"Can you SEE me now?" A Measurement Study of Mobile Video Calls

Chenguang Yu, Yang Xu, Bo Liu and Yong Liu
Department of Electrical and Computer Engineering
Polytechnic Institute of New York University
Brooklyn, New York, 11201
Email: {cyu03, yxu10, bliu01}@students.poly.edu, yongliu@poly.edu

***Abstract*—Video telephony is increasingly being adopted by end consumers. It is extremely challenging to deliver video calls over wireless networks. In this paper, we conduct a measurement study on three popular mobile video call applications: FaceTime, Google Plus Hangout, and Skype, over both WiFi and Cellular links. We study the following questions: 1) how they encode/decode video in realtime under tight resource constraints on mobile devices? 2) how they transmit video smoothly in the face of various wireless network impairments? 3) what is their delivered video conferencing quality under different mobile network conditions? 4) how different system architectures and design choices contribute to their delivered quality? Through detailed analysis of measurement results, we obtain valuable insights regarding the unique challenges, advantages and disadvantages of existing design solutions, and possible directions to deliver high-quality video calls in wireless networks.**

I. Introduction

Video telephony is increasingly being adopted by end consumers. Video calls augment voice calls with live visual interaction between users. Just as they shift from fixed landline phones to mobile phones for voice calls, users prefer to make *untethered* video calls using their mobile devices, e.g., smartphones, tablets and laptops, instead of sitting in front of their desktop computers. Mobile devices are connected to the Internet through either WiFi or Cellular networks, which are known to be much more heterogeneous and volatile than wireline networks. It is already very challenging for wireless service providers to deliver universal high-quality voice calls to all their customers. All providers strive to assure their customers that they have a satisfactory answer to the basic voice call quality question–*"Can you hear me now?"*, through their services, as well as commercials. Due to its higher bandwidth requirement, it is much more challenging to deliver high-quality video calls than voice calls. Even though video calls are already very popular on various mobile platforms, there is very limited understanding about the answer to the basic video call quality question – *"Can you SEE me now?"*

Towards obtaining more understanding, we conduct a measurement study on three most popular mobile video call applications: FaceTime, Google Plus Hangout, and Skype, in both WiFi and Cellular networks. Our study is focused on the following questions:

- *how they encode and decode video in realtime under tight resource constraints on mobile devices, such as battery, CPU, and screen size?*
- *how they transmit the encoded video smoothly in face of various wireless network impairments, including bursty loss, highly variable delay and competing cross-traffic?*
- *what is their delivered video conferencing quality, both video perceptual quality and video delay, under different real mobile network conditions?*
- *how different system architectures and design choices adopted by each system contribute to user-perceived video conferencing quality?*

To answer those challenging questions, we leverage our previous measurement study of computer-based video conferencing systems [20]. We extend the active and passive measurement methodologies developed in [20] to work with mobile devices. To account for the inherent heterogeneity and volatility of wireless networks, we measure the three systems under a wide range of network conditions, including a controlled network emulator, a campus WiFi network with strong and weak signal receptions, and a Cellular network of one top-three US carrier with and without user mobility. We collect extensive measurement traces at packet level and video level. Through analysis of measurement results, we obtain valuable insights regarding the unique challenges, advantages and disadvantages of the existing design solutions, and possible new directions to deliver high-quality video calls in wireless networks. Our findings are summarized as following.

1) With a strong WiFi/Cellular connection, modern smartphones are capable of encoding, transmitting and decoding high quality video in realtime;
2) Mobile video call quality is highly vulnerable to bursty packet losses and long packet delays, which are sporadic on wireless links with weak receptions;
3) While FEC can be used to recover random packet losses, the inability to differentiate congestion losses from random losses can trigger vicious congestion cycle, and significantly degrade user video call experience;
4) Conservative video rate selection and FEC redundancy schemes often lead to better video conferencing quality, compared with more aggressive schemes;
5) End-to-end video delay is highly correlated to end-to-end packet delay in Cellular networks, regardless of the

978-1-4799-3360-0/14/$31.00 © 2014 IEEE

signal strength.

The rest of the paper is organized as follows. We briefly discuss the related work in Section II. Our measurement platform and methodology are introduced in Section III. The key design choices of the three systems are presented in Section IV. In Section V, we first present the delivered video call quality of each system under different wireless environments. We then study how the quality of each system is affected by network impairments, and compare the efficiency and robustness of different design choices made by each system. The paper is concluded in Section VI.

II. Related Work

There are lots of measurement studies on WiFi networks, e.g., [3], [12], and Cellular networks, e.g., [19], [11], [8]. There are also some studies to compare the performance of WiFi and Cellular networks [1], [18], [6]. They concluded that WiFi provides higher download rate and smaller latency than 3G. Most recently, Huang *et al.*[9] studied the performance and power characteristics of 4G LTE networks. They observed that LTE can offer higher downlink and uplink throughput than 3G and even WiFi. Different from those studies, we focus on the performance of video calls over WiFi and Cellular networks. Most measurement studies of realtime communications over the Internet focused on Skype's VoIP service. Baset *et al* [2] analyzed Skype's P2P topology, call establishment protocol, and NAT traversal mechanism. Skype's FEC mechanism was studied in [10], [22]. In [4], Huang *et al.* proposed a user satisfaction index model to quantify VoIP user satisfaction. More recently, there are some measurement studies on video calls. Cicco *et al.*[5] measured the responsiveness of Skype video calls to bandwidth variations. In [23], we conducted a measurement study of Skype two-party video calls under different network conditions. In [20], we measured three computer-based multi-party video conferencing solutions: iChat, Google+ hangout, and Skype. Different from those studies, we focus on mobile video calls over wireless networks in this paper.

III. Measurement Platform

FaceTime, Skype, and Google+ Hangout all use proprietary protocol and encode their signaling and data. Using methodology similar in [20], we measure them as "black boxes", reverse-engineer their design choices, and compare their performance in wireless networks. We performed IP-level packet sniffing, application-level information window capturing, and video-level quality analysis. Among the three, only Google+ offers multi-party conferencing feature on mobile platforms. We therefore restrict our comparison study to two-party video calls.

