QWatch: Detecting and Locating QoE anomaly for VoD in the Cloud

Chen Wang*†, Hyong Kim*, Ricardo Morla†
chenw@cmu.edu, kim@ece.cmu.edu, ricardo.morla@fe.up.pt
*Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA
†INESC Porto and Faculty of Engineering, University of Porto, Porto, Portugal

***Abstract*—Commercial large-scale VoD systems such as Netflix and Hulu rely on CDNs to deliver videos to users around the world. Various anomalies occur often and degrade users' Quality of Experience (QoE). Detecting and locating such anomalies are highly complex due to a large number of different entities involved in the end-to-end video delivery. These entities include VoD provider, CDN/Cloud providers, transit ISPs, access ISPs, and end user devices. QoE perceived by the users is a critical metric for VoD providers. We propose *QWatch*, a scalable monitoring system, which detects and locates anomalies based on the end user QoE in real-time. We evaluate *QWatch* in a controlled VoD system and production Microsoft Azure Cloud and CDN. *QWatch* effectively detects and locates QoE anomalies in our extensive experiments. We discuss insights obtained from running VoD system with 200 worldwide users in production Cloud[1].**

I. 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) [1]. CDN, such as Akamai, Level 3 Communications and Limelight Networks, provide the content delivery [2]. Cloud providers, such as Microsoft and Amazon, 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.

Anomalies affect Service Level Agreements (SLAs), such as virtual machine (VM) uptime and availability. SLA violations sometimes do not degrade user QoE as shown in our experiments. We believe SLA is not sufficient to ensure QoE. Server system metrics, such as utilization and throughput of CPU, memory, disk and network do not fully reflect the user experience of VoD in the Cloud. There are many other factors in Cloud including transit and client networks that impact 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.

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 would 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 in later sections.

II. Related Work

Earlier works of anomaly detection in VoD services use various system parameters to infer QoE issues. Ajay et al [3] collect various system metrics in a large IPTV network and apply supervised learning algorithm to learn how these metrics are related to customer call records noted as anomalies. [4] [5] detect anomalies based on critical network/server metrics that could possibly impact end-user QoE. They tend to have many false positives and false negatives. Selecting 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 [6]. Its detection requires off-line computation. There are several works on identifying, locating and diagnosing QoE issues. Junchen et al analyze end-user data using unsupervised learning to find the root cause of QoE problems [6]. Giorgos et al diagnose QoE issues by supervised learning on various network and system metrics from different vantage points [7]. There are also commercial data analysis programs that

[1]This work was supported in part by the FCT under Grant SFRH/BD/51150/2010, CMU-SYSU CIRC, SYSU-CMU IJRI and an Azure Research grant provided by Microsoft Corporation.

9781-5090-1445-3/16$31.00 © 2016 IEEE (CloudCom'16)

statistically infer possible root causes of QoE issues (YouTube) [8] [9]. Our previous work analyze QoE data in real time to manage and control VoD systems [10] [11]. In this work, we apply the QoE analysis for anomaly detection and localization.

III. System Overview

A. Background

VoD systems mostly use third-party CDNs for the content caching and content delivery. CDNs cache popular videos in their edge servers in different geographical locations. Video contents are delivered to users from the closest edge server to the user. CDN could reduce network latencies to improve end user QoE.

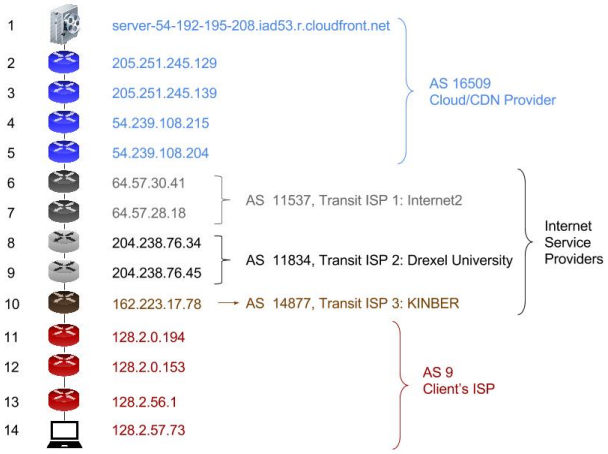


Fig. 1. An example video delivery path from AWS CloudFront to Carnegie Mellon University Network

We use a device in Carnegie Mellon University network to stream a video from a VoD website cached in Amazon CloudFront. The video goes through several networks. These networks are managed by the Cloud/CDN provider and multiple transit ISPs and a local ISP. Anomalies can occur in any part of this delivery path shown in Figure 1. In this experiment, there are 5 ISPs involved in the end-to-end video delivery path. When the user QoE degrades, 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.

