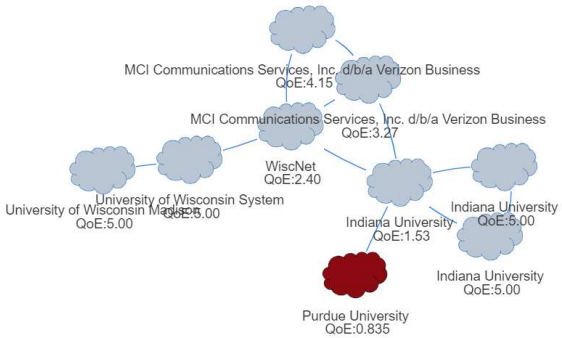
6.3.1 Root Cause Analysis for Persistent QoE Anomalies

Transit and access networks in our experiments cause most persistent QoE anomalies. We analyze two persistent QoE anomalies that are identified in transit/access networks. From the latency measurements to those anomalous systems, we show that latencies to the anomalous networks increase and fluctuates.

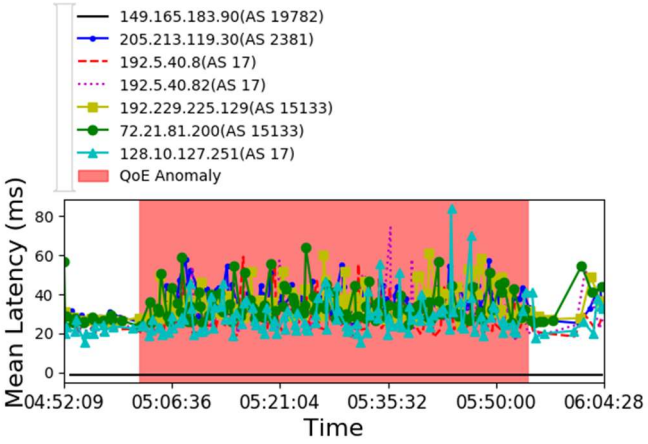
6.3.1.1 Persistent QoE anomaly identified in access network



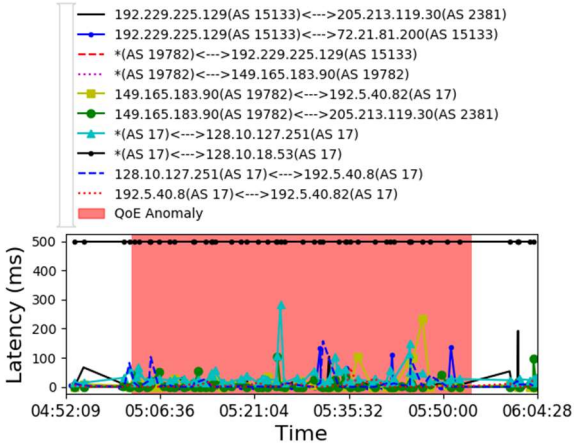
(a) Localization of suspect nodes



(b) QoE Anomaly Origins identified by QRank



(c) Latencies to all routers involved in the video streaming



(d) Inferred latencies of all links in the video streaming

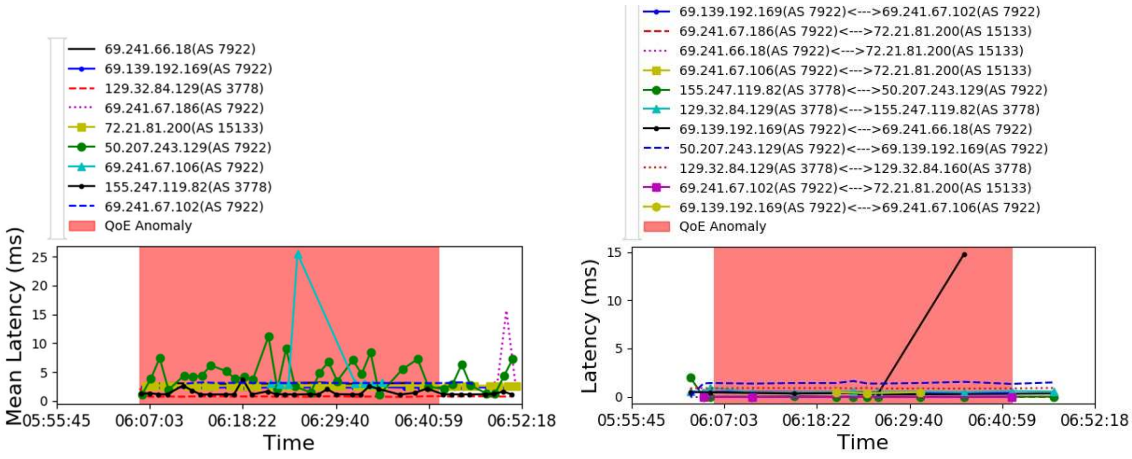
Figure 58: A persistent QoE anomaly identified in access network