A. Testbed

1) Overall Platform: Our measurement platform (shown in Fig. 1) consists of two parts: wireless user side and wireline user side. At the wireless user side, a smartphone is connected to the Internet through WiFi or 3G/HSPA cellular data service

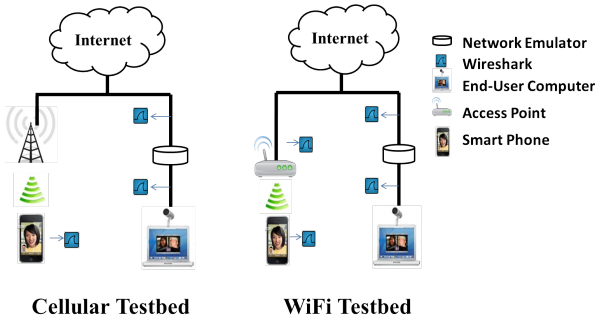


Fig. 1. Testbeds for Mobile Video Call Measurement

provided by one top-three US carrier. At the wireline user side, a PC or Mac is connected to the Internet through campus Ethernet. Software-based network emulators are inserted on both ends of the connection to emulate network conditions in controlled experiments. Packet traces are captured at different points using Wireshark. Experimental video calls are established between the smartphone and the computer. To emulate a consistent and repeatable video call, we choose a standard TV news video sequence "Akiyo" from JVT (Joint Video Team) test sequence pool. The sequence has mostly head and shoulder movements. It is very similar to a video-call scenario. In order to inject this video sequence into the video call systems, at the computer side, we use a virtual video camera tool [7]. Since we cannot find a virtual camera tool for our smartphone, we simply focus the smartphone's camera to a screen displaying the "Akiyo" video.

2) Smartphone Hacks: Since Facetime is only available on Apple devices, we have to use iPhone for our experiments. The *integrated/closed* hardware and software design of iPhone significantly increases the measurement difficulty. We have to go through several stages of hacks to prepare the smartphone for measurement study.

- *Privilege escalation*: We use an unlocked iPhone 4S, which underwent jailbreak to allow third-party application installation [16];
- *Measurement tool installation*: Several tools are downloaded from multiple sources that allow us to collect CPU and network statistics, capture packets and screen, enable ad-hoc connection, and remotely login to iPhone, etc. Please see our technical report [21] for a complete list of tools and their usages.
- *Remote phone control*: All three video call applications are by default on single task mode, which means if we switch to another application, the ongoing call will be put on hold and the video will freeze. In order to run `tcpdump` or `ping` on the iPhone during a video call, we use a computer or another Android phone to remotely login and control the iPhone via SSH, which will not interrupt the video call. For experiments without mobility, we connect the phone to a computer via a USB cable. Then we use a tool *iTool* on the computer to build a USB tunnel between the *localhost* and the iPhone, which allow us to access the iPhone from the computer. For experiments on a moving train, we control the iPhone

using another Android phone (Google Nexus 4). We set the iPhone to WiFi ad-hoc mode with the help of *MyWi*. Then the Android phone can access and control the iPhone via SSH over ad-hoc WiFi.

3) Wireless Network Settings: To test application performance under different wireless conditions, we conduct WiFi and Cellular experiments at several locations with strong and weak signal receptions. We use `SpeedTest` to verify the actual upload throughput. In WiFi experiments, upload throughput at strong signal locations is above 10 mbps, and around $200-300$ kbps at weak signal locations; in Cellular experiments, upload throughput at strong signal locations is above $1,000$ kbps, and at weak signal locations is around $100-150$ kbps. To test video call performance with mobility, we also conducted experiments on subway trains when they run on the ground. For controlled experiments, we use a software network emulator, NEWT [13], to emulate a variety of network attributes, such as propagation delay, random packet loss, and available bandwidth.

B. Information Collection

We collect measurement from multiple channels.

1) *IP Packet Traces:* We sniff packets at both the computer side (with *wireshark*) and the smartphone side (with command line `tcpdump`). Collected packet traces are used to analyze protocols and calculate network level statistics, such as packet delay, loss, and loss burst, etc.
2) *Video Quality Logs:* At the computer side, Skype and Google+ report technical information about the received video quality through their application windows, such as video rates, frame rates, RTT, et al, [15]. We use a screen text capture tool [17] to capture these information periodically. The sampling interval is 1 second.
3) *End-to-End Video Delay Samples:* Same as in our previous work [20], we use end-to-end video delay as an important measure of video call quality. End-to-end video delay is defined as the time lag from when a video frame is generated on the sender side till it is displayed on the receiver's screen. It consists of video capturing, encoding, transmission, decoding, and rendering delays.

As illustrated in Fig. 2, to measure the one-way video delay of a video call, we put computer A and phone B close to each other. The "Akiyo" video is being played on computer A. Meanwhile, a stopwatch application is also running on A. We then start the video call between A and B, with the camera of B focused on the "Akiyo+Stopwatch" video on A's screen. Through the video call application, phone B sends the captured "Akiyo+Stopwatch" video to A. On computer A, we put the received video window next to the original source video window. By comparing the readings from the two stopwatch videos on computer A's screen, we can get the one-way video delay from phone B to computer A. To automatically collect video delay information from the two stopwatches, we write a script to capture the screen of computer A once every 100 millisecond, and then decode the captured stopwatch images using an Optical Character Recognition (OCR) software.

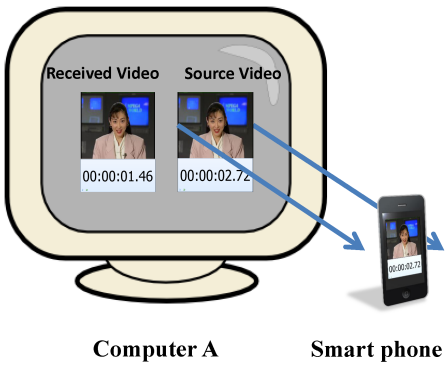


Fig. 2. One-way video delay testbed.

