

# “Can you SEE me now?": A 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, NY, USA 11201

{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, in both WiFi and Cellular networks. Our study is focused on answering the following questions: 1) how they encode and decode video in realtime under tight resource constraints of mobile devices? 2) how they transmit the encoded video smoothly in face of various wireless network impairments, including bursty loss, highly variable delay and competing cross-traffic? 3) what is their delivered video conferencing quality, both video perceptual quality and video delay, under different real mobile network conditions? 4) how different system architectures and design choices adopted by each system 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 new directions to deliver high-quality video calls in wireless networks

## 1. INTRODUCTION

Video telephony is increasingly being adopted by end consumers. Video calls augment voice calls with live visual interaction between users. Just like 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 using 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. “Can you hear me now?” is the basic voice call quality question which all providers try hard to assure their customers that they have a satisfactory answer for, through their services, as well as commercials. Due to its higher bandwidth requirement, it is much more challenging to deliver high-quality video

calls over wireless networks. Even though video call applications are already very popular on various mobile platforms, none of them is provided by wireless service providers. It is therefore not their responsibility to answer the basic video call quality question, “Can you SEE me now?”. As a result, there is very limited understanding about video call quality in wireless networks.

Towards obtaining more understanding, we conduct a measurement study on three popular mobile video call applications: FaceTime [1], Google Plus Hangout [14], and Skype [28], 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 of 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 their delivered quality?*

To answer those challenging questions, we leverage on our previous measurement study of computer-based video conferencing systems [32]. We extend the active and passive measurement methodologies developed in [32] 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 reception, and one of the top-three USA commercial 3/4G network with and without user mobility. We collect an extensive set of measurement traces at both packet level and video level. Through detailed analysis of measurement results, we obtain valuable insights

regarding the unique challenges, advantages and disadvantages of existing design solutions, and possible new directions to deliver high-quality video calls in wireless networks. Specifically, our major 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. End-to-end video delay is highly correlated to end-to-end packet delay in cellular networks, regardless of the signal strength.
4. While FEC can be used to recover random packet losses, the inability to differentiate congestion losses from random losses can trigger vicious congestion cycles, and significantly degrade user video call experience.
5. Conservative video rate selection and FEC redundancy schemes often lead to better video conferencing quality, compared with more aggressive schemes.

The rest of the paper is organized as follows. We briefly discuss the related work in Section 2. Our measurement platform and methodology are introduced in Section 3. The key design choices of the three systems are presented in Section 4. In Section 5, 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 6.

## 2. RELATED WORK

As the popular usage of wireless technologies, there are a lot of measurement work about cellular and WiFi performance. Some papers studies the behavior and characteristics of WiFi network [6, 19]. The work of Tan *et al.*[30] is one of the first empirical study for 3G cellular network. Their work studied throughput and other performance characteristics in the 3G environment. Then, a lot of publications could be found related to cellular network measurement, like developing tools to measure performance, studying some explicit performance properties, measuring performance under some specific usage environments etc [18, 12, 3, 27, 13]. There are also some studies to compare and contrast cellular and WiFi performance [2, 29, 10]. Their conclusions show that compared with 3G cellular network, WiFi provides better download performance and

smaller latency. Most recently, Huang *et al.*[16] studied the performance and power characteristics in the 4G LTE networks. They observe that LTE can offer higher downlink and uplink throughput than 3G and even WiFi.

Most of the measurement studies of realtime communications over the Internet were focused on Skype’s VoIP service. Baset *et al* [4] first analyzed Skype’s P2P topology, call establishment protocol, and NAT traversal mechanism. Since then, a lot of papers have been published on Skype’s overlay architecture, P2P protocol, VoIP traffic and the quality of Skype’s voice-over-IP (VoIP) calls[5, 15, 7, 17, 33]. Skype’s FEC mechanism and its efficiency for voice streaming were studied in [17, 33]. In [7], Huang *et al.* proposed a user satisfaction index model to quantify VoIP user satisfaction. Cicco *et al.*[8] proposed a congestion control model for Skype VoIP traffic.

More recently, there are some measurement work on video telephony. Cicco *et al.*[9] measured the responsiveness of Skype video calls to bandwidth variations. They conclude that Skype’s response time to bandwidth increase is too long. In [34], we conducted an extensive measurement study of Skype two-party video calls under different network conditions. Based on the measurement results, we propose models for Skype video calls’ rate control, FEC redundancy, and video quality. In [32], we investigated system architecture, video generation and adaptation, packet loss recovery, user quality-of-experience for three existing multi-party video conferencing solutions: iChat, Google+ hangout, and Skype.

