An Experimental Evaluation of Apple Siri and Google Speech Recognition

Mehdi Assefi, Guangchi Liu, Mike P. Wittie, Clemente Izurieta
Department of Computer Science, Montana State University
(mehdi.assefi, guangchi.liu, mwittie, clemente.izurieta)@cs.montana.edu

Abstract

We perform an experimental evaluation of two popular cloud-based speech recognition systems. Cloud-based speech recognition systems enhances Web surfing, transportation, health care, etc. Using voice commands helps drivers stay connected to the Internet by avoiding traffic safety risks. The performance of these type of applications should be robust under difficult network conditions. User frustration with network traffic problems can affect the usability of these applications. We evaluate the performance of two popular cloud-based speech recognition applications, Apple Siri and Google Speech Recognition (GSR) under various network conditions. We evaluate transcription delay and accuracy of transcription of each application under different packet loss and jitter values. Results of our study show that performance of cloud-based speech recognition systems can be affected by jitter and packet loss; which are commonly occurring over WiFi and cellular network connections.

keywords: Cloud Speech Recognition, Quality of Experience, Software Measurement, Streaming Media, Real-time Systems.

1 Introduction

Performance evaluation of cloud-based speech recognition systems under different network conditions has received much less attention than other streaming systems. Although Apple Siri and Google Speech Recognition (GSR) are very popular applications that help users to interact with search engines using voice commands, an experimental evaluation of these applications is noticeably missing.

Delay and accuracy of the voice recognition process is an important parameter that affects the quality a user's experience with cloud-based speech recognition applications. Streaming voice from the client to the server and converting it to text are two phases of this process and should have the minimum possible delay in order to satisfy the quality of a user's experience. Delays of this process should also be consistent under all different network conditions.

To date, there has not been an extensive evaluation of how Siri and GSR perform under different network conditions.

In this paper, we design and implement an experimental evaluation of Siri and GSR. We evaluate these applications under different packet loss and jitter values and measure the delay of each under difficult network conditions. Specifically, we employ two models to evaluate the effects of packet loss and jitter, respectively. Each model is designed to evaluate two factors (jitter or packet loss) with one blocking variable on the response variable - delay. The blocking variable is the application (GSR and Siri), for both of the experiments. An ANOVA test is used to evaluate effects of packet loss and jitter for each experiment respectively. Results of our study show that delays in both applications are affected by packet loss and jitter.

The remainder of this paper is organized as follows. In Section II we explore related work. In Section III we describe our experimental methods. In Section IV we describe overall results. In Section V we describe our experimental design and the mathematical model used to analyze experimental data. Section VI discusses results. Finally, in Section VII we discusses threats to validity of our experiment and conclude in Section VIII.

2 Related Work

A measurement study on Google+, iChat, and Skype was performed by Yang Xu *et al.* [21]. They explored the architectural features of these applications. Using passive and active experiments, the authors unveiled some performance details of these applications such as video generation and adaption techniques, packet loss recovery solutions, and end-to-end delays. Based on

[1]A brief experimental study on Siri and Google Speech Recognition is reported in "Impact of the network performance on cloud-based speech recognition systems" in which, a solution that uses network coding to improve the performance of cloud-based speech recognition applications has been proposed. The aforementioned paper is published in ICCCN 2015 [4]. In this paper, we design and implement an extensive experimental evaluation of Apple Siri and Google Speech Recognition under different network conditions to compare the performance of these applications under different conditions.

their experiments the server location had a significant impact on user performance and also loss recovery in server-based applications. They also argued that using batched re-transmissions was a good alternative for real time applications instead of using Forward Error Correction (FEC) –an error control technique in streaming over unreliable network connections.

Te-Yuan Huang *et al.* did a measurement study on the performance of Skype's FEC mechanism [14]. They studied the amount of the redundancy added by the FEC mechanism and the trade-offs between the quality of the users' experience and also the resulting redundancy due to FEC. They tried to find an optimal level of redundancy to achieve the maximum quality of the users' experience.