When the phone and computer are not in the same location, e.g., in our mobility experiments on subway, we cannot measure the one-way delay as in Fig. 2. Instead, we can measure the *round-trip video delay* using the scheme illustrated in Fig. 3. There are totally five stopwatch videos during a video

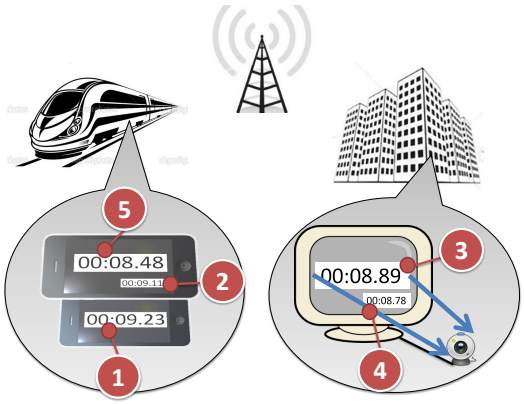


Fig. 3. Round-trip video delay testbed on subway.

call. Stopwatch ① is a stand-alone application running on a separate Android phone. During a video call, the iPhone captures the video of stopwatch ①, the captured video is marked as stopwatch ② on the iPhone screen. The captured stopwatch video is then sent to the receiving computer, on which it is displayed as stopwatch ③. After this, the receiver focuses its own camera to the received stopwatch ③ on its own screen, and the captured video, marked as ④, is sent back to the iPhone through the video call system. The iPhone finally displays the video received from the computer on its screen, marked as ⑤. Round-trip video delay can be calculated as the reading difference between stopwatch ② and stopwatch ⑤. Similar to the one-way delay measurement, by periodically capturing the iPhone screen using a script and analyzing the captured images using OCR, we can collect a large number of round-trip video delay measurements.

IV. How Do They Work - Key Design Choices

We first need to understand the three systems' design choices on system architecture, video generation & adaptation, and packet loss recovery, etc. Leveraging on our study in [23], [20] for their corresponding computer versions, we

are able to discover important design choices of Google+ and Skype's mobile versions. We also obtain good understanding on FaceTime for the first time.

A. Architecture and Protocol

Similar to its computer version, Google+ mobile version is also *server-centric*. Our mobile phone is always connected to a Google conferencing server located close to New York City, with RTT of 14ms to a computer in our campus network. There is no direct communication between the phone and the computer in all our experiments. Google+ uses UDP and only switches to TCP if we deliberately block UDP traffic. All the voice and video data are encapsulated in RTP packets.

Skype mobile is still *hybrid*: sometime our mobile phone connects to the computer directly (mostly when using WiFi), sometime it routes the video call through a relay server (mostly when using Cellular). At the transport layer, Skype uses UDP or TCP. Compared with the computer version, Skype mobile is more likely to use TCP and relay server. This might because it is more complicated to establish a direct connection between mobile devices. Instead of RTP, Skype uses its own protocol to encapsulate voice and video. Skype relay servers are at different locations with RTTs to our campus network ranging from 4 to 37 ms.

In our WiFi experiments, FaceTime mostly uses direct P2P connection between the smartphone and the computer. In our Cellular experiments, the smartphone and the computer are connected through relay servers, with RTTs to our campus network ranging from 2ms to 20ms. FaceTime always use UDP, no video call can be established if we block UDP traffic.

TABLE I
SYSTEM ARCHITECTURE AND PROTOCOL COMPARISON

System	P2P or Server	UDP or TCP	RTP
Skype	Relay server or P2P	UDP, may use TCP	No
Google+	Server-centric	UDP, only use TCP when UDP is blocked	Yes
FaceTime	WiFi: mostly P2P Cellular: relay server	Always use UDP	Yes

B. Video Encoding

Network conditions, such as available bandwidth, packet loss and delay, are inherently dynamic, in wireless environment. To meet the tight video playout deadline, only very limited receiver-side buffering is allowed to smooth out bandwidth variations and delay jitters. To maintain a smooth video call, the source has to adapt its video encoding strategy to network conditions. All three systems are capable of generating video at different rates in realtime. We probe their video encoding parameter ranges by throttling the end-to-end bandwidth from the smartphone to the computer, using the network emulator. On the computer side, both Skype and Google+ report total rate, frame rate and resolution of the video received from the smartphone in an application information window. Their video encoding parameter ranges are reported in Table II. Using the same RTP header analysis technique introduced in [20], we verified that Google+ still uses layered video coding on mobile phones. Both temporal and spatial scalability are used to generate video in a wide rate range. It is well-known that layered video coding is computation-intensive. In our experiments with Google+, the iPhone CPU utilization is close to 100%, 50% higher than FaceTime and Skype. Google+ also consumes 40% more power than FaceTime and Skype. Please see our technical report [21] for detailed CPU and battery consumption statistics. It is still amazing that Google managed to implement realtime layered video coding on mobile phones.

TABLE II
VIDEO RATE RANGES AND ENCODING PARAMETERS

System	Range (kbps)	Resolution	FPS
Skype	10 - 620	480*360, 320*240	1-12
Google+ Hangout	20 - 800	480*360, 240*180	1-15
FaceTime	10 - 820	??*??	1-30