In Figure 58, we show an example of persistent QoE anomaly on user "planetlab2.cs.purdue.edu" identified in access network AS17 (Purdue University). Figure 58 (a) draws the located suspect nodes that are exclusively on the routes of users with QoE anomalies. Figure 58 (b) highlights the access network that is identified by QRank as the anomalous system. QRank compares QoE scores among all networks involved and identify the one with the lowest aggregated QoE (0.835) as the anomalous system. We then show the probed latencies to all routers

that are on the user's path in Figure 58 (c). It shows that the latencies from the user to the routers in the network AS 17 fluctuate a lot. AS17 is the access network. The latencies to those routers are expected to be much lower than routers in other networks. However, we observe that the latencies to the router 128.10.127.251 in AS17 is similar to or just slightly lower than the latencies to the server 72.21.81.200. Sometimes, the latencies to 128.10.127.251 can increase up to longer than 80 milliseconds. The maximum latency to the router is even longer than the maximum latency to the server. It indicates that the end-to-end latency to the server is mainly induced by the latency in access network AS17. We then estimate the link latencies from per traceroute data probed every 10 minutes. We draw the estimated latencies for all links on the user's path in Figure 58 (d). We observe that the latencies for links in AS17 are on average larger and fluctuate more. Both users through AS17 network experience persistent QoE anomalies. It indicates the emulated DASH users choose the lowest bitrate during the anomaly. It is reasonable to infer that there is not enough capacity in network AS17, which incur the QoE anomalies

6.3.1.2 Persistent QoE anomaly identified in transit network

In Figure 59, we show the network measurements during an example persistent QoE anomaly. The anomaly is identified in transit network AS7922 (Comcast Cable Communications, LLC). The QoE anomaly last 2193 seconds on emulated user "planetlab1.temple.edu". We probe all routers on the user's path to its cache server. The latencies to all routers from the user is shown in Figure 59 (a). The monitoring agent on the user probes all routers with 10 pings every minute. The latency is estimated by the mean of round trip times of 10 pings. It shows that the latencies from user to several routers in AS7922 increases frequently. These routers include 69.241.67.106, 50.207.243.129, and 69.241.67.186.



(a) **Left:** Mean latencies probed from a user to all routers on the session's path (Mean latencies from 10 pings) (b) **Right:** Inferred latencies on all links (Traceroute)

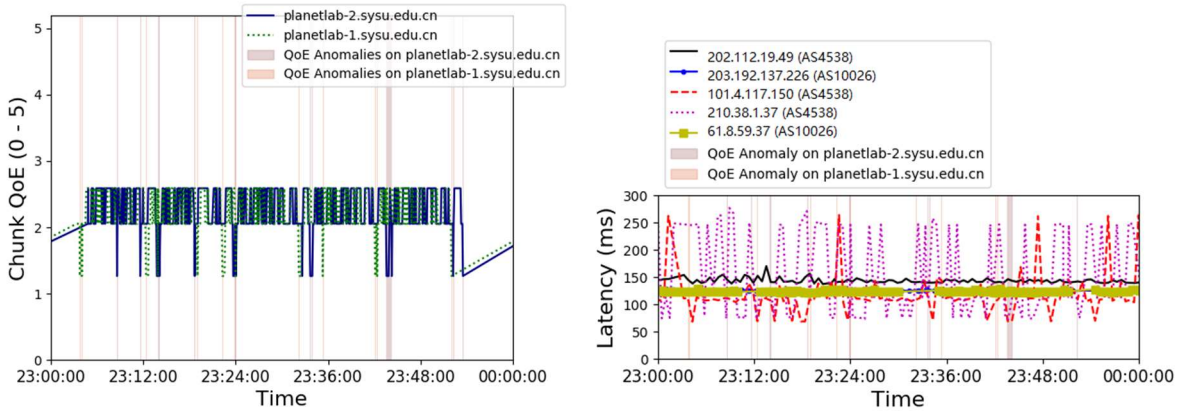
Figure 59: A persistent QoE anomaly identified in transit network

We also probe the cache server with traceroute measurement every 10 minutes. We estimate link latencies from latencies of adjoining hops. The link latencies during the QoE anomaly period is shown in Figure 59 (b). It shows that the latencies on link between router 50.207.243.129 and the router 69.139.192.169 are on average higher than latencies on other links. It can be verified in Figure 59 (a) as well. The probed latencies to router 69.139.192.169 is low. The maximum RTT probed on 69.139.192.169 is 2.5259 ms. However, the probed latencies to router 50.207.243.129 is on average high (4.7534 ms) and is with a lot of fluctuations. The maximum RTT to 50.207.243.129 is 33.4833 ms. This can be caused by dynamic queue length on this router or. It also indicates that the traffic through the transit network is bursty. This might cause the QoE anomalies on users who stream through this network.

6.3.2 Root Cause Analysis for Recurrent QoE Anomalies

During 100-hour experiments, we find that access networks, transit networks and the user devices incur recurrent QoE anomalies. We in the following give examples to analyze possible root causes of these recurrent QoE anomalies.

6.3.2.1 Recurrent QoE anomaly identified in access network



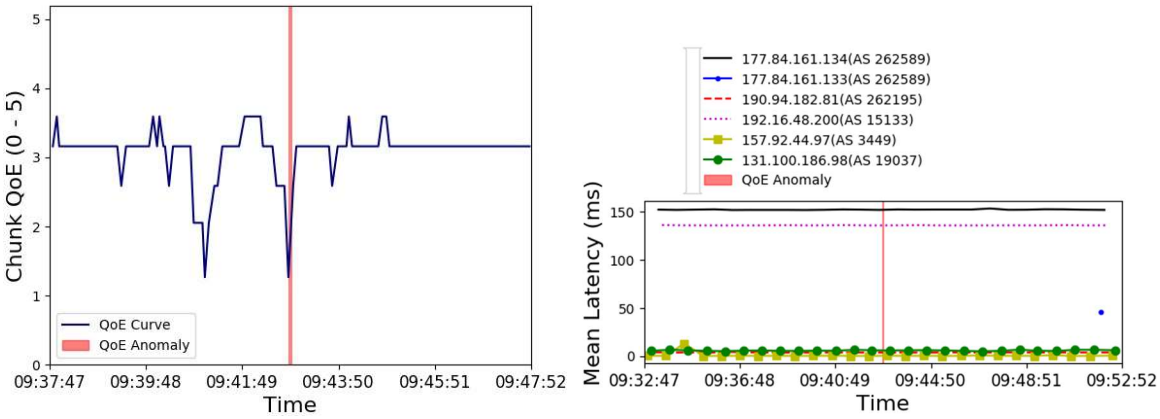
(a) Left: QoEs on anomalous users (b) Right Latencies probed from a monitoring agent to routers in suspect networks.