Te-Yuan Huang *et al.* also performed a study on voice rate adaption of Skype under different network conditions [13]. Results of this study showed that using public domain codecs was not an ideal choice for users' satisfaction. In this study, they considered different levels of packet loss to run their experiment and came up with a model to control the redundancy under different packet loss conditions.

Kuan-Ta Chen *et al.* proposed a framework for users' QoE measurement [6]. Their proposed framework was called OneClick, and provided a dedicated key that could be pressed by users whenever they felt unsatisfied by the network conditions with streaming media. OneClick was implemented on two applications –instant messaging applications, and shooter games.

Another framework that quantified the quality of a user's experience was proposed by Kuan-Ta Chen *et al* [7]. The proposed system was able to verify participants' inputs, so it supported crowd-sourcing. Participation is made easy in this framework, and it also generates interval-scale scores. They argue that researchers can use this framework for measuring the quality of a users' experience without affecting quality of the results and achieve a higher level of diversity in users' participation while also keeping a cost low.

A delayed-based congestion control is proposed and developed by Lukasz Budzisz *et al.* [5]. The proposed system offers low standing queues and delay in homogeneous networks, and balanced delay-based and loss-based flows in heterogeneous networks. They argue that this system can achieve these properties under different loss values, and outperform TCP flows. Using experiments and analysis, they demonstrate that this system guarantees aforementioned properties.

Hayes *et al.* proposed an algorithm which tolerates non-congestion related packet loss [11]. They proved experimentally that the proposed algorithm improves the throughput by 150% under packet loss of 1% and improves the ability to share the capacity by more than 50%.

Akhshabi *et al.* proposed an experimental evaluation of rate adaption algorithms for streaming over HTTP [1, 2]. They experimentally evaluated three common video streaming applications under a range of bandwidth values. Results of this study showed that congestion control of TCP and its reliability requirement does not necessarily affect the performance of such streaming applications. Interaction of rate-adaption logic and TCP congestion control is left as an open research problem.

Chen *et al.* experimentally studied performance of multipath TCP over wireless networks [8]. They measured the latency resulting from different cellular data providers. Results of this study show that Multipath TCP offers a robust data transport under various network traffic conditions. Studying the energy costs and performance trade-offs should be considered as a possible extension of this study.

Google is currently working on a new transport protocol for the Internet which is called QUIC(Quick UDP Internet Connections) [16]. QUIC uses UDP and solves problems of packet delay under different packet loss values in TCP connections. QUIC solves this problem by multiplexing and FEC.

An experimental investigation on the Google Congestion Control (GCC) in the RTCWeb IETF WG was performed by Cicco *et al.* [9]. They implemented a controlled testbed for their experiment. Results of this experimental study show that the proposed algorithm works well but it does not utilize the bandwidth fairly when it is shared by two GCC flows or a GCC and a TCP flow.

Cicco *et al.* have also experimentally investigated the High Definition (HD) video distribution of Akamai [10]. They explained details of Akamai's client-server protocol which implements the quality adaption algorithm. Their study shows that the proposed technique encodes any video at five different bit rates and stores all of them at the server. Server selects the bit rate that matches the bandwidth that is measured based on the signal receiving from the cilent. The bitrate level adaptively changes based on the available bandwidth. Authors of the paper also evaluated the dynamics of the algorithm in three scenarios.

Winkler *et al.* ran a set of experiments to asses quality of experience on television and mobile applications [19, 20]. Their proposed subjective experiment considers different bitrates, contents, codec, and network traffic conditions. Authors of the paper used Single Stimulus Continous Quality Evaluation (SSCQE) and Double Stimulus Impairment Scale (DSIS) on the same set of materials and compared these methods and analyzed results of experiments in view of codec performance.

A mesh-pull-based P2P video streaming using Fountain codes is proposed by Oh *et al.* [15]. The proposed system offers fast and smooth streaming with low complexity. Experimental evaluations show that the proposed system has better performance than existing buffer-map-based video streaming systems under packet loss values. Considering jitter as another important factor and evaluation of behavior of proposed system considering jitter values can be a potential extension of this study.