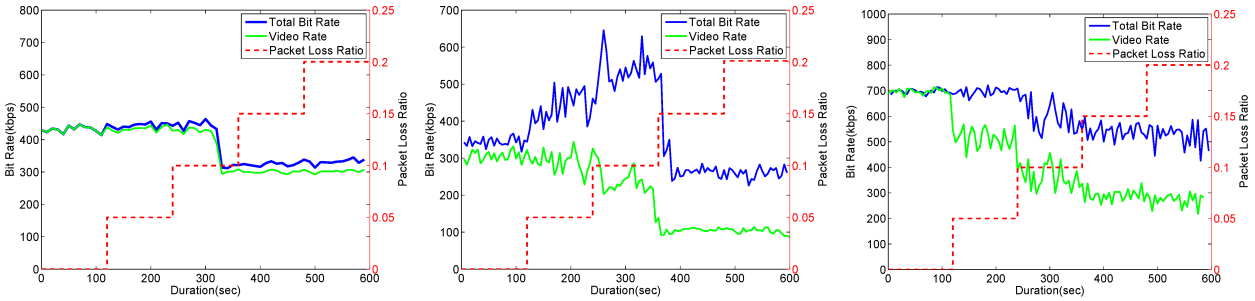
Unfortunately, FaceTime reports very limited information about its video encoding parameters. We derive FaceTime's video rate from the captured video trace, by discounting FEC packets. (We will describe how we identify FaceTime's FEC packets in the following section.) To estimate its frame rate, we first calculate the timestamp difference of two adjacent RTP packets. If the RTP timestamp difference is zero, they are from the same video frame. Let Δ be the minimum non-zero timestamp difference. Any packet pair with timestamp difference Δ must be from two adjacent video frames. We then use the difference t_c of the packet capture time of the pair to approximate the gap between the generation time of their corresponding frames. Then the frame rate can be estimated as $1/t_c$. The inferred frame rate ranges from 1 to 30 FPS.

C. Loss Recovery

Wireless networks have both congestion losses and random losses. To cope with losses, Skype, Google+ and Facetime all use redundant data to protect video data. To gain more insights about their loss recovery strategies, we conduct controlled experiments and systematically inject random packet losses to path from the smartphone to the computer. As illustrated in Figure 4, we started with zero loss rate, then increase loss rate by 5% every 120 seconds. We record the video rate and sending rate of each system.

As indicated in Figure 4(a), Google+'s total sending rate is only slightly higher than its actual video rate. This is consistent with our finding in [20] for Google+ computer version, which selectively retransmits lost packets. Through RTP packet analysis, we verified that Google+ mobile version also employs *selective retransmission*: lost video packets from the base layer will be retransmitted, and lost packets from the upper layers may not be recovered. We showed in [20] that Google+'s retransmission strategy is highly robust to packet losses in wireline networks. We will study its efficiency in wireless networks in the next section. Finally, Google+ reduces its sending rate and video rate as the loss rate goes over 10%.

In Figure 4(b), Skype's redundancy traffic is significantly higher. As packet loss rate increases, the video rate decreases,



(a) Google+ (b) Skype (c) FaceTime

Fig. 4. Redundancy Adaptation as Packet Loss Ratio Increases

while the total sending rate increases. This agrees with the finding in [23] and [20] that Skype employs adaptive-but-aggressive FEC scheme. As will be shown in the next section, Skype's aggressive FEC may lead to a vicious cycle. Both video rate and sending rate drop down significantly after the packet loss rate increases to 15%, but the FEC redundancy ratio is still very high.

TABLE III
RTP PACKET TRACE OF FACETIME

(a) No packet loss injected

Length(byte)	Sequence Number	TimeStamp	Mark
1013	8593	12819453	0
1013	8594	12819453	0
1010	8595	12819453	1
1200	8596	12820189	1
922	8597	12820924	0
917	8598	12820924	1
764	8599	12821660	0
758	8600	12821660	1
771	8601	12822396	0
766	8602	12822396	1

(b) 15% packet loss injected

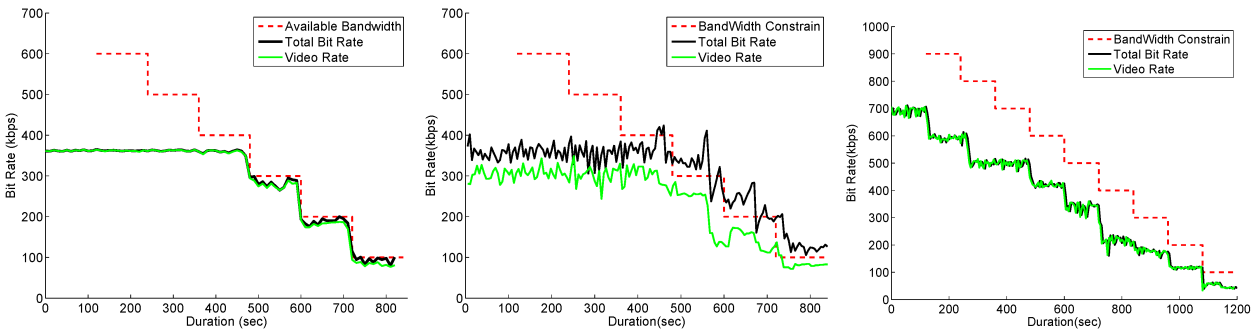
Length(byte)	Sequence Number	TimeStamp	Mark
461	54231	51971449	0
457	54232	51971449	0
454	54233	51971449	1
465	54233	51971449	0
465	54233	51971449	0
465	54233	51971449	0
1365	54234	51972945	0
1361	54235	51972945	0
1361	54236	51972945	0
1359	54237	51972945	1
1373	54237	51972945	0
1373	54237	51972945	0
1373	54237	51972945	0
287	54238	51974186	0
287	54239	51974186	0
285	54240	51974186	1
295	54240	51974186	0
295	54240	51974186	0
295	54240	51974186	0

FaceTime's redundancy ratio in Figure 4(c) lies in between Google+ and Skype. We now closely examine its loss recovery strategy. In Table III, we compare RTP header traces of FaceTime without and with packet losses. Without packet loss, RTP packet sequence number increases at pace 1. All packets carrying the same timestamp are from a same video frame, the last packet carries Mark 1. Due to video encoding structure, some frames are larger, and have more packets. But all RTP packets contain more than 750 bytes. For the trace with packet loss, immediately following the last packet of a frame, we spot some packets, (marked in shade), carrying the same sequence number and timestamp as the last packet of the frame. The payload of those packets are all different from each other, suggesting that they are not duplicate packets. They have identical length, which is larger than the length of all the previous packets in the frame. Finally, with packet loss, all frames are broken into multiple packets, some with short length, e.g., 285. In all our experiments, FaceTime generates much more shorter packets after we inject packet losses.