B. System Design

QWatch deploys an agent in the client's video player, 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 Service Level Agreement (SLA).

Once 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.

C. 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 as shown in Figure 2. The commercial Clouds, such as Google Cloud, Amazon Web Service and Microsoft Azure, provide DNS load balancing services. Within a Cloud region, *QWatch* provisions one *locator agent* for K clients. For $N > K$ clients, $\lceil N/K \rceil$ *locator agents* are provisioned in one Cloud region. Within a Cloud region, simple load balancing mechanisms 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 that region.

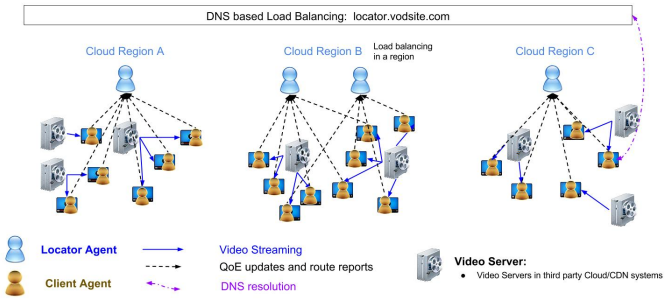


Fig. 2. *QWatch* design with horizontal scaling

IV. QoE Anomaly Detection and Localization

A. QoE Anomaly Detection

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 impact users' decision to continue the VoD service. VoD providers need to maintain at least q_0 to retain users [12]. VoD providers often conduct subjective studies to obtain q_0 for QoE [13].

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. In our system, the *client agent* monitors end user QoE $q(t)$ in real time and detects QoE anomalies when $q(t) < q_0$. QoE is a measure of acceptability of an application or service perceived subjectively by end users [14]. Various metrics, such as the join time of a video sessions, the video bitrate, the freezing time and the frequency of freezing have been considered to model QoE [15] [16].

Dynamic Adaptive Streaming over HTTP (DASH) [17] is currently the de facto video streaming technology in many commercial VoD systems (e.g. YouTube and Netflix) [18].

In DASH, a video is encoded in multiple bitrates and each bitrate version is split into a series of fixed length segments, called chunks. DASH players detect the network throughput in real time and adaptively select the bitrate for every chunk. The video bitrate and the freezing time may change for every chunk.

QWatch adopts a chunk based QoE model proposed in [19]. The chunk QoE model considers both the freezing time and the video bitrate. The chunk QoE model is a cascading model that combines existing QoE models of the freezing time and bitrate as shown in Equation (1). QoE is computed for each video chunk in the Mean Opinion Score (MOS) system [20]. QoE varies from 1 to 5 corresponding to user acceptance levels. The minimum acceptable QoE is when $q_0 = 1$.

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

The existing freezing time model and bitrate model are proposed by psychology study [21] and human vision study [22] respectively, as shown in Equations (2) and (3).$a_1, a_2, c_1 \sim c_3$ are fitted coefficients from subjective studies.

$$Q_{\text{freezing}}(\tau) = \begin{cases} 5 - \frac{c_1}{1+(\frac{c_2}{\tau})^{c_3}} & t > 0 \\ 5 & t = 0 \end{cases} \tag{2}$$

$$Q_{\text{bit-rate}}(r) = a_1 \ln \frac{a_2 r}{r_{\max}} \tag{3}$$

B. QoE Anomaly Localization

If a streaming session has an acceptable QoE, all nodes in its path are assumed to be functioning normally. We assume that if a node has an anomaly, all video sessions going through that node would have an 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 sessions 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 that is the only *suspect* node on a session's delivery path with unacceptable QoE. Rest of the nodes in this delivery path are all 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. Figure 3 illustrates how anomalies in server, router and client can be located. In Figure 3 (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 3 (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 and client X has acceptable QoE, then X is labeled *Normal* and S_1 anomaly can be located. Figure 3 (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 3 (a), X is the only *Suspect* node so it can be determined as an anomaly node. In (b) and (c), X is not in the path 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 user sessions sharing the particular path in question.