Figure 60: A reccurent QoE anomaly identified in access network

Figure 60 (a) showed recurrent QoE anomalies on user “planetlab-1.sysu.edu.cn” and “planetlab-2.sysu.edu.cn” who both access Internet through the network AS4538. The QRank identifies the access network AS4538 as the anomalous system. We then probe latencies from a monitoring agent in the closest Azure region to several routers in the network AS4538 and some routers in transit network AS10026. In Figure 60 (b). It shows that the probing latencies to the routers in AS4538 increase above 200 ms frequently. We also highlighted all anomaly periods in red and pink bars in Figure 60 (b). Many QoE anomalies co-occur with those latency peaks. From

Figure 60 (a), we see that both emulated users have QoE just above 2 when there are no QoE anomalies. It can be inferred that there is not sufficient capacity for them to stream higher bitrates. When the probing latencies to routers in AS4538 increases, the packets going through the network gets longer delays, resulting QoE below 2. We also study the probing latencies to various routers in AS4538 through a 2-day period. The latencies show strong fluctuations. We find that the peak latencies (larger than 200 ms) occur recurrently with an average interval of 83.83 seconds. It always goes up to 250 ms.

6.3.2.2 Recurrent QoE anomaly identified in transit network



(a) Left: QoEs on the user with recurrent QoE anomaly (b) Right: Latencies probed from the user to all routers on user's path to its cache server

Figure 61: A reccurent QoE anomaly identified in recurrent network

In Figure 56, we show the count of recurrent QoE anomalies identified in various transit networks. The top anomalous transit networks are AS 3491, AS19037, AS262195, AS262589, etc. We pick up a QoE anomaly on a user streaming through AS262589 to show possible root causes of QoE anomalies in transit networks. The recurrent QoE anomaly is on user "planet-lab4.uba.ar". The anomaly occurs very often during the streaming. We only draw QoE curve in a 10-minute

time window as shown in Figure 61 (a). QRank identifies the anomaly in the transit network AS262589. We let a monitoring agent on the user probing all routers on its route to the cache server. Figure 61 (b) shows the latencies to all routers. The latencies to all routers seem very steady. There is one router 177.84.161.134 in AS262589 with long latency (around 150 ms). The probed latency to the router is even longer than the probed latency to the server, which is 192.16.48.200 in AS15133. It is either due to the long queue length on the router or the low priority of Ping traffic on the router. Long queue indicates that the capacity of the router is barely adequate to handle the traffic. We also observe that user "planet-lab4.uba.ar" overall has a medium QoE with chunk QoE values from 2 to 3. The QoE frequently drops below 2 but the QoE anomalies usually last less than 10 seconds. We infer that the recurrent QoE anomalies are related to the recurrent increases of traffic in AS262589. As the capacity in AS262589 is barely adequate and can provide QoE just above 2, a slide increase in traffic can decrease the user QoE. Besides, given a certain bitrate, the volume of video traffic also dynamically changes according to its content.

6.3.2.3 Recurrent QoE anomaly identified in user devices

From Figure 49, we notice that the user with the most number of recurrent QoE anomalies is "planetlab1.rutgers.edu". The anomalous system for the anomaly is identified in the device. The device is a PlanetLab node installed with Fedora 14 Laughlin OS. It runs our emulation code of DASH player in the environment of Python 2.7.0. In Figure 62, QRank identifies the client node as the anomalous node because it is the only node exclusively on anomalous user's path. There are many such QoE anomalies. We find that all persistent and recurrent QoE anomalies in devices are on PlanetLab nodes. We infer that these PlanetLab nodes may have outbound capacity limit. There are only 4.9% QoE anomalies identified in Azure A0 and A2 instances. These are all occasional

QoE anomalies. Azure A0 and A2 instances are the most economical virtual machines that share physical resources with other tenants. Their performance can degrade occasionally under interference.