All these observations strongly suggest that a frame-based FEC scheme is implemented by FaceTime. Original video packets in a frame are put into one FEC block. Redundancy packets are generated to protect original packets. A FEC redundancy packet has to be longer than all original packets it protects. Since it is generated immediately after a video frame is encoded, it has the same timestamp as the original video packets in that frame. Finally, if a FEC block only has one original video packet, then the FEC redundancy ratio has to be multiple of 100%, which is too coarse. Also short FEC blocks (in terms of the number of packets) is vulnerable to bursty loss. To achieve finer FEC redundancy control and higher robustness against bursty losses, for a small video frame that can be fit into one large packet, one should packetize the frame into multiple small packets, and put them into one long FEC block. This explains why FaceTime generates more short packets when packet losses are injected.

D. Rate Control

To avoid congestion along the video transmission path, all three applications adapt their sending rates and video rates to the available network bandwidth. We test their bandwidth tracking capability through a sequence of bandwidth limiting experiments. As illustrated in Figure 5, we use network emulator to set the available network bandwidth, and then record their sending rate and video rate. We start with "unlimited" bandwidth, and record their rates. Both Google+ and Skype set their video rates between 300 and 400 kbps. FaceTime starts at 700 kbps. Two minutes into the "unlimited" bandwidth setting, we set the available bandwidth to be 200 kbps higher than each system's current sending rate, and then keep dropping the bandwidth limit by 100kbps every 2 minutes. While all



(a) Google (b) Skype (c) FaceTime

Fig. 5. Total Rate and Video Rate Adaptation with Available Bandwidth

three system can pick a video rate lower than the available bandwidth, their aggressiveness is quite different.

- Skype chooses a very aggressive video rate to fully utilize the available bandwidth. Since Skype use FEC, the sending rate even often exceeds the available bandwidth. This will cause congestion losses, which in turn trigger more aggressive FEC. We will revisit this in Sec. V-B.
- Google+ also sets its video rate close to the available network bandwidth. Since Google+ uses retransmission, its sending rate is very close to its video rate, and mostly below the bandwidth constraint. It won't trigger many congestion losses.
- When there is bandwidth limit, FaceTime is the most conservative among the three. It always reserve a considerable bit rate margin. Interestingly, it can always track the available bandwidth well. Even though FaceTime also uses FEC, due to its conservative video rate selection, it will not trigger congestion losses by itself. So the FEC redundancy is kept very low in this set of experiments.

V. Conferencing Quality over Wireless

Wireless networks are much more volatile than wireline networks. The unique challenge for mobile video call is to maintain stable realtime video streaming quality in the face of various network impairments, such as random and bursty packet loss, long delay and delay jitter, and time-varying cross-traffic. In this section, we analyze the impact of network impairments on the delivered video quality.

A. Video Conferencing Quality

As with any video streaming service, a user in a video conferencing is sensitive to the perceptual quality of the delivered video, which is determined by various encoding parameters, such as video frame size, frame rate, and quantization levels [14]. The delivered user perceptual video quality increases as the delivered video rate increases. Since video conferencing is to facilitate realtime interaction, users are also highly sensitive to end-to-end video delays, video playback continuity and smoothness. To achieve good overall video conferencing quality, a user's video has to be streamed consistently at high rate and low delay.

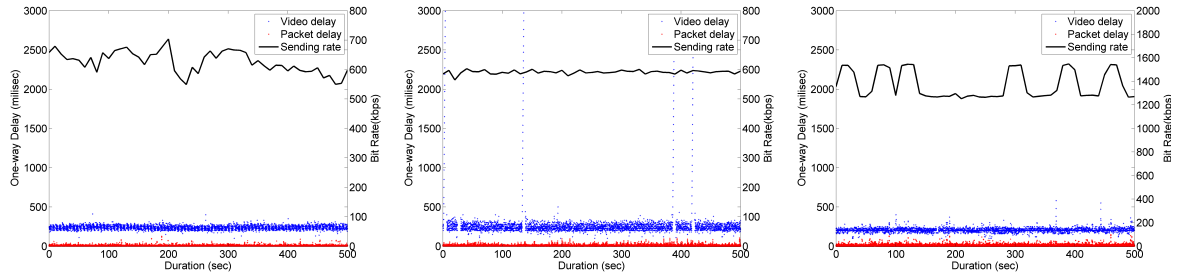
We first compare the voice delay of the three systems over WiFi or cellular with the voice delay of the native phone call service from the carrier. We use the voice delay measurement technique developed in [20]. As reported in Table IV, the phone call service is more stable than the three systems. It is not surprising, given that wireless carriers reserve bandwidth for their phone call services. Another interesting finding is that, over Cellular network, the mean voice delay of Skype is the lowest among all applications, including phone service. One explanation is that Skype does not synchronize voice and video and maintains a short playback buffer for voice.

TABLE IV
Voice Delay (millisec) in a 5-Minute Call

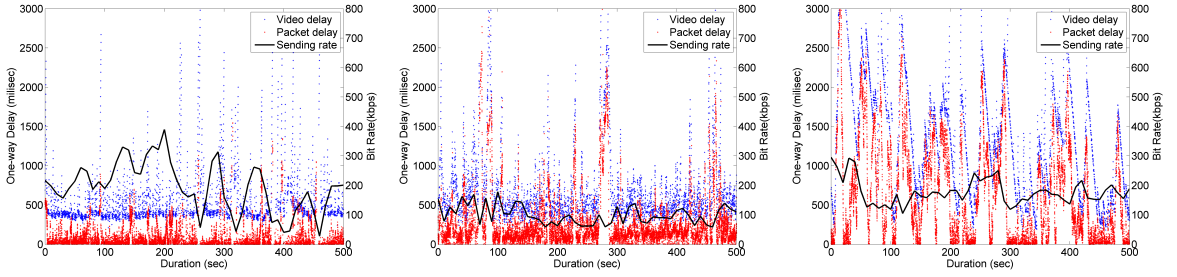
Wireless Type	Phone		FaceTime		Skype		Google+	
	mean	std	mean	std	mean	std	mean	std
Cellular	336.8	32.5	554.9	133.8	205.2	99.2	392.2	345.7
WiFi	N/A		325.0	115.8	160.3	103.8	153.8	123.1