C. Implementation of QoE Anomaly Detection

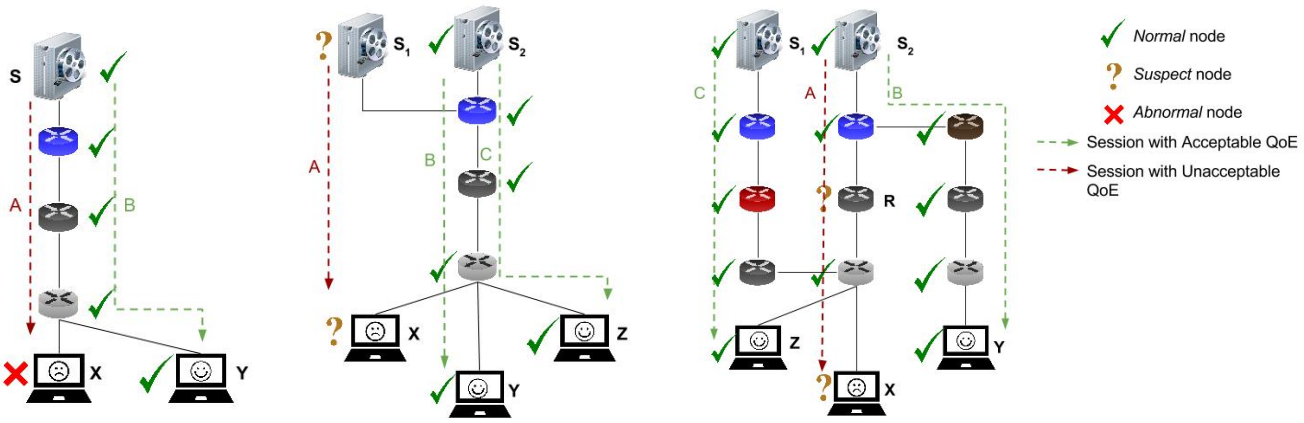
The client agent runs the *QoE Anomaly Detection Algorithm (QADA)*. The *client agent* traces its path to the video server and reports to its *locator agent*. For each video chunk, the *client agent* evaluates the chunk QoE $q(i)$ and compares with q_0. If the chunk QoE $q(i) \geq q_0$, there is no QoE anomaly. The *client agent* then reports acceptable QoE to its *locator agent* periodically. If the chunk QoE $q(i) \leq q_0$, QoE anomaly is detected and the *client agent* updates the *locator agent* immediately. The client agent runs *QADA* until the streaming session ends.

ALGORITHM 1: QoE Anomaly Detection Algorithm (QADA)

Data: Reporting period T; Chunk length: T_0; SLA for QoE: q_0
Connect to the closest locator L_k by domain name
Download the DASH description file (MPD) from a CDN host by the video URL
Obtain the cache server address S by the response
Trace the route from current client C to S and report the route $R = (C, S)$ to the locator agent L_k
Compute the reporting period in number of chunks: $N_T = T/T_0$
while *Video streaming not ends* **do**
    Download next video chunk i
    Compute QoE for current chunk $q(i)$
    Obtain current server S_i from chunk response
    if $S_i \neq S$ then
        Get the new route to the new server: $R = (C, S_i)$
        Report new route R to the locator
    if $q(i) > q(0)$ then
        QoE status $Q_i =$ *Acceptable* at time t_i
        if $(i > 0)$ *and* $(i \mod N_T == 0)$ then
            Report Update: $U_i = (t_i, Q_i, R)$ to the locator
    else
        QoE status $Q_i =$ *Unacceptable* at time t_i
        Report Update: $U_i = (t_i, Q_i, R)$ to the locator.