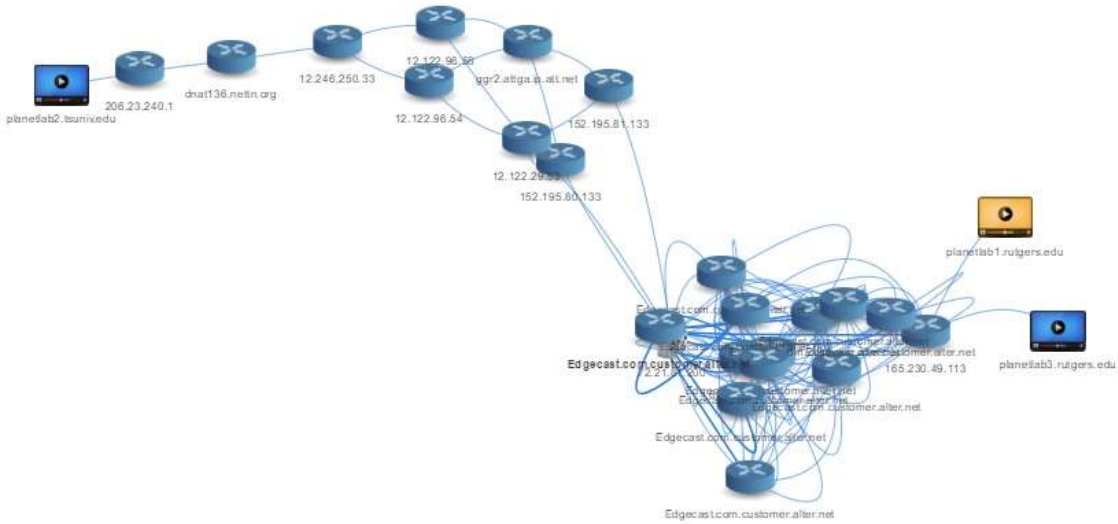
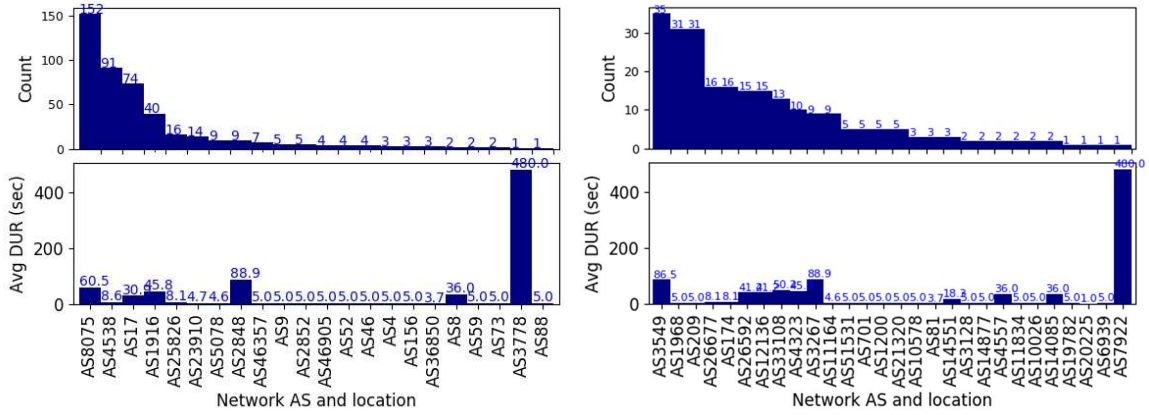


Figure 62: QoE anomaly identified in device

6.3.3 Root Cause Analysis for Occasional QoE Anomalies



(a) Access networks (b) Transit networks

Figure 63: Occasional QoE anomaly identified in various networks

Occasional QoE anomalies are widely identified in various types of networks. In the above sections, we discuss persistent and recurrent QoE anomalies. More than 95% of all persistent and recurrent QoE anomalies are identified in 3 access and 2 transit networks. The occasional QoE anomalies are identified in more networks. The occasional QoE anomalies distribute over various anomalous networks following long tail distributions as shown in Figure 63. There are no special patterns observed on latencies to these networks. These anomalies usually last very short period and do not recur. We therefore infer that these occasional QoE anomalies are caused by occasional bursty traffic in these networks. There is only one occasional QoE anomaly identified in the Cloud network. It is on user "planetlab1.cesnet.cz". We study the anomaly identification result from QRank. We find that the anomaly is identified in all systems in the video streaming. QRank identifies anomalous system purely based on users' QoE. If there is no other users using the same network, QRank assumes the network is a suspect to cause the anomaly. We notice that the only QoE anomaly identified in Cloud network is due to the insufficient accuracy of QRank system. If more users are using the Cloud network, the QRank would be able to identify the QoE anomaly in one true anomalous system rather than in all networks.

6.4 Summary

As video services starts migrating to the Cloud, video service providers are wondering whether the Cloud can provide good Quality of Experience (QoE) for their users. In this paper, we ran 124 emulated users around the world to perform DASH video streaming from Microsoft Azure Cloud CDN to measure the performance of the Cloud CDN in terms of user QoE. We collected QoE anomalies from 12400 video sessions and identified the anomalous systems that cause those QoE anomalies. Interestingly, the Cloud CDN does not incur any QoE anomalies. Instead, transit

networks, access networks and devices are major causes of QoE anomalies. Besides, more than 91.4% QoE anomalies are experienced by only 15.32% users and these users experience QoE anomalies either persistently or recurrently. 2 transit networks and 3 access networks incur more than 95% of all persistent QoE. 6 access networks and 10 transit networks incur more than 95% of all recurrent QoE anomalies. If capacity in these anomalous networks can increase, more than 95% QoE anomalies would not occur. We conclude that to provide good QoE for video services, the Cloud provider should work with access/transit ISPs to increase the capacity of end-to-end connections.

7. CONCLUSIONS

We believe that end-user QoE provides accurate assessment of the system performance. We propose to apply end user QoE in the management and control of large-scale VoD systems in the Cloud. Specifically, we verify the effectiveness of QoE based management and control in three systems.