We are more interested in the video quality of the three systems over either WiFi (Figure 6) or Cellular (Figure 7), with strong or weak signal. For each system in each network condition, we plot the total video rate, the one-way packet delay calculated from packet traces, and the one-way video delay collected using the technique presented in Section III-B. The rate curves visually illustrate the video quality variations over time. The measured video delays not only quantify how much delay each system introduces to realtime user interaction, but also enable us to assess the video playback continuity and smoothness. Specifically, spikes in a video delay curve are resulted from video freezes. In the example of Figure 2, whenever the received video freezes, the received stopwatch freezes. Meanwhile, the stopwatch in the source video continues to advance. Consequently, the reading gap of the two stopwatches ramps up quickly until the freeze stops and new video is rendered in the received video window.

It is also interesting to notice how different systems recover from video freezes. For FaceTime and Google+, after each delay spike, video delay jumps back to the normal level; but in Skype, video delay gradually goes back to the normal level. We believe this is due to different policies to handle delayed video frames. In FaceTime and Google+, whenever there is a video freeze, they choose not to display the subsequent frames with long delays. Video playback resumes only after a new video frame is received with acceptable delay. That is when the measured video delay jumps back to the normal level. Skype chooses to display received frames with long delays. To catch up with the realtime conferencing, Skype also plays

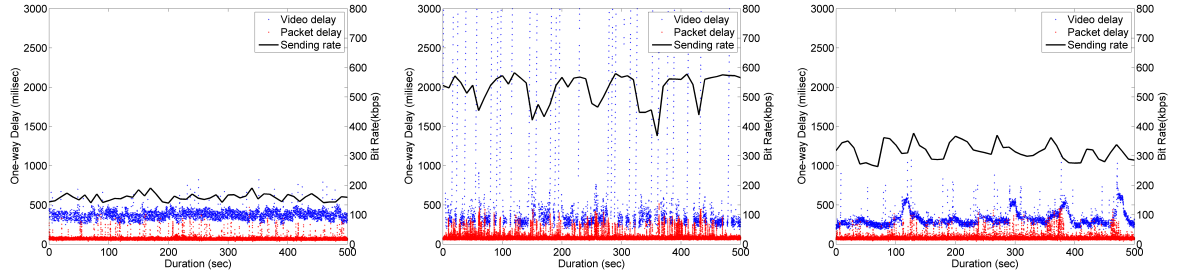


(a) FaceTime - strong signal (b) Google+ Hangout - strong signal (c) Skype - strong signal

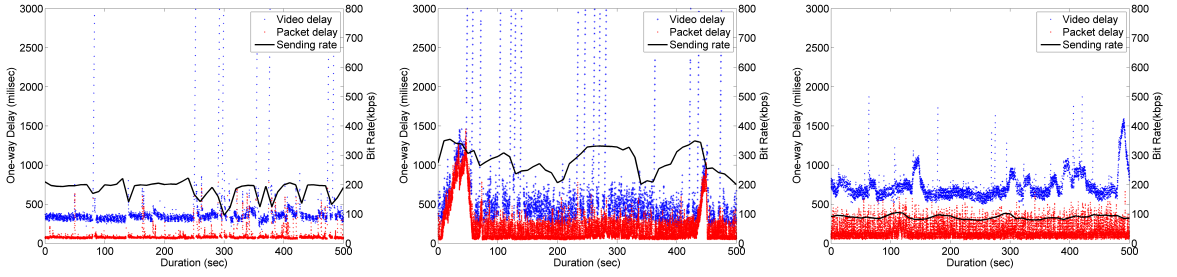


(d) FaceTime - weak signal (e) Google+ Hangout - weak signal (f) Skype - weak signal

Fig. 6. One-way Video Delay for FaceTime Skype and Google+ Hangout over WiFi



(a) FaceTime - strong signal (b) Google+ Hangout - strong signal (c) Skype - strong signal



(d) FaceTime - weak signal (e) Google+ Hangout - weak signal (f) Skype - weak signal

Fig. 7. One-way Video Delay for FaceTime Skype and Google+ Hangout over Cellular

the delayed frames in a fast-forward fashion. This explains the gradual video delay decrease after each spike. Skype's scheme to handle delayed frames has a major problem when packet delay is bursty. This is obvious in our experiments on subway, as illustrated in Figure 8. The round-trip video delay can go as high as 14 seconds. Such high video delay spikes are likely triggered by severely delayed video packets during hand-over periods. When delayed video packets finally arrive, Skype displays the corresponding late video frames in a fast-forward fashion to catch up the realtime video playback. It takes long time for Skype to bring the video delay back to the normal level. This significantly degrades user's conferencing experience. In the same subway experiment setting, Google+ and FaceTime also experience high video delay spikes. But they recover faster than Skype by discarding overly delayed frames. Due to space limit, we refer interested readers to our technical report [21] for the video delay curves of Google+ and FaceTime on subway.

The major problem of Google+ Hangout is video freezes, with duration of 3 to 5 seconds. Skype also experiences video freezes. The difference is the durations of freezes are shorter - most time are within 2 seconds. But besides freezes, Skype

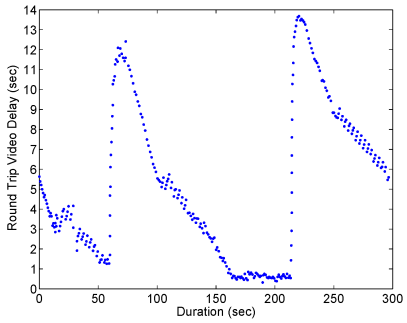


Fig. 8. Skype Round-Trip Video Delay On Subway

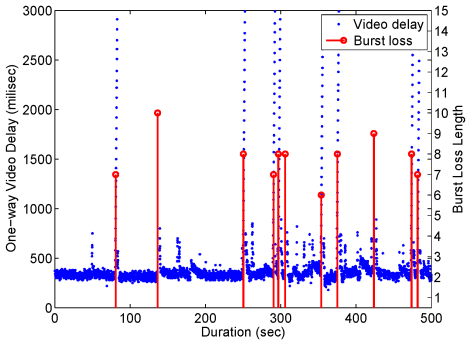


Fig. 9. Correlation between Packet Loss Bursts and Video Freeze In FaceTime over Weak Cellular Link

also suffers nonuniform speed playback. FaceTime has the best performance in terms of video smoothness among three applications. But still, freeze happens from time to time, lasting around 0.5 to 1 second. Given the observed video quality, we want to find out how various network impairments contribute to video conferencing quality degradation.