Application of Fountain Multiple Description Coding (MDC) in video streaming over a heterogeneous peer to peer networks is considered by Smith *et al.* [17]. They conclude that Fountain MDC codes are favorable in such cases, but there are some restrictions in real-world P2P streaming systems.

Finally, Vukobratovic *et al.* proposed a novel multicast streaming system that is based on Expanding Window Fountain (EWF) codes for real-time multicast [18]. Using Raptor-like precoding has been addressed as a potential improvement in this area.

3 Experimental Testbeds

We design and implement our experimental testbed to study the performance of Apple Siri and GSR under loss and jitter. Clients transmit voice data through a network traffic shaper, in which is we change jitter and packet loss values in the communication network. We set a bandwidth to 2Mbps which is typical on 3G connections [12]. The server receives voice data, translates the voice into text, and sends the text and search results based on the converted text to the client. The client calculates the delay of the server response. To calculate the accuracy of transcription we use Levenshtein distance [22]. Accuracy is measured as the match percentage of the original string used to generate the voice and the resulting transcription. The client uses Wireshark Version 1.12.4 to timestamp the traffic of voice transmission to and from the server [3]. We developed a Windows application using Visual C# to timestamp the voice playback. All experiments are performed on a Windows 7 platform for GSR, and on iOS 7.0 for Siri. The traffic shaper is a `netem` box which runs the Fedora Linux operating system. We ran our experiment 30 times for each value of loss and jitter and for each cloud speech recognizer.

3.1 Experimental Testbed for GSR

We use the GSR service available in Google Chrome. There is also another alternative for using Google voice recognition. Google offers a voice recognition Web service that can be used in Windows applications. Figure 1 shows the architecture of our experimental setup.

Clients transmit voice packets to the Google server through the `netem` box that changes network traffic performance. We used a recorded voice with a length of 26.4 seconds for all experiments in order to have a consistent measurement. Google starts to recognize voice as soon as it receives the first voice packet, and sends converted text back to the client. The client records the time of each packet and also voice transmission time to calculate the transcription time of the experiment. The client also compares the resulting text to the original string; which was used to generate the voice command and calculates transmission accuracy using the Levenshtein distance [22].

3.2 Experimental Testbed for Siri

The experimental setup for Siri is similar to GSR. We use an iPhone as the client. A client is connected to the Internet through a WiFi router then to a `netem` box. Here we also used Wireshark to timestamp the transmission of voice packets and reception of results from the Siri server. Figure 2 depicts this setup.

4 Overall Results

To investigate the effect of packet loss and jitter on delay and accuracy, we generate packet loss from 1% to 5% and jitter from 20 ms to 200 ms respectively on our testbeds and observe the resulting accuracy and delay. Siri and GSR both keep 100% accuracy under high values of packet loss and jitter, so we just consider delay values in the rest of our study. Overall results are shown in Figures 3 to 6, where the y axis displays delay(s), and the x axis displays packet loss (percentile) and jitter (ms), respectively. There are increasing trends as packet loss and jitter increases, for both Siri and GSR. For GSR, an increase of 1 packet loss unit (percentile), leads to delay increases in the range of 0-100 ms. An increase of 1 unit (20 ms) in packet loss leads to increases in delay from 0-100 ms. In addition, the variance of delay also increases as packet loss and jitter increase, indicating a trend of instability. For Siri, the increase in 1 unit (percentile) packet loss leads to increases in delay of 200 ms; which is worse than GSR. On the other hand, jitter has less impact on delay. In addition, the variance of delay is unchanged, compared to GSR.

5 Experiment Design

We evaluate our results and data using mathematical models and an ANOVA test. Our response variable is delay of transcription and our factors are loss and jitter.

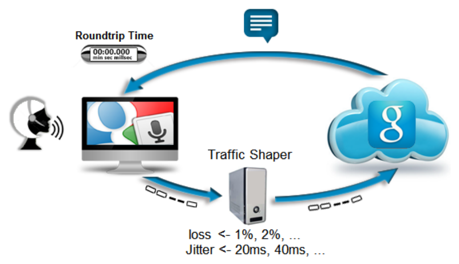

Figure 1: Experimental testbed for GSR.