QMan controls the server selection adaptively based on end user QoE. Our extensive evaluations show that by using QoE as a principle to select servers, QMan improves overall user QoE over common measurement based server selection. QMan system provides 9% to 30% more users with QoE above the MOS "Good" level than the existing measurement based server selection systems. For users who are impacted by dynamic and severe interference in the Cloud, QMan can improve the QoE from the "bad" to "fair" in MOS level. Results also show that QMan can discover operational failures by QoE based server monitoring and prevents streaming session crashes.

QWatch uses end-user QoE to detect QoE anomalies and correlate users' data to locate QoE anomalies. We run extensive experiments in a controlled VoD system and production Cloud (Azure Cloud and CDN) to validate QWatch's accuracy in detection and localization. We find numerous false positives and false negatives in production Cloud when system metric based anomaly detection methods are used. QWatch correctly detects and locates anomalies in controlled experiments.

QRank learns the performance of various suspect systems via end user QoE and identifies the QoE anomaly in the systems with low QoE scores. By injecting anomalies in different systems in a controlled VoD system, we verify that QRank can correctly identify the anomalous system causing the QoE anomaly. We run QRank for video streaming in the production Cloud

environment for 2 days. By analyzing 9140 QoE anomalies from 12400 video sessions in the Cloud, we find that the Cloud CDN server does not incur any QoE anomalies. Instead, transit networks, access networks and devices are major causes of QoE anomalies. Among those anomalies, a small number of users (15 out of 124) experience around 87.8% anomalies recurrently. 11 out of 124 users experience long QoE anomalies, on average lasting more than 15 minutes. From QRank identification results, we show that more than 95% of persistent and recurrent QoE anomalies are incurred by less than 10 networks. We argue that to provide good QoE for video services, the Cloud provider should identify networks that may become bottlenecks for high quality video streaming and appropriate peer with Internet Service Providers (ISPs) to improve the capacity of end-to-end connections.

8. FUTURE WORK

QMan system shows that applying user QoE in the adaptive control of server selection is effective in improving overall user experience. In order to apply QMan, VoD providers need to have the access to select at server level. Most CDN providers do not grant this control to VoD providers. To have some flexibility in choosing server, VoD systems adopts multi-CDN strategies. Netflix and Hulu use Akamai, Level 3, and Limelight at the same time. As QoE is an effective measurement to predict the performance of various systems including the CDN, the QMan system can be extended to multi-CDN adaptation systems in the future.

By running QRank systems in three production Clouds, Microsoft Azure, Google Cloud and Amazon Web Service, we find that popular Cloud providers differ a lot in infrastructure deployment, technology adoption, network peering and pricing. All of these would affect user QoE for a particular application. How an application provider chooses the Cloud to achieve both cost-efficient services and good overall user QoE is a challenging task.

From our experiments, we show that the accuracy of QWatch and QRank heavily relies on the number of users using a system. The more user QoE observed on a system, the more accuate the QoE based monitoring can provide. If end user QoE data can be collected from large-scale commercial VoD, such as YouTube and Netflix, interesting insights can be gained about the performance issues in Internet, production Cloud and CDN systems.

REFERENCES

[1] Sandvine. "Global Internet Phenomena Report, 2H 2014". https://www.sandvine.com/trends/global-internet-phenomena/

[2] X. Zhang, J. Liu, B. Li, and P. Yum, "DONet/Coolstreaming: A Data-driven Overlay Network for Live Media Streaming", *Proceedings of IEEE 27th Conference on Computer Communications (INFOCOM)*. Vol.3, pp.2102-2111. IEEE, 13-17 March 2005.

[3] Huang, Yan, et al. "Challenges, design and analysis of a large-scale p2p-vod system." *ACM SIGCOMM computer communication review*. Vol. 38. No. 4. ACM, 2008.

[4] Silverston, Thomas, and Olivier Fourmaux. "Measuring p2p iptv systems." *Proc. of ACM NOSSDAV*. Vol. 7. ACM, 2007.

[5] Vakali, Athena, and George Pallis. "Content delivery networks: Status and trends." *IEEE Internet Computing,* vol. 7, no.6, pp. 68-74. IEEE, 2003.

[6] The Netflix Tech Blog. "*Four Reasons We Choose Amazon's Cloud as Our Computing Platform*". December, 2010.

[7] Armbrust, Michael, et al. "A view of cloud computing." *Communications of the ACM*, 53(4), p50-58. ACM, 2010.

[8] Wu, Yu, et al. "Cloudmedia: When cloud on demand meets video on demand." *31st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2011.

[9] Tickoo, Omesh, et al. "Modeling virtual machine performance: challenges and approaches." *ACM SIGMETRICS Performance Evaluation Review*. Vol. 37, no.3, pp55-60. ACM, 2010

[10] Fiadino, P.; D'Alconzo, A.; Bar, A.; Finamore, A.; Casas, P., "On the detection of network traffic anomalies in content delivery network services," *26th International Teletraffic Congress (ITC)*, pp.1-9, 9-11. IEEE, 2014

[11] Casas, P.; D'Alconzo, A.; Fiadino, P.; Bar, A.; Finamore, A., "On the analysis of QoE-based performance degradation in YouTube traffic," *10th International Conference on in Network and Service Management (CNSM)*, pp.1-9, 17-21 Nov. IEEE, 2014

[12] Markopoulou, Athina, et al. "Characterization of failures in an operational IP backbone network." *IEEE/ACM Transactions on Networking (TON)*, Vol. 16, No. 4, pp749-762. ACM, 2008

[13] Google Video Quality Report, “The Methodology – How we verify that an Internet Provider can consistently serve YouTube in HD?”, https://www.google.com/get/videoqualityreport/#methodology

[14] Jiang, Junchen, et al. "Shedding light on the structure of internet video quality problems in the wild." *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013.