## 3. MEASUREMENT PLATFORM

FaceTime, Skype, and Google+ Hangout all use proprietary protocols and encode their signaling and data. Using methodology similar to those developed in [32] for studying computer-based video conferencing systems, we measure them as “black boxes”, reverse-engineer their design choices, and compare their performance in challenging wireless environment. 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. We restrict our study to two-party video calls.

### 3.1 Testbed

#### 3.1.1 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 Cellular (cellular data service is provided by one of the top three US carriers). At the wireline user side, a PC or Mac is connected to the Internet through campus Ethernet. Software-based net-

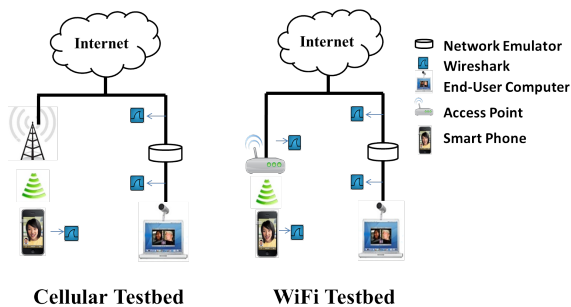


Figure 1: Measurement Testbed

work 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 [11]. 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.

### 3.1.2 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.

1. *Privilege escalation*: We use a unlocked iPhone 4S, which underwent jailbreak to allow third-party application installation [25];
2. *Measurement tool installation*: Several tools are downloaded from multiple sources. We list their names, sources, and brief function descriptions in Table 1.
3. *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 (Google Nexus 4) to login and control the iPhone via SSH, which will not interrupt the video call application. For experiments without mobility, we connect the phone to a computer via a USB cable. Then we use a software called iTool on the computer to build a

USB tunnel between the *localhost* and the iPhone. Then we can access the iPhone from the computer using the command `ssh 127.0.0.1` directly. For experiments on a moving train, it is not convenient to use a control computer. We instead control the iPhone using another Android phone. 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 WiFi.

Table 1: Measurement Tool List For The Phone In TestBed

Name	Source	Description
Skype	Apple Store	Skype video call application
Google+	Apple Store	Video call function is embedded in the “Hangout” section
SpeedTest	Apple Store	Quick test about current ping value, throughput to a local SpeedTest server
OpenSSH	Cydia	Enable login to iPhone from a remote computer
APT 0.7 HTTPS Method	Cydia	Enable downloading application from Debian to iPhone
ping	Debian	Ping function
top	Debian	Display the CPU usage of current processes
tcpdump	Debian	Packet sniffing on iPhone
My3G	Cydia	Force FaceTime using Cellular instead of WiFi
MyFi	Cydia	Enable WiFi Adhoc mode on iPhone.
Display Recorder	Cydia	Record the screen of iPhone.

### 3.1.3 Wireless environment setting

To test application performance under different wireless conditions, we design a set of WiFi and Cellular experiments with both weak and strong signals. Mobile users experience different network conditions when they are in different locations. In our experiments, we carefully choose several typical locations which can represent strong signal case or weak signal case. We use `SpeedTest` to verify the actual upload throughput. In WiFi experiments, upload throughput for strong signal locations is above 10 mbps, and around 200 – 300 kbps for weak signal locations; in Cellular experiments, upload throughput for strong signal locations is above 1,000 kbps, and for 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 [20] to emulate a variety of network attributes, such as propagation delay, random packet loss, and available

bandwidth.

### 3.2 Information collection

For each experiment, we collect information of video calls from multiple channels.

1. *IP Packet Traces*: We sniff packets at both the computer side (with Wireshark [31]) and the smart-phone side (with command line `tcpdump`). Collected packet traces are used to analyze protocols and calculate network level statistics, such as packet delay, packet 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, [24]. We use a screen text capture tool [26] to capture these information periodically. The sampling interval is 1 second.
3. *End-to-end Video Delay Samples*: Same as in our previous work [32], 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 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 the 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.

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 Figure 2. Instead, we can measure the *round-trip video delay* using the scheme illustrated in Figure 3.

There are totally five stopwatch videos during a video 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.

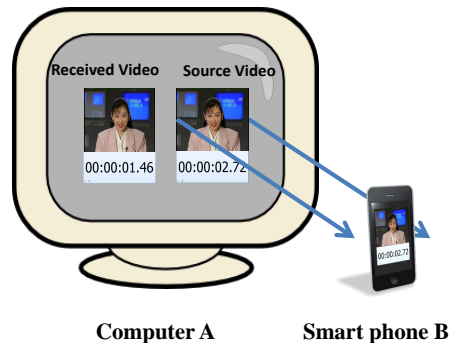


Figure 2: One-way video delay testbed.