D. Topology Discovery from $traceroute$

When a *locator agent* receives path information from *client agents*, the network topology is obtained for all the video sessions. *Client agents* in *QWatch* probe their video servers at the start of each video streaming session and also when



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

Fig. 3. QoE anomaly localization in various cases

they are assigned to new servers. *Client agents* report path data to their closest *locator agents*. We use *traceroute* to obtain path data from *client agents* to construct tree topology connecting clients to the video server. If one client streams videos from multiple servers, then its path data is used to construct a client rooted tree to multiple video servers. Upon receiving a path data, the *locator agent* updates the regional topology accordingly. Path 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 every two consecutive nodes as an edge. *QWatch* maintains the topology graphs per Cloud region. The *locator agent* updates the *QWatch* if it discovers new nodes and edges. The *locator agent* obtains the ISP name and the AS number of a valid IP node from a commercial API [23]. Router level topology discovery has been well studied in [24] [25] and works well for *QWatch*.

E. Implementation of QoE Anomaly Localization

When the *locator agent* receives a QoE update, it processes the updates according to the *QoE Anomaly Localization Algorithm (QALA)* 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 agent* then logs the localization results and waits for the next update messages.

ALGORITHM 2: QoE Anomaly Localization Algorithm (QALA)

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

1 **while** *Receving Update* $U = (t, Q, R)$ *from a client* **do**
2 **if** $Q ==$ Acceptable **then**
3 **Get all nodes** $\{N_i\}$ **in** R
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 **else**
8 **Get all nodes** $\{N_i\}$ **in** R
9 **Initialize the number of suspect nodes as** $n^s = 0$
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's status as** $S_{N_i} =$ ***Suspect***
16 **Label** N_i **with** $L_{N_i} = (t, S_{N_i})$
17 n^s++
18 **if** $n^s == 1$ **then**
19 **Find the node** N_s **with the latest label** $L_{N_s} = (t_{N_s}, S_{N_s})$ **where** $S_{N_s} ==$ ***Suspect***
20 **Update the label for** N_s **as** $L_{N_s} = (t_{N_s}, S_{N_s})$ **where** $S_{N_s} =$ ***Abnormal***
21 **Find all nodes** $\mathbf{N_A} = \{N_a | (S_{N_a} ==$ ***Suspect***$)$ **or** $(S_{N_a} ==$ ***Abnormal***$)\}$
22 **Log anomaly event** $E_t = (t, Q, R, \mathbf{N_A})$

V. 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.

Controlled Environment Setup: The VoD system runs in 3 servers and 8 clients. The network topology of the VoD system is shown in Figure 4. Three servers are deployed in two regions of Microsoft Azure Cloud. Eight clients are deployed in 3 campus networks in PlanetLab [26]. 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.

Production Environment Setup: 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 would provide the best performance. We run 200 clients in PlanetLab to emulate users around the world.

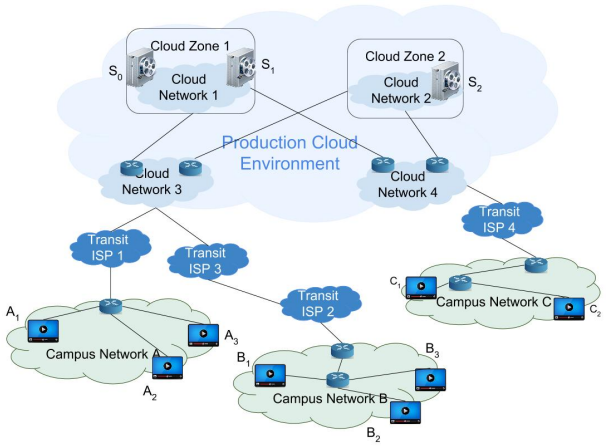


Fig. 4. The topology of the experimental VoD

We provision 5 *locator agents* in different regions in Azure Cloud to serve 200 clients at different geographical locations as shown in Figure 5. We choose $K = 100$ and provision *locator agents* in 5 available zones in Azure.