[15] Yaping Zhu; Helsley, B.; Rexford, J.; Siganporia, A.; Srinivasan, S., "LatLong: Diagnosing Wide-Area Latency Changes for CDNs," *IEEE Transactions on Network and Service Management*, vol.9, no.3, pp.333-345. IEEE, September 2012

[16] Nathuji, Ripal, Aman Kansal, and Alireza Ghaffarkhah. "Q-clouds: managing performance interference effects for qos-aware clouds." *Proceedings of the 5th European conference on Computer systems*. ACM, 2010.

[17] Casas P., D'Alconzo A., Fiadino P., Bar A., Finamore A., Zseby T., “When YouTube Does not Work—Analysis of QoE-Relevant Degradation in Google CDN Traffic," *IEEE Transactions on in Network and Service Management*, vol.11, no.4, pp.441-457. IEEE, Dec. 2014

[18] Shackelford, Adam. "CloudFront and DNS Management." *Chapter 4 in Beginning Amazon Web Services with Node. js*. pp.93-120. Apress, 2015.

[19] Padhy, Rabi Prasad, Manas Ranjan Patra, and Suresh Chandra Satapathy. "X-as-a-Service: Cloud Computing with Google App Engine, Amazon Web Services, Microsoft Azure and Force." *Com. Int. J. Comput. Sci. Telecommun*. Vol.2, Issue 9, pp8-16. IJCST, 2011.

[20] Ganjam, Aditya, et al. "C3: Internet-Scale Control Plane for Video Quality Optimization." *Proceedings of the 12th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2015.

[21] Liu, Xi, et al. "A case for a coordinated internet video control plane." *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 2012.

[22] Mukerjee, Matthew K., et al. "Practical, Real-time Centralized Control for CDN-based Live Video Delivery." *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015.

[23] "Statistics of YouTube", https://www.youtube.com/yt/press/statistics.html

[24] "Number of Netflix streaming subscribers worldwide from 3rd quarter 2011 to 2nd quarter 2015 (in millions)", http://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide/

[25] Torres, Ruben, et al. "Dissecting video server selection strategies in the youtube CDN." *31st International Conference on Distributed Computing Systems (ICDCS).* IEEE, 2011.

[26] Adhikari, Vijay Kumar, et al. "Unreeling netflix: Understanding and improving multi-cdn movie delivery." *2012 Proceedings IEEE INFOCOM*. IEEE, 2012.

[27] Adhikari, V.K.; Guo, Y.; Hao, F.; Hilt, V.; Zhang, Z.-L.; Varvello, M.; Steiner, M., "Measurement Study of Netflix, Hulu, and a Tale of Three CDNs," *IEEE/ACM Transactions on Networking*, no.99, pp.1-1. ACM, 2014.

[28] Adhikari, V.K.; Jain, S.; Zhi-Li Zhang, "Where Do You "Tube"? Uncovering YouTube Server Selection Strategy," *2011 Proceedings of 20th International Conference on in Computer Communications and Networks (ICCCN)*, pp.1-6. IEEE, 2011

[29] Su, Ao-Jan, et al. "Drafting behind Akamai (travelocity-based detouring)." *ACM SIGCOMM Computer Communication Review*. Vol. 36. No. 4. ACM, 2006.

[30] Barker, Sean Kenneth, and Prashant Shenoy. "Empirical evaluation of latency-sensitive application performance in the cloud." *Proceedings of the first annual ACM SIGMM conference on Multimedia systems.* ACM, 2010

[31] Pertet, Soila, and Priya Narasimhan. "Causes of failure in web applications (cmu-pdl-05-109)." *Parallel Data Laboratory (2005)*: 48.

[32] R. Jain, "The Art of Computer System Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling" *New York NY: Wiley-Interscience*, 1991.

[33] Amazon Web Service, "Amazon CloudFront Service Level Agreement", http://aws.amazon.com/cloudfront/sla/

[34] Google Cloud, "Google Compute Engine Service Level Agreement (SLA)", https://cloud.google.com/compute/sla?hl=en

[35] Aceto, Giuseppe, et al. "Cloud monitoring: A survey." *Computer Networks 57.9 (2013)*: 2093-2115.

[36] Olshefski, David P., Jason Nieh, and Dakshi Agrawal. "Inferring client response time at the web server." *ACM SIGMETRICS Performance Evaluation Review.* Vol. 30. No. 1. ACM, 2002.

[37] Wei, Jianbin, and Cheng-Zhong Xu. "sMonitor: A Non-Intrusive Client-Perceived End-to-End Performance Monitor of Secured Internet Services." *USENIX Annual Technical Conference, General Track.* 2006.

[38] Fu, Yun, et al. "EtE: Passive End-to-End Internet Service Performance Monitoring." *USENIX Annual Technical Conference, General Tra*ck. 2002.

[39] Kim, Jiyeon; Kim, Hyong S., "PBAD: Perception-Based Anomaly Detection System for Cloud Datacenters," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on* , vol., no., pp.678-685, 2015

[40] ITU-T RECOMMENDATION, P. "Subjective video quality assessment methods for multimedia applications." (1999): 34-35.