B. Impact of Packet Losses

Naturally, we first check whether video freezes are triggered by packet losses. We identify packet losses by matching packet traces collected at the sender side and the receiver side. For FaceTime and Skype, packets can be matched by their payload even if they go through a relay server. Google+ relay servers change packet payload, packets are instead matched by their RTP headers. For FaceTime and Skype, packet losses identified this way are indeed the end-to-end packet losses. Google+ employs selective persistent retransmission, the identified packet losses are the end-to-end packet losses not recovered by Google+'s retransmission algorithm, which affect the delivered video quality.

TABLE V
PACKET LOSSES TWO SECONDS BEFORE FREEZES

Process	FaceTime		Skype		Google+	
	Cell	WiFi	Cell	WiFi	Cell	WiFi
Before freezes	5.00	9.98	3.70	8.59	2.63	4.86
Overall	0.43	4.58	0.33	1.78	0.44	4.71

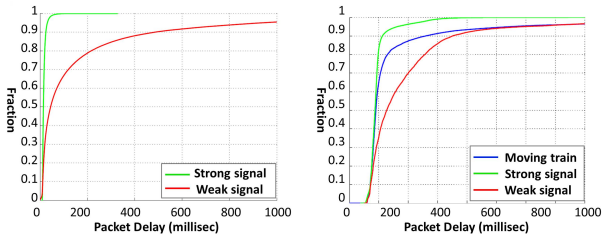
For each video freeze in Figure 6 and 7 which lasts for at least 1 second, we count how many packets were lost in the two-second period immediately before the freeze. We then calculate the average loss number over all freezes, and compare it with the average packet loss number in all two-second periods over the entire experiment. Table V shows that in most cases, the average packet loss number before a video freeze is significantly higher than the overall average, suggesting a strong correlation between video freezes and packet losses. Meanwhile, it should also be noticed that in some cases (e.g. Google+ with strong WiFi signal in Table V), the correlation is weak. We conjecture that in those cases video freezes are mainly due to long packet delays and delay jitters. We will come back to this issue in Section V-C.

TABLE VI
DISTRIBUTION OF LOSS BURST LENGTH

Systems		WiFi		Cellular	
		Strong	Weak	Strong	Weak
Facetime	L_1	3.2	752	0.8	80.4
	L_2	0.4	83.4	0.2	4.4
	L_{3+}	4	13.6	9	6.2
Google+	L_1	17.2	839.8	2.8	48.0
	L_2	2.6	139.2	4.0	13.6
	L_{3+}	7.2	18.0	5.2	5.8
Skype	L_1	3.2	354.8	7.8	102.4
	L_2	0	21.8	1.2	14.0
	L_{3+}	1.2	14.8	1.0	8.2

Skype and FaceTime both adopt FEC to recover from packet losses. Due to realtime encoding and decoding, their FEC blocks have to be short. *However, it is well-known that short FEC blocks are vulnerable to bursty packet losses.* In Table VI, we present the numbers of loss bursts with length 1 (L_1), length 2 (L_2), and length longer than 2 (L_{3+}) for five groups of 600-second video calls in both strong and weak signal. We can see that weak signals triggers bursty losses. As for WiFi, due to higher video throughput, we see more bursty losses than Cellular. Figure 9 visually illustrates the correlation between loss bursts and freezes in FaceTime over weak cellular link.

Packet losses can have more profound negative impact on video conferencing quality than broken FEC blocks. As discussed in Section IV-C and IV-D, Skype aggressively increases its FEC ratio as packet loss rate goes up. Meanwhile, Skype doesn't change its video rate and still tries to take all the available bandwidth. Whenever there is congestion, the available bandwidth shrinks, and packets get lost, Skype still try to add more FEC redundant packets to recover from packet losses. Those FEC redundant packets add fuel to the fire and further increase packet losses. We demonstrated this vicious circle by adding bandwidth constraint to the WiFi router. Due to space limit, please see our technical report [21] for detail.



(a) Over WiFi (b) Over Cellular

Fig. 10. CDF Plot Of One-Way Packet Delay

C. Impact of Packet Delays

Google+ uses application layer retransmission to recover

from packet loss. It works well in computer-based video calls where each computer is connected to a close-by conferencing server with RTT around 10 ms [20]. Unfortunately, wireless links introduce long RTTs. In [18], a ping-style measurement showed that in New York the average RTT of WiFi and Cellular is 111.9 ms and 282.0 ms, respectively, and New York's RTT performance is already on the upper-middle class among all 15 metro areas. We also measured the one-way packet delay between our video sender and receiver. Figure 10 shows that one-way delay variance could go as high as 400 ms. Under such long delays, a retransmitted packet is often useless since it has passed its realtime playback deadline.