Fig. 5. The topology of the experimental VoD

VI. Evaluation of QoE Anomaly Detection

A. Evaluation of Controlled Environment

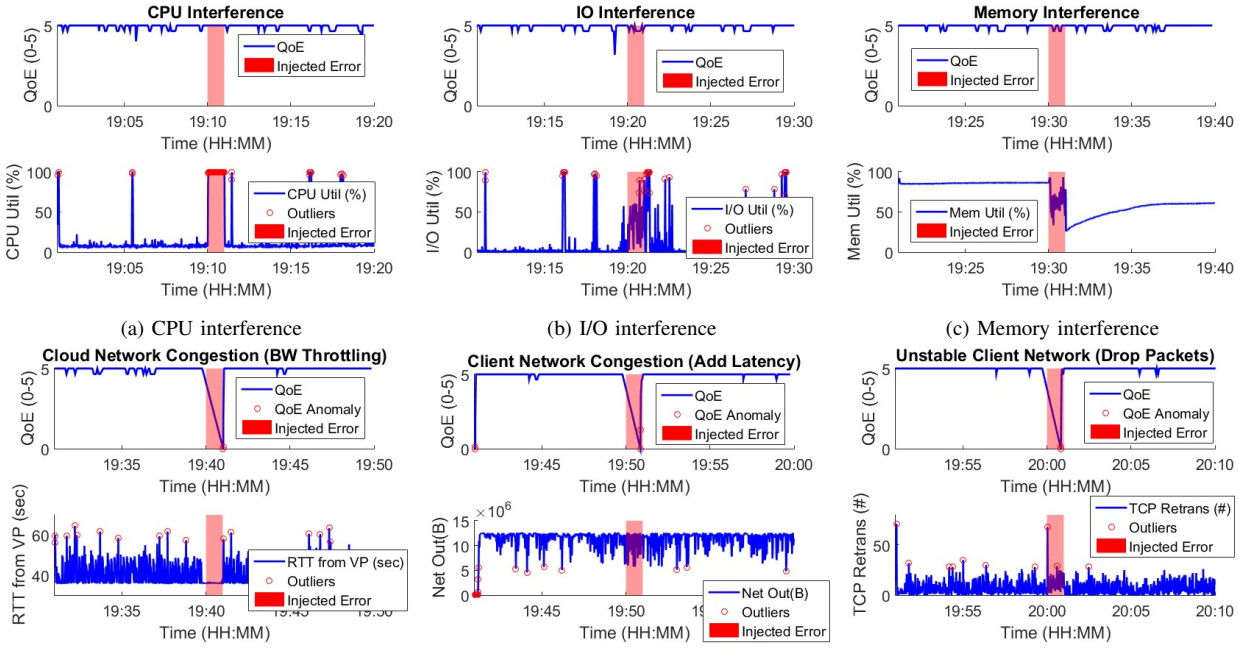
We first consider the effectiveness of system metrics, such as CPU/I/O/memory utilizations, network latency and throughput for anomaly detection in VoD in the Cloud. These system metrics can be obtained in commercial Clouds as well (e.g. AWS CloudWatch and Azure Cloud monitor). We show several examples 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 [27]. The statistical outlier is defined as data outside the range of 3 standard deviation [28]. 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, memory utilization, server outbound traffic throughput, network latency between the server and a vantage point, and the number of TCP retransmissions. All the metrics are collected by Performance Co-Pilot [29]. 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 [30] and various network errors are emulated by the Linux network emulator [31].

Figure 6 shows numerous false positives and false negatives when system metrics are used to detect QoE anomalies. Figure 6 (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 6 (b) considers I/O utilization metric with QoE anomalies. Many false positive alarms result from I/O interferences. However, I/O interferences do not impact end user QoE. Similarly in Figure 6 (c), we show a false positive case where memory interference impact memory utilization metric but have little impact on user QoE. Figure 6 (d) and (e) compare QoE anomalies with network metrics on server S_2 with network errors. Figure 6 (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 6 (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 6 (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 [32]. 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.