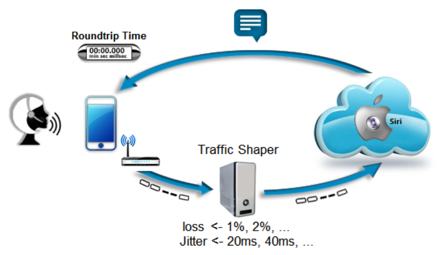

Figure 2: Experimental testbed for Siri.

5.1 Model

Since data is collected by varying jitter and packet loss respectively, we designed two models to assess the effect of jitter and packet loss on delay. Also, since data is collected from two applications (i.e., Siri and GSR), we treat the application as a blocking variable. Hence, we set up two models for jitter and packet loss respectively. Each model contains one factor and one blocking variable. For the first model, the response variable is delay, the independent variable is jitter and the blocking variable is application. Also, to guarantee that the assumptions still hold for the following ANOVA tests, we apply a logarithmic transformation on the response variable. Hence, the first model can be expressed as:

$$log(y_{ij}) = \mu + \alpha_i + \beta_j + e_{ij} \tag{1}$$

where α is the jitter, and β represents the application. Similarly, the second model can be expressed as:

$$log(y_{ij}) = \mu + \gamma_i + \beta_j + e_{ij} \tag{2}$$

where γ is the jitter, and β represents the application.

For model 1, the factor (jitter) has 10 alternatives; which are the jitter duration values ranging from 20 to 200 ms. For model 2, the factor (packet loss) has 5 alternatives; which are the proportion of lost packets ranging from 1% to 5%. The blocking variable for both models has 2 alternatives; which are GSR and Siri, respectively.

5.2 Assumption of Normality Check

Some assumptions should be checked before conducting the ANOVA tests. In this Section, the interaction of independent variables, the normality of errors and the constant variance of errors are tested for normality.

We first test the interaction between factors. In Figures 7 and 8, the errors (residuals) give us confidence that they are constantly distributed as the fitted values change, indicating that the interactions between the blocking variable and factor are trivial for both jitter (Eq. 1) and packet loss models (Eq. 2).

Secondly, the error variance of each model also appears constant. Figures 9 through 12 show that errors (residuals) appear constant as the independent variables (jitter/packet loss and application) change, indicating that the error of models 1 and 2 are constant.

Finally, figures 13 and 14 show that the error distributions for model 1 and model 2 are normal, indicating that the assumption of normally distributed errors holds for both of the models. In summary, all the the assumptions for conducting an ANOVA test hold for both models (Eq. 1 and Eq. 2).

Table 1: Statistical Findings of Jitter and Packet Loss

Jitter	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(<F)**
Jitter	9	1.013	0.113	34.27	<2e-16
App	1	7.77	7.77	2364.79	<2e-16
errors (residuals)	177	0.582	0.003	–	–
Packet Loss	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(<F)**
Packet Loss	4	1.056	0.264	27.66	<2e-16
App	1	17.025	17.025	1782.7	<2e-16
errors (residuals)	135	1.289	0.010	–	–

6 Results

Table 1 provides conclusive evidence that roundtrip delay of GSR and Siri are affected by both jitter (p-value = 2e-16, f-value =34.27 on 9 df.) and packet loss (p-value = 2e-16 , f-value =27.66 on 4 df.). Jitter causes packets to arrive out of order and TCP needs to reorder packets before delivering them to the application layer. TCP also re-transmits lost packets. Both packet loss and jitter reduce the voice stream quality and this affects the performance of the speech recognition.