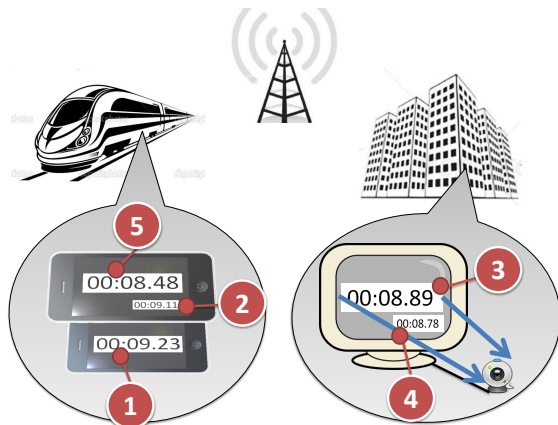


Figure 3: Round-trip video delay testbed on subway.

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.

## 4. HOW DO THEY WORK - KEY DESIGN CHOICES

Before investigating user video call experience of all three systems, we need to understand their design choices in terms of system architecture, video generation & adaptation, and packet loss recovery, etc. Leveraging on our study in [34, 32] for their corresponding computer versions, we are able to discover important de-



sign choices of Google+ and Skype’s mobile versions. We also obtain good understanding on FaceTime, which was not covered by our previous study.

## 4.1 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 be 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. The architecture and protocol comparison is summarized in Table 2

**Table 2: System Architecture and Protocol Comparison**

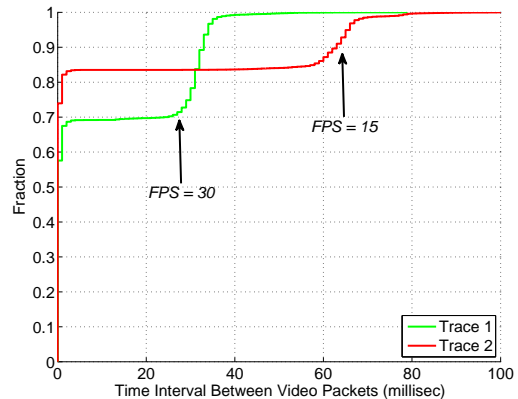
System	P2P or Server	UDP or TCP	RTP
Skype	Relay server or P2P	UDP, may use TCP	No
Google+	Server-centric	UDP, only use TCP when UDP is blocked	Yes
Facetime	WiFi: mostly P2P Cellular: relay server	Always use UDP	Yes

## 4.2 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 band-

**Table 3: 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



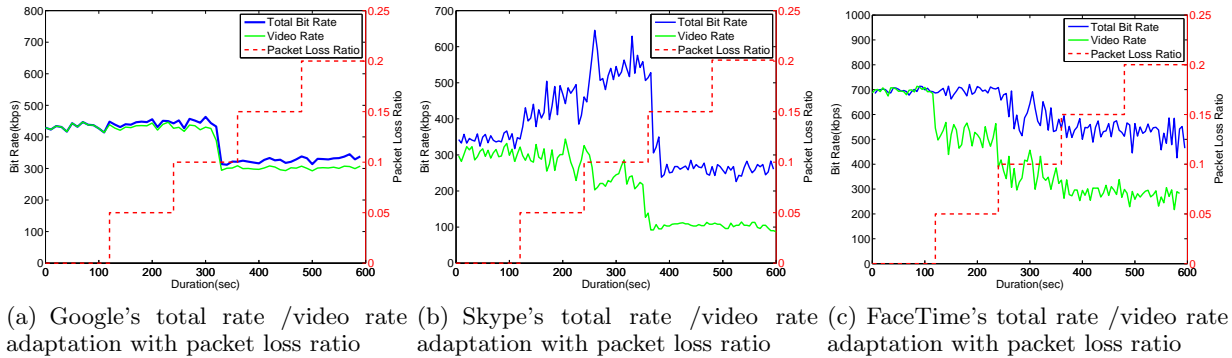
**Figure 4: CDF Plot of Time Interval Between Video Packets in Facetime.**

width 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 3. Using the same RTP header analysis technique introduced in [32], we verified that Google+ still uses layered video coding on mobile phones, thanks to the abundant computation power on smartphones. Both temporal and spatial scalability are used to generate video in a wide rate range.

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  $\Delta$  be the minimum non-zero timestamp difference. Any packet pair with timestamp difference  $\Delta$  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  $1/t_c$ . Figure 4 plots the CDF of inter-arrival time of video packets in two traces. Based on the high density around 33ms and 66ms, we can infer the frame rate to be 30 FPS and 15 FPS respectively. We don’t know FaceTime’s video resolution.

## 4.3 Loss Recovery

Wireless networks have both congestion losses and



**Figure 5: Redundancy Adaptation of FaceTime, Skype and Google+ Hangout**

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 5, 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 5(a), Google+'s total sending rate is only slightly higher than its actual video rate. This is consistent with our finding in [32] 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 [32] 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 5(b), Skype's redundancy traffic is significantly higher. As packet loss rate increases, the video rate decreases, while the total sending rate increases. This agrees with the finding in [34] and [32] 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.

FaceTime's redundancy ratio in Figure 5(c) lies in between Google+ and Skype. We now closely examine its loss recovery strategy. In Table 4, 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

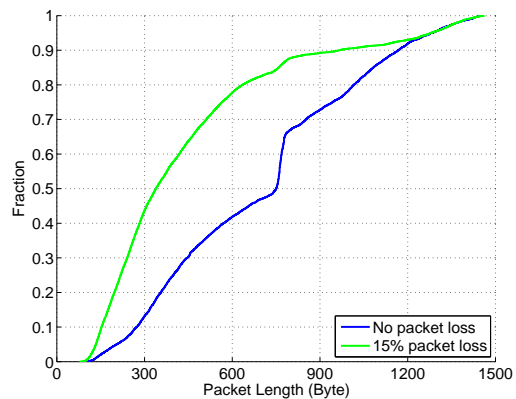
**Table 4: RTP Packet Trace of FaceTime**  
(a) No packet loss injected

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

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. Figure 6 plots the packet length distribu-



**Figure 6: Packet Length CDF Plot of FaceTime**

tions with and without packet losses. It is obvious that FaceTime generates much more shorter packets after we inject packet losses.

All of 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, as illustrated in Table 4 and Figure 6.

#### 4.4 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 7, 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 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 may cause more aggressive FEC.
- Google+ also sets 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 becomes the most conservative one 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.

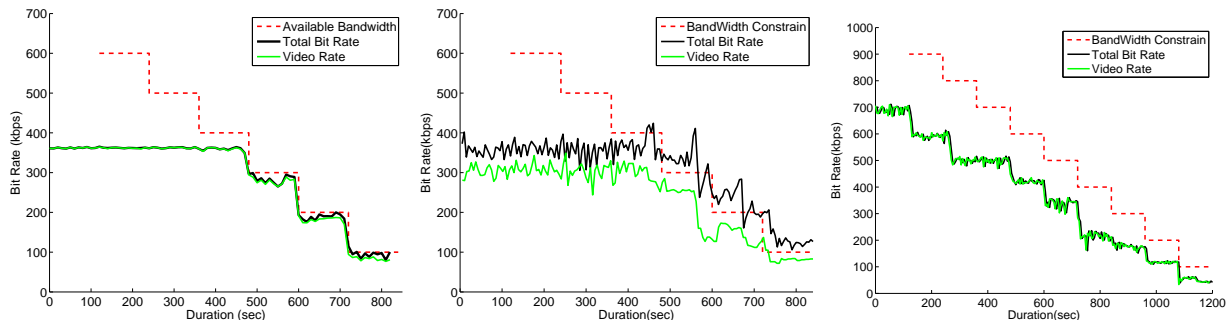
#### 4.5 Power Consumption

Compared with computers, mobile devices are much more constrained by CPU cycles and battery supplies. It is therefore very important to gauge the CPU and battery consumption of the three mobile video call applications. Table 5 reports the CPU consumption of the top-3 processes when we use each of the three applications for a five minutes video call on WiFi or Cellular. Since the iPhone has dual-core, the full utilization is 200%. To test the power consumption of video

**Table 5: Top-3 CPU-consuming Processes**

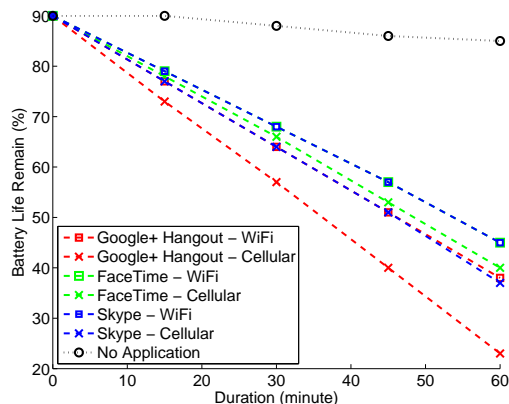
Process	FaceTime		Skype		Google+	
	Cell	WiFi	Cell	WiFi	Cell	WiFi
mediaserve	52.3%	61.3%	13.2%	12.3%	15.3%	15.5%
backboardd	9.4%	10.5%	8.9%	8.2%	12.0%	12.1%
Skype	-	-	70.1%	86.0%	-	-
EmSea	-	-	-	-	138.2%	134.4%
Total	61.7%	71.8%	92.2%	106.6%	165.5%	161.6%

call applications, we start a video call for one hour and continuously monitor the remaining battery life during the call. In Figure 8, each system consumes more power over Cellular than over WiFi. This is expected, as Cellular transceiver consumes more power than WiFi transceiver. Among the three application, Google+ is the most power demanding, similar to the CPU consumption. This might again due to Google+ uses realtime layered coding, which is known to consume more power than non-layered video coding. FaceTime is slightly more power efficient than Skype, possibly again because it is an integrated app in iPhone.



(a) Google’s total rate /video rate adaptation with available bandwidth (b) Skype’s total rate /video rate adaptation with available bandwidth (c) FaceTime’s total rate /video rate adaptation with available bandwidth

**Figure 7: Video Rate Adaptation of FaceTime Skype and Google+ Hangout**



**Figure 8: Battery Life Remaining During 1 hour Video Calls (starting for 90% full)**

## 5. VIDEO CONFERENCING QUALITY UNDER NETWORK IMPAIRMENTS

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 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 in the three systems.

### 5.1 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 [21, 22, 23]. 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 data has to be streamed consis-

tently at high rate and low delay, with low rate and delay variations.

Here we first compare the voice delay of the three systems over WiFi or cellular with the voice delay of the integrated voice call service from the service providers. We use the voice delay measurement technique developed in [32]. As reported in Table 6, the integrated voice service is more stable than the three systems. It is not surprising, given that wireless carriers reserve bandwidth for their voice services.

**Table 6: Voice Delay (millisec) During A Regular Phone Call Or Video Calls in 5 Minutes**

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 9) or Cellular (Figure 10), with strong or weak signal. For each system in each network condition, we plot the total video traffic rate and the one-way video delay collected using the technique presented in Section 3.2. 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 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 Face-



Time and Google+, whenever there is a video freeze, they choose not to display subsequent frames with long delays. Video playback resumes only after a 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 the delayed frames in a fast-forward fashion. This explains the gradual video delay decrease after each spike.

The major problem of Google+ Hangout is video freeze, with duration lasting 3 to 5 seconds. Skype also experiences video freeze. The difference is the duration of freeze is shorter - most time are within 2 seconds. But besides freezes, Skype 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. Same video delay measurement was performed on a moving subway. As showed in Figure 11, we are surprised by the finding that the round-trip video delay can go as high as 14 seconds. For Skype, after a long delay, it takes so long to recover video delay to normal. Video call user had to suffer from delayed video playback as long as 100 seconds. Instead, FaceTime and Google+ choose to give up those delayed video frames and quickly go back to normal playback after video freezing.

Given the observed video quality, we want to find out how various network impairments contribute to video conferencing quality degradation.

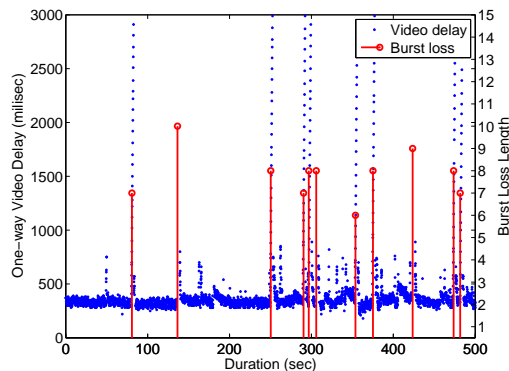
## 5.2 Impact of Packet Loss

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 7: Number of Packet Losses Within 2 Seconds Before Video Freezes**

Process	FaceTime		Skype		Google+	
	Cell	WiFi	Cell	WiFi	Cell	WiFi
Before freezing	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 9 and 10 which lasts for at least 1 second, we count how many packets were lost in the two-second period immediately before the



**Figure 12: Impact of Bursty Packet Losses on Video Freeze In FaceTime over Weak Cellular**