B. Evaluation in Production Environment

Commercial CDN providers offer their own monitoring systems. They log errors in the cache servers that could impact end user QoE. Common metrics are the HTTP response time, the edge 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 as shown in Figure 7. Figure 7 (a) shows the logged HTTP response time and detected anomalies. There are numerous anomalies detected before 00:10. Figure 7 (b) shows QoE anomalies detected by *QWatch*. Anomalies in Figure 7 (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 fail-over 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 7 (b). These anomalies are not captured by measurement in CDN as shown in Figure 7 (a). These experiments in production environment demonstrate that



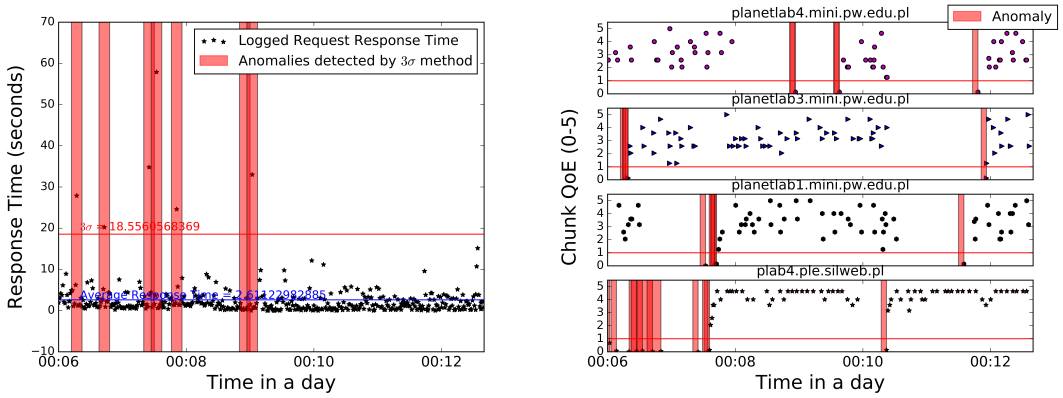
(a) CPU interference (b) I/O interference (c) Memory interference

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

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

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

Fig. 6. QoE anomaly localization in various cases



(a) Response time logged in CDN and anomalies detected

(b) QoE anomalies detected in end users

Fig. 7. CDN measurement vs. End-user QoE

there are numerous false positive and negatives in existing measurement based anomaly detection methods.

VII. Evaluation of QoE Anomaly Localization

A. Evaluation in Controlled Environment

We inject anomalies at various components including server S_1, Cloud Network 1, Campus Network A and Client A_1 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 8 shows the entire nodes involved in the experimental VoD system. Later figures only show affected components. Figure 9 (a) shows the localization results for two QoE anomalies caused by S_1. Client A_2 and B_2 are affected and their QoE degrades. A_2 and B_2 have

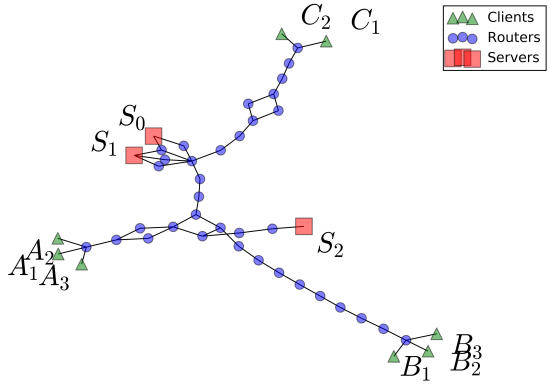
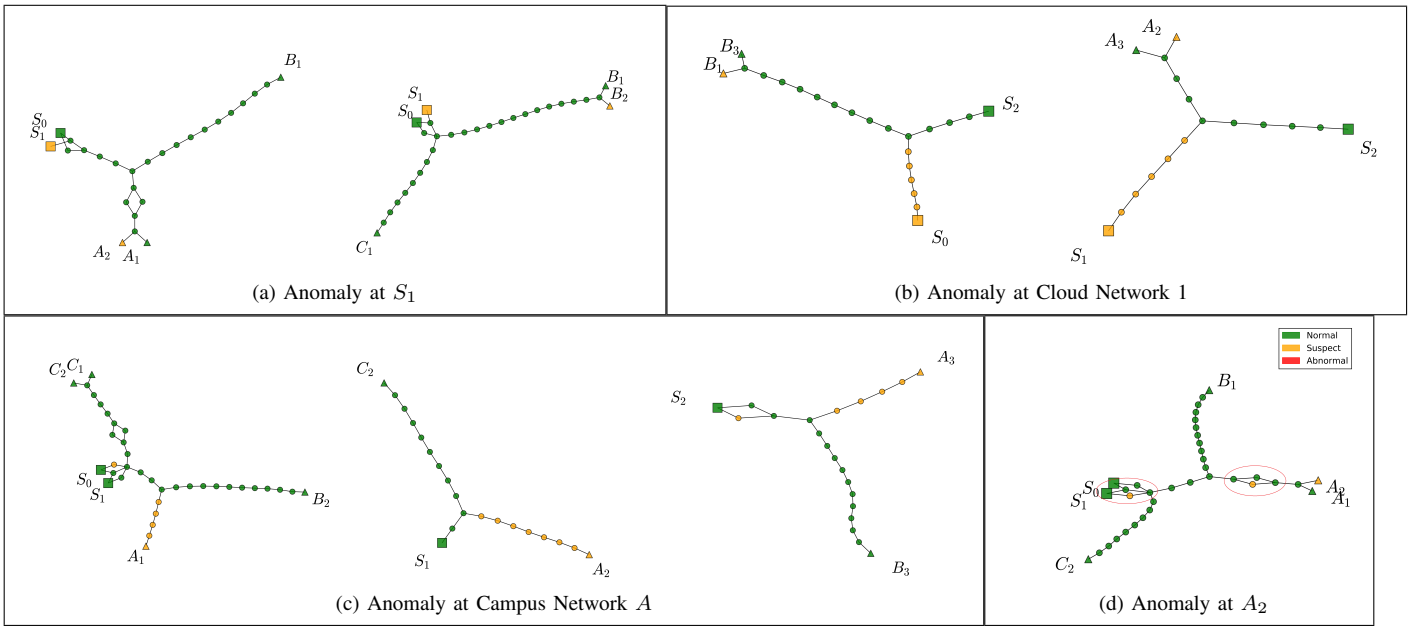