From Figure 6 and 7, we can visually observe the correlation between packet delays and video delays. To quantify the correlation, we calculate the correlation coefficients between the two at different time lags. Specifically, Let $D_p(t)$ be the measured delay of a packet captured at the receiver side at its local clock time t; $D_v(t)$ be the stopwatch reading difference recognized from the receiver screenshot captured at its local clock time t. Using the collected packet delay and video delay traces, we can estimate the cross-correlation function between video delay and packet delay as:

$$\gamma_{v,p}(\tau) = \frac{E[(D_v(t+\tau) - \bar{D}_v)(D_p(t) - \bar{D}_p)]}{\sqrt{E[(D_v(t) - \bar{D}_v)^2]E[(D_p(t) - \bar{D}_p)^2]}},$$

where $\bar{D}_v$ and $\bar{D}_p$ are the observed average video delay and packet delay in each experiment. Note, due to video playback buffering, decoding and rendering delays, the video displayed in the received video window at time t are decoded from packets received before t. So the maximum cross-correlation should be observed at some time lag $\tau > 0$. As reported in Table VII, the correlation is not significant in strong WiFi networks, but is very obvious for weak WiFi, and Cellular, strong or weak. This is mainly because Cellular introduces longer packet delays than WiFi even with strong signals.

TABLE VII
CORRELATION BETWEEN PACKET DELAY AND VIDEO DELAY

Systems	WiFi		Cellular	
	Strong	Weak	Strong	Weak
Facetime	0.06	0.61	0.48	0.41
Google+	0.13	0.54	0.33	0.46
Skype	0.22	0.74	0.19	0.59

VI. CONCLUSION

In this paper, we presented our measurement study on mobile video call systems. Through an extensive set of measurements over a wide range of wireless network conditions, we showed that mobile video call quality is highly vulnerable to bursty packet losses and long packet delays; while FEC can be used to recover random packet losses, the inability to differentiate congestion losses from random losses can trigger vicious congestion cycles; conservative video rate selection and FEC redundancy schemes often lead to better video conferencing quality; and end-to-end video delay is highly correlated to end-to-end packet delay in cellular networks, regardless of the signal strength. Insights obtained in this study can be used to guide the design of new solutions to deliver high-quality video calls in wireless networks.

REFERENCES

[1] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting mobile 3g using wifi: Measurement, design, and implementation. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, pages 209–222, June 2010.

[2] S. A. Baset and H. G. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *Proceedings of IEEE International Conference on Computer Communications*, pages 1–11, April 2006.

[3] V. Brik, S. Rayanchu, S. Saha, V. Shrivastava, and S. Banerjee. A measurement study of a commercial-grade urban wifi mesh. In *Proceedings of Internet Measurement Conference*, pages 111–124, October 2008.

[4] K. Chen, T. Huang, P. Huang, and C. Lei. Quantifying skype user satisfaction. In *Proceedings of ACM Special Interest Group on Data Communication*, volume 36, Oct 2006.

[5] L. D. Cicco, S. Mascolo, and V. Palmisano. Skype video congestion control: an experimental investigation. *Computer Networks*, Feb 2011.

[6] P. Deshpande, X. Hou, and S. R. Das. Performance comparison of 3g and metro-scale wifi for vehicular network access. In *Proceedings of Internet Measurement Conference*, pages 301–307, November 2010.

[7] e2eSoft. Vcam: Webcam emulator. http ://www.e2esoft.cn/vcam/.

[8] A. Elmokash, A. Kvalbein, J. Xiang, and K. R. Evensen. Characterizing delays in norwegian 3g networks. In *Passive and Active Measurement Conference*, pages 136–146, March 2012.

[9] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th International Conference on Mobile systems, Applications, and Services*, pages 225–238, June 2012.

[10] T. Huang, K. Chen, and P. Huang. Tuning skype redundancy control algorithm for user satisfaction. In *Proceedings of IEEE International Conference on Computer Communications*, April 2009.

[11] K. Jang, M. Han, S. Cho, H.-K. Ryu, J. Lee, Y. Lee, and S. Moon. 3g and 3.5g wireless network performance measured from moving cars and high-speed trains. In *Proceedings of ACM Mobile Internet through Cellular Networks*, pages 19–24, September 2009.

[12] K. LaCurts and H. Balakrishnan. Measurement and analysis of real-world 802.11 mesh networks. In *Proceedings of Internet Measurement Conference*, pages 123–136, November 2010.

[13] Microsoft Resarch Asia. Network Emulator for Windows Toolkit (NEWT). http ://blogs.msdn.com/b/lkruger.

[14] Y.-F. Ou, Y. Xue, Z. Ma, and Y. Wang. A Perceptual Video Quality Model for Mobile Platform Considering Impact of Spatial, Temporal, and Amplitude Resolutions. In *IEEE Workshop on Image, Video, and Multidimensional Signal Processing*, pages 117 – 122, Jun. 2011.

[15] P. Spoerry. Hidden features in google+ hangouts. http ://plusheadlines.com/hidden-features-googleplus-hangouts/1198/.

[16] redsn0w. Homepage. http ://blog.iphone-dev.org/.

[17] Renovation Software. Text grab for windows. http ://www.renovation-software.com/en/text-grab-sdk/textgrab-sdk.html.

[18] J. Sommers and P. Barford. Cell vs. wifi: On the performance of metro area mobile connections. In *Proceedings of Internet Measurement Conference*, pages 301–314, November 2012.

[19] W. L. Tan, F. Lam, and W. C. Lau. An empirical study on 3g network capacity and performance. In *Proceedings of IEEE International Conference on Computer Communications*, pages 1514–1522, May 2007.

[20] Y. Xu, C. Yu, J. Li, and Y. Liu. Video telephony for end-consumers: Measurement study of google+, ichat, and skype. In *Proceedings of Internet Measurement Conference*, pages 371–384, November 2012.

[21] C. Yu, Y. Xu, B. Liu, and Y. Liu. "Can you SEE me now?", A Study of Mobile Video Calls. *Technical Report, Polytechnic Institute of New York University http://eeweb.poly.edu/faculty/yongliu/docs/info14TR.pdf*.

[22] T. yuan Huang, K. ta Chen, and P. Huang. Could skype be more satisfying? a QoE-Centric study of the fec mechanism in an internet-scale voip system. *IEEE Network*, 24(2):42, Mar 2010.

[23] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang. Profiling skype video calls: Rate control and video quality. In *Proceedings of IEEE International Conference on Computer Communications*, pages 621–629, March 2012.