The application, on the other hand, affects the delay much more seriously. Specifically, the f-values of application for jitter and packet loss are 1782.7 and 2364.79 on 1 df., respectively. To test the difference between Siri and GSR on delay, we also employ a Walch t-test to compare the samples obtained from Siri and GSR. The mean difference between Siri and GSR is 1.666s, with 95% confidence interval from -1.736 to -1.600, (p-value $<$ 2.2e-16). This suggests that there

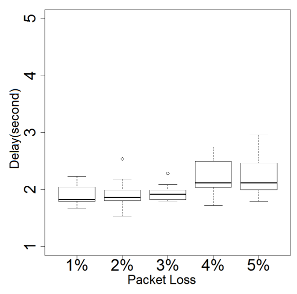

Figure 3: Impact of packet loss on delay of GSR

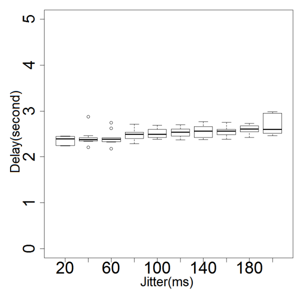

Figure 4: Impact of jitter on delay of GSR

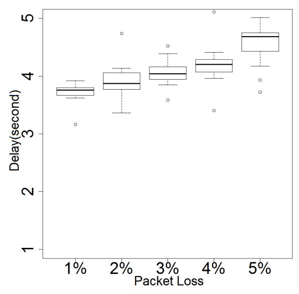

Figure 5: Impact of packet loss on delay of Siri

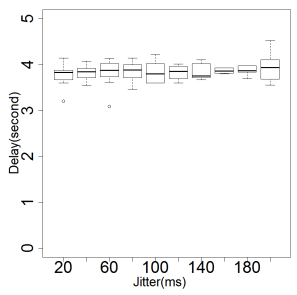

Figure 6: Impact of jitter on delay of Siri

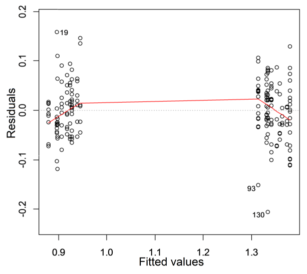

Figure 7: Jitter: Fitted Values vs. Errors (Residuals)

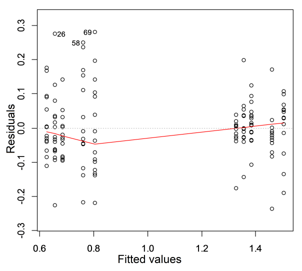

Figure 8: Packet Loss: Fitted Values vs. Errors (Residuals)

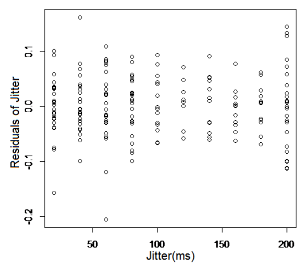

Figure 9: Jitter: Fitted Values vs. Errors (Residuals)

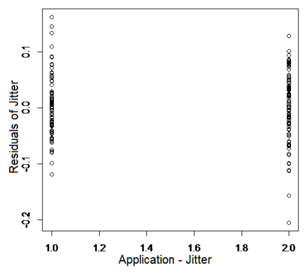

Figure 10: Application (Jitter): Fitted Values vs.Errors (Residuals)

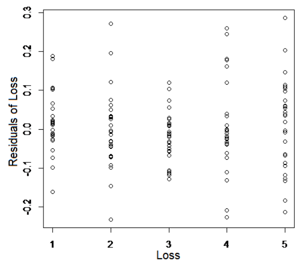

Figure 11: Packet Loss: Fitted Values vs.Errors (Residuals)

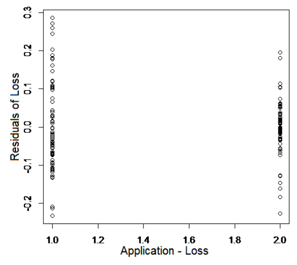

Figure 12: Application (Packet Loss): Fitted Values vs.Errors (Residuals)