Fig. 8. Topology of experimental VoD with entire nodes

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



(a) Anomaly at S_1 (b) Anomaly at Cloud Network 1

(c) Anomaly at Campus Network A (d) Anomaly at A_2

Fig. 9. QoE anomaly localization

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 9 (b) 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 as *Suspects*. In this example, Cloud network 1 and servers connecting to Cloud network 1 are both located as *Suspects* of QoE anomalies. Further troubleshooting is needed to obtain localization with higher resolution.

An anomaly is injected in the campus network A in Figure 9 (c). Clients A_1, A_2 and A_3 connect to campus network A with QoE anomalies. *QWatch* correctly labels all nodes in the client network as *Suspects*. An anomaly is injected at client A_2 in Figure 9 (d). *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 9 (d) 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 conjecture 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.

B. 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 9. Figure 10 shows the count of QoE anomalies located in different components. The data is collected from the *locator agent* in the east US region. There are 219 QoE anomalies detected during 1 hour in the region. Figure 10 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, we suspect that Azure's algorithm is not as dynamic as we would expect. A large number of QoE anomalies are also located

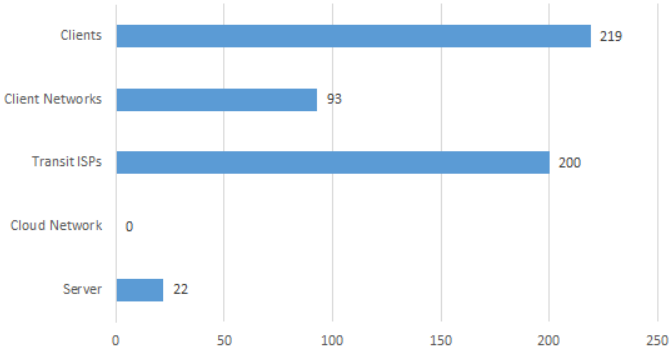


Fig. 10. QoE anomalies located at different components

in transit ISPs. Our 200 clients around the world connect to Azure CDN through different ISPs. Results show that a majority of QoE anomalies label 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.

C. Scalability Analysis

The *locator agents* are deployed on Basic A1 type VMs in Microsoft Azure. The average time to locate a QoE anomaly is around 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 to maintain 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.

VIII. Conclusion

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 only 200 users. Results show that a major of QoE anomalies are located in 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 dynamic as we would expect.

References

[1] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling netflix: Understanding and improving multi-cdn movie delivery," in *IEEE INFOCOM*, 2012.