[41] G12, I. T. U. T. "*Definition of Quality of Experience.*" TD 109rev2 (PLEN/12), Geneva, Switzerland (2007): 16-25

[42] Garcia, M.-N., et al. "*Quality of experience and HTTP adaptive streaming: A review of subjective studies*", QoMex, 2014

[43] Van Kester, S., et al. "Estimating the impact of single and multiple freezes on video quality." *IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics*, 2011.

[44] Reichl, Peter, Bruno Tuffin, and Raimund Schatz. "*Logarithmic laws in service quality perception: where microeconomics meets psychophysics and quality of experience.*" Telecommunication Systems 52.2 (2013): 587-600.

[45] Stockhammer, Thomas. "Dynamic adaptive streaming over HTTP --- standards and design principles." *Proceedings of the second annual ACM conference on Multimedia systems (MMSys)*. ACM, 2011.

[46] Akhshabi, Saamer, Ali C. Begen, and Constantine Dovrolis. "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP." *ACM Proceedings of the second annual ACM conference on Multimedia systems (MMSys)*. ACM, 2011.

[47] Zhang, Weiwen, et al. "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks." *IEEE Transactions on Multimedia*. IEEE, 15.6 (2013): 1431-1445.

[48] Mok, Ricky KP, et al. "QDASH: a QoE-aware DASH system." *Proceedings of the 3rd Multimedia Systems Conference (MMSys)*. ACM, 2012.

[49] Mok, Ricky KP, Edmond WW Chan, and Rocky KC Chang. "Measuring the quality of experience of HTTP video streaming." *2011 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2011.

[50] De Vriendt, Johan, Danny De Vleeschauwer, and Doug Robinson. "Model for estimating QoE of video delivered using HTTP adaptive streaming." *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2013.

[51] Shaikh, Anees, Renu Tewari, and Mukesh Agrawal. "On the effectiveness of DNS-based server selection." INFOCOM 2001. *Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies.* Vol. 3. IEEE, 2001.

[52] Pan, Jianping, Y. Thomas Hou, and Bo Li. "An overview of DNS-based server selections in content distribution networks." *Computer Networks 43.6 (2003)*: 695-711.

[53] Bakiras, Spiridon. "Approximate server selection algorithms in content distribution networks." *IEEE International Conference on Communications (ICC)*. Vol. 3. IEEE, 2005.

[54] Gao, Jun, and Peter Steenkiste. "Design and evaluation of a distributed scalable content discovery system." *IEEE Journal on Selected Areas in Communications*, 22.1 (2004): 54-66.

[55] Chen Wang, Hyong Kim, Ricardo Morla "QoE Driven Server Selection for VoD in the Cloud," *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, IEEE, 2015

[56] Barto, Andrew G. "*Reinforcement learning: An introduction*". MIT press, 1998

[57] D. Mytthon. (2014, March) 5 things you probably don't know about Google Cloud. [Online]. Available: https://gigaom.com/2014/03/02/5-things-you-probably-dont-know-about-google-cloud/

[58] Cha, Meeyoung, et al. "Analyzing the video popularity characteristics of large-scale user generated content systems." *IEEE/ACM Transactions on Networking (TON)* 17.5 (2009): 1357-1370.

[59] B. Hubert, "The wonder shaper", http://lartc.org/wondershaper/, 2002.

[60] Amos Waterland, "The stress tool", http://people.seas.harvard.edu/~apw/stress/

[61] Yin, Xiaoqi, et al. "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP." *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM).* ACM, 2015.

[62] Dykes, Sandra G., Kay Robbins, and Clinton L. Jeffery. "An empirical evaluation of client-side server selection algorithms." *Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM).* Vol. 3. IEEE, 2000.

[63] Andresen, Daniel, et al. "Sweb: Towards a scalable World Wide Web server on multicomputers." *Proceedings of International Parallel Processing Symposium (IPPS'96).* IEEE, 1996.

[64] Fiadino, P.; D'Alconzo, A.; Bar, A.; Finamore, A.; Casas, P., "On the detection of network traffic anomalies in content delivery network services," *26th International in Teletraffic Congress (ITC)*, pp.1-9, 9-11 Sept. 2014

[65] D'Alconzo, Alessandro, et al. "Who to blame when YouTube is not working? Detecting anomalies in CDN-provisioned services." *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2014.

[66] Akhshabi, Saamer, Ali C. Begen, and Constantine Dovrolis. "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP." *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011.

[67] Olfati-Saber, Reza, Alex Fax, and Richard M. Murray. "Consensus and cooperation in networked multi-agent systems." *Proceedings of the IEEE,* vol. 95, no. 1, pp.215-233. IEEE, 2007

[68] Panait, Liviu, and Sean Luke. "Cooperative multi-agent learning: The state of the art." *Autonomous Agents and Multi-Agent Systems*, vol. 11, no.3, pp.387-434. ACM, 2005.

[69] Talia, Domenico. "Clouds meet agents: Toward intelligent cloud services." *IEEE Internet Computing*, vol. 2, pp78-81. IEEE, 2012

[70] Bendoukha, Sofiane, Daniel Moldt, and Thomas Wagner. "Enabling cooperation in an inter-cloud environment: An agent-based approach." *24th International Workshop on Database and Expert Systems Applications (DEXA)*. IEEE, 2013.

[71] Shilton, Katie, et al. "Participatory design of sensing networks: strengths and challenges." *Proceedings of the Tenth Anniversary Conference on Participatory Design 2008*. Indiana University, 2008.

[72] Liang, Huiguang, et al. "I've heard you have problems: Cellular signal monitoring through UE participatory sensing." *2014 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2014.