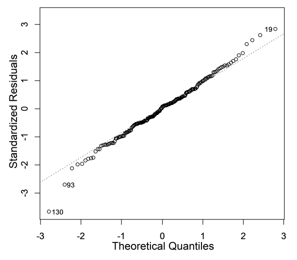

Figure 13: Normality of Jitter

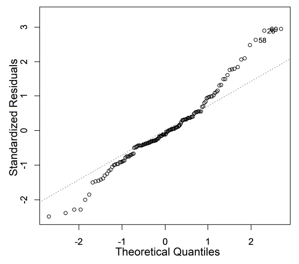

Figure 14: Normality of Packet Loss

exists a statistically significant difference between GSR and Siri. By examining the response variable for each application separately, Siri causes much more delay than GSR. This is because the algorithm employed by Siri keeps the resulting text accurate by starting the speech recognition process just after receiving the voice date. That means Siri needs to receive the entire voice stream before starting to generate the text. As a result, this increases the delay in processing the whole text and accounts for the majority of the total delay. GSR, on the other hand, keeps the result accurate by adaptively adjusting the transport and application layers and so it offers less delay even under high values of packet loss and jitter compared to Siri.

7 Threats to Validity

7.1 Threats to Internal Validity

One of the possible threats to internal validity is the hardware limitations of the devices running GSR and Siri. More specifically, the processing speed of memory and CPU will affect the processing of data streams in a PC. Another possible threat is the status of the PC. For example, when the OS is busy, it does not have enough time to respond to the interruptions generated from GSR or Siri, hence generating and thus affecting delay.

7.2 Threats to External Validity

All of the experiments were conducted in our lab and through our campus network. It is likely that the configuration of our campus network is different from other networks, such as firewalls and TCP/UDP controls. Hence, the conclusion obtained from the experiment cannot be generalized to common network environments. In addition, the available bandwidth of different regions in United States is different. It is possible that this diversity affects the conclusion that it cannot be applied to the other regions in United States. Finally, the sample is small (the evaluation is run on one desktop in a laboratory setting). A larger scale experiment running on more desktops, as well as laptops and smart phones, will lessen external threats.

7.3 Threats to Construct Validity

Since the delay generated by the Internet (e.g., router, DNS, etc.) is complicated and unpredictable, it is hard to say the extent to which packet loss and jitter impact delay. Also, the transportation and routing layers employ self-adaptive mechanisms to adjust the performance of specific applications, e.g., GSR and Siri. In the end, both the jitter and the packet loss are generated by a specific program (i.e., simulated), rather than real network conditions. It is hard to know whether the simulated impact has the same effects of real jitter or packet loss.

8 Conclusions and Future Work

We designed and implemented experimental evaluations of Siri and GSR. Using the collected data from our experiments, we designed two models to evaluate the effects of jitter and packet loss separately. After conducting ANOVA tests for each experiment, we found that the effects of packet loss and jitter on delay are statistically significant but the impact is not important compared to the one that comes from the application, because from the table we can see that the application generated most of the impact. In addition, we found that GSR performs better than Siri when measuring delay.

Delay of both applications is affected by packet loss and jitter. In order to design and implement real-time cloud speech recognition applications for more critical tasks, there should be mechanisms to measure loss/jitter tolerant systems. Network coding is a possible solution which can be used to reduce the effect of packet loss and jitter [4, 15, 17, 18]. Using TCP keeps these applications accurate under packet loss and jitter values, but as we saw in our results, it affects the roundtrip delay. By using UDP and network coding, we can keep the system accurate under different values of jitter and packet loss while we reduce the resulting delay. Future cloud based speech recognition applications that use cellular networks are still required to overcome this problem; which is due to the presence of jitter from packet transmission over different paths.

This experiment can also be extended by running Siri and GSR over different cellular networks, and adding the celluar data provider as another blocking variable.

Running the experimental setup over a wide geographical range of clients and also using different cellular data providers can result in more accurate results. Considering clients with a diversity of hardware and software configurations can be another extension for this research.

References

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

[2] Saamer Akhshabi, Sethumadhavan Narayanaswamy, Ali C Begen, and Constantine Dovrolis. An experimental evaluation of

rate-adaptive video players over http. *Signal Processing: Image Communication*, 27(4):271–287, 2012.