[2] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: a platform for high-performance internet applications," *ACM SIGOPS Review*, 2010.

[3] A. A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao, "Towards automated performance diagnosis in a large IPTV network," in *ACM SIGCOMM Review*, 2009.

[4] S. K. Barker and P. Shenoy, "Empirical evaluation of latency-sensitive application performance in the cloud," in *ACM SIGMM*, 2010.

[5] P. Fiadino, A. D'Alconzo, A. Bär, A. Finamore, and P. Casas, "On the detection of network traffic anomalies in content delivery network services," in *ITC*, 2014.

[6] J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "Shedding light on the structure of internet video quality problems in the wild," in *ACM CoNEXT*. ACM, 2013.

[7] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, K. Papagiannaki, and P. Steenkiste, "Identifying the root cause of video streaming issues on mobile devices," in *ACM CoNEXT*, 2015.

[8] A. D'Alconzo, P. Casas, P. Fiadino, A. Bar, and A. Finamore, "Who to blame when youtube is not working? detecting anomalies in CDN-provisioned services," in *IEEE IWCMC*, 2014.

[9] P. Casas, A. D'Alconzo, P. Fiadino, A. Bär, A. Finamore, and T. Zseby, "When youtube does not workanalysis of qoe-relevant degradation in Google CDN traffic," *IEEE TNSM*, 2014.

[10] C. Wang, H. Kim, and R. Morla, "Users know better: A QoE based Adaptive Control System for VoD in the Cloud," in *IEEE GLOBECOM*, 2015.

[11] ——, "QoE Driven Server Selection for VoD in the Cloud," in *IEEE CLOUD*, 2015.

[12] Y. Zhao, H. Jiang, K. Zhou, Z. Huang, and P. Huang, "Meeting service level agreement cost-effectively for video-on-demand applications in the cloud," in *IEEE INFOCOM*, 2014.

[13] E. Marilly, O. Martinot, H. Papini, and D. Goderis, "Service level agreements: a main challenge for next generation networks," in *IEEE ECUMN*, 2002.

[14] I. Rec, "P. 10/G. 100 Amendment 1, New appendix I–Definition of quality of experience (QoE)," *International Telecommunication Union*, 2007.

[15] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "A quest for an internet video quality-of-experience metric," in *ACM HotNet*, 2012.

[16] M.-N. Garcia, F. De Simone, S. Tavakoli, N. Staelens, S. Egger, K. Brunnström, and A. Raake, "Quality of experience and HTTP adaptive streaming: A review of subjective studies," in *IEEE QoMEX*, 2014.

[17] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *ACM MMSys*, 2011.

[18] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *ACM MMSys*, 2011.

[19] C. Wang, H. Kim, and R. Morla, "QMan: A QoE based management system for large-scale VoD in the Cloud," *CMU Technical Report*, 2016.

[20] I. Rec, "P. 800.1, Mean opinion score (MOS) terminology," *International Telecommunication Union, Geneva*, 2006.

[21] S. Van Kester, T. Xiao, R. Kooij, K. Brunnström, and O. Ahmed, "Estimating the impact of single and multiple freezes on video quality," in *IS&T/SPIE Electronic Imaging*, 2011.

[22] P. Reichl, B. Tuffin, and R. Schatz, "Logarithmic laws in service quality perception: where microeconomics meets psychophysics and quality of experience," *Springer Telecommunication Systems*, 2013.

[23] "An ip lookup api." [Online]. Available: https://ipinfo.io/

[24] B. Huffaker, D. Plummer, D. Moore, and K. Claffy, "Topology discovery by active probing," in *IEEE SAINT*, 2002.

[25] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient algorithms for large-scale topology discovery," in *ACM SIGMETRICS Review*, 2005.

[26] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Review*, 2003.

[27] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM CSUR*, 2009.

[28] F. Pukelsheim, "The three sigma rule," *The American Statistician*, 1994.

[29] "Performance co-pilots." [Online]. Available: http://www.pcp.io/

[30] "Stress — tool to impose load on and stress test systems." [Online]. Available: http://linux.die.net/man/1/stress

[31] "Network emulator in linux traffic control facilities." [Online]. Available: http://man7.org/linux/man-pages/man8/tc-netem.8.html

[32] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, 2013.