[73] Lamport, Leslie, Robert Shostak, and Marshall Pease. "The Byzantine generals problem." *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 4, issue 3, pp382-401. ACM, 1982.

[74] Haeberlen, Andreas, and Petr Kuznetsov. "The fault detection problem." *Principles of Distributed Systems*, pp99-114. Springer Berlin Heidelberg, 2009.

[75] Castro, Miguel, and Barbara Liskov. "Practical Byzantine fault tolerance." *OSDI*. Vol. 99. 1999.

[76] Microsoft Azure Cloud, "Microsoft Azure Service Level Agreements", https://azure.microsoft.com/en-us/support/legal/sla/

[77] Zhao, Yuhong, et al. "Meeting service level agreement cost-effectively for video-on-demand applications in the cloud." 2014 Proceedings IEEE INFOCOM. IEEE, 2014.

[78] Marilly, Emmanuel, et al. "Service level agreements: a main challenge for next generation networks." 2nd European Conference on Universal Multiservice Networks (ECUMN 2002). IEEE, 2002.

[79] Hobfeld, Tobias, et al. "Challenges of QoE management for cloud applications." IEEE Communications Magazine. IEEE, 2012.

[80] Chun, Brent, et al. "PlanetLab: an overlay testbed for broad-coverage services." ACM SIGCOMM Computer Communication Review 33.3 (2003): 3-12.

[81] Google Cloud Platform, "Compute Engine ---- Scalable, High-Performance Virtual Machines", https://cloud.google.com/compute/

[82] Tran, Hai Anh, et al. "QoE-based server selection for content distribution networks." IEEE Transactions on Computers 63.11 (2014): 2803-2815.

[83] Nygren, Erik, Ramesh K. Sitaraman, and Jennifer Sun. "The Akamai network: a platform for high-performance internet applications." SIGOPS Operating Systems Review. ACM, 2010.

[84] Yin, Hao, et al. "Content delivery networks: a bridge between emerging applications and future IP networks." IEEE Network. IEEE, 2010.

[85] "Amazon Web Service", https://aws.amazon.com/

[86] "Microsoft Azure", https://azure.microsoft.com/en-us/

[87] "Amazon CloudFront", https://aws.amazon.com/cloudfront/

[88] "AWS Elastic Load Balancing", https://aws.amazon.com/elasticloadbalancing/

[89] "Microsoft Azure Load Balancer", https://azure.microsoft.com/en-us/services/load-balancer/

[90] "Amazon CloudWatch", https://aws.amazon.com/cloudwatch/

[91] "How to monitor Cloud service?", Microsoft Azure, https://azure.microsoft.com/en-us/documentation/articles/cloud-services-how-to-monitor/

[92] Mahimkar, Ajay Anil, et al. "Towards automated performance diagnosis in a large IPTV network." SIGCOMM Review. ACM, 2009.

[93] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, Dina Papagiannaki, Peter Steenkiste, "Identifying the Root Cause of Video Streaming Issues on Mobile Devices", CoNEXT. ACM, 2015.

[94] An IP info lookup API. https://ipinfo.io/

[95] Huffaker B, Plummer D J, Moore D J, et al. "Topology discovery by active probing". Symposium on Applications and the Internet, 2002.

[96] Donnet B, Raoult P, Friedman T, et al. "Efficient algorithms for large-scale topology discovery". Sigmetrics Performance Evaluation Review, 2005.

[97] Donnet B, Raoult P, Friedman T, et al. "Deployment of an Algorithm for Large-Scale Topology Discovery". IEEE Journal on Selected Areas in Communications, 2006.

[98] Spring N, Mahajan R, Wetherall D, et al. "Measuring ISP topologies with Rocketfuel". IEEE/ACM Transactions on Networking, 2004.

[99] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." CSUR. ACM, 2009.

[100] Pukelsheim, Friedrich. “The Three Sigma Rule”. The American Statistician. American Statistical Association, 1994.

[101] “Performance Co-Pilots”, http://www.pcp.io/

[102] “Stress – tool to impose load on and stress test systems”, http://linux.die.net/man/1/stress

[103] “Network Emulator in Linux traffic control facilities”, http://man7.org/linux/man-pages/man8/tc-netem.8.html

[104] Aceto Giuseppe, et al. "Cloud monitoring: A survey." Computer Networks. ELSEVIER, 2013

[105] "View Amazon EC2 Metrics", http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/viewing_metrics_with_cloudwatch.html.

[106] Breitman, Karin, et al. "When TV dies, will it go to the cloud?" Computer 43.4 (2010): 81-83.

[107] Wang, Chen, Hyong Kim, and Ricardo Morla. "Users Know Better: A QoE based Adaptive Control System for VoD in the Cloud." Global Communications Conference (GLOBECOM), 2015 IEEE. IEEE, 2015.

[108] Wang, Chen, Hyong Kim, and Ricardo Morla. "QWatch: Detecting and Locating QoE Anomaly for VoD in the Cloud." Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on. IEEE, 2016.

[109] Ahmed, Adnan, Zubair Shafiq, and Amir Khakpour. "QoE Analysis of a Large-Scale Live Video Streaming Event." Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science. ACM, 2016.

[110] Jiang, Junchen, et al. "Pytheas: Enabling Data-Driven Quality of Experience Optimization Using Group-Based Exploration-Exploitation." NSDI. 2017.

[111] IP Location, https://www.iplocation.net/, visited on June 26, 2017

[112] BGP view, https://bgpview.io/, visited on June 26, 2017