[3] Jay Beale Angela Orebaugh, Gilbert Ramirez and Joshua Wright. Wireshark and ethereal network protocol analyzer toolkit. *Syngress Media Inc*, 2007.

[4] Mehdi Assefi, Mike P. Wittie, and Allan Knight. Impact of network performance on cloud speech recognition. *ICCCN*, IEEE. Aug. 2015.

[5] Ł Budzisz, Rade Stanojević, Arieh Schlote, Fred Baker, and Robert Shorten. On the fair coexistence of loss-and delay-based tcp. *IEEE/ACM Transactions on Networking (TON)*, 19(6):1811–1824, 2011.

[6] Kuan-Ta Chen, Cheng-Chun Tu, and Wei-Cheng Xiao. Oneclick: A framework for measuring network quality of experience. In *INFOCOM 2009, IEEE*, pages 702–710. IEEE, 2009.

[7] Kuan-Ta Chen, Chen-Chi Wu, Yu-Chun Chang, and Chin-Laung Lei. A crowdsourceable qoe evaluation framework for multimedia content. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 491–500. ACM, 2009.

[8] Yung-Chih Chen, Yeon-sup Lim, Richard J Gibbens, Erich M Nahum, Ramin Khalili, and Don Towsley. A measurement-based study of multipath tcp performance over wireless networks. In *ACM IMC*. ACM, 2013.

[9] Luca De Cicco, Gaetano Carlucci, and Saverio Mascolo. Experimental investigation of the google congestion control for real-time flows. In *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, pages 21–26. ACM, 2013.

[10] Luca De Cicco and Saverio Mascolo. *An experimental investigation of the Akamai adaptive video streaming*. Springer, 2010.

[11] David A Hayes and Grenville Armitage. Improved coexistence and loss tolerance for delay based tcp congestion control. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 24–31. IEEE, 2010.

[12] Junxian Huang, Qiang Xu, Birjodh Tiwana, Z Morley Mao, Ming Zhang, and Paramvir Bahl. Anatomizing application performance differences on smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010.

[13] Te-Yuan Huang, Kuan-Ta Chen, and Polly Huang. Tuning skype's redundancy control algorithm for user satisfaction. In *INFOCOM 2009, IEEE*, pages 1179–1187. IEEE, 2009.

[14] Te-Yuan Huang, Polly Huang, Kuan-Ta Chen, and Po-Jung Wang. Could skype be more satisfying? a qoe-centric study of the fec mechanism in an internet-scale voip system. *Network, IEEE*, 24(2):42–48, 2010.

[15] Hyung Rai Oh and Hwangjun Song. Mesh-pull-based p2p video streaming system using fountain codes. In *Computer Communications and Networks (ICCCN)*. IEEE, Jul. 2011.

[16] J. Roskind. *QUIC: Design Document and Specification*, Dec. 2013. `https://docs.google.com/a/chromium.org/document/d/1RNHkxVvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34`.

[17] Guillaume Smith, P Tournoux, Roksana Boreli, Jérôme Lacan, and Emmanuel Lochin. On the limit of fountain mdc codes for video peer-to-peer networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, Jun. 2012.

[18] Dejan Vukobratovic, Vladimir Stankovic, Dino Sejdinovic, Lina Stankovic, and Zixiang Xiong. Scalable video multicast using expanding window fountain codes. *IEEE Transactions on Multimedia*, 11(6):1094–1104, Oct. 2009.

[19] Stefan Winkler and Ruth Campos. Video quality evaluation for internet streaming applications. In *Electronic Imaging 2003*, pages 104–115. International Society for Optics and Photonics.

[20] Stefan Winkler and Frédéric Dufaux. Video quality evaluation for mobile streaming applications. In *Visual Communications and Image Processing 2003*, pages 593–603. International Society for Optics and Photonics, 2003.

[21] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. Video telephony for end-consumers: measurement study of google+, ichat, and skype. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 371–384. ACM, 2012.

[22] Li Yujian and Liu Bo. A normalized levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095, Jun. 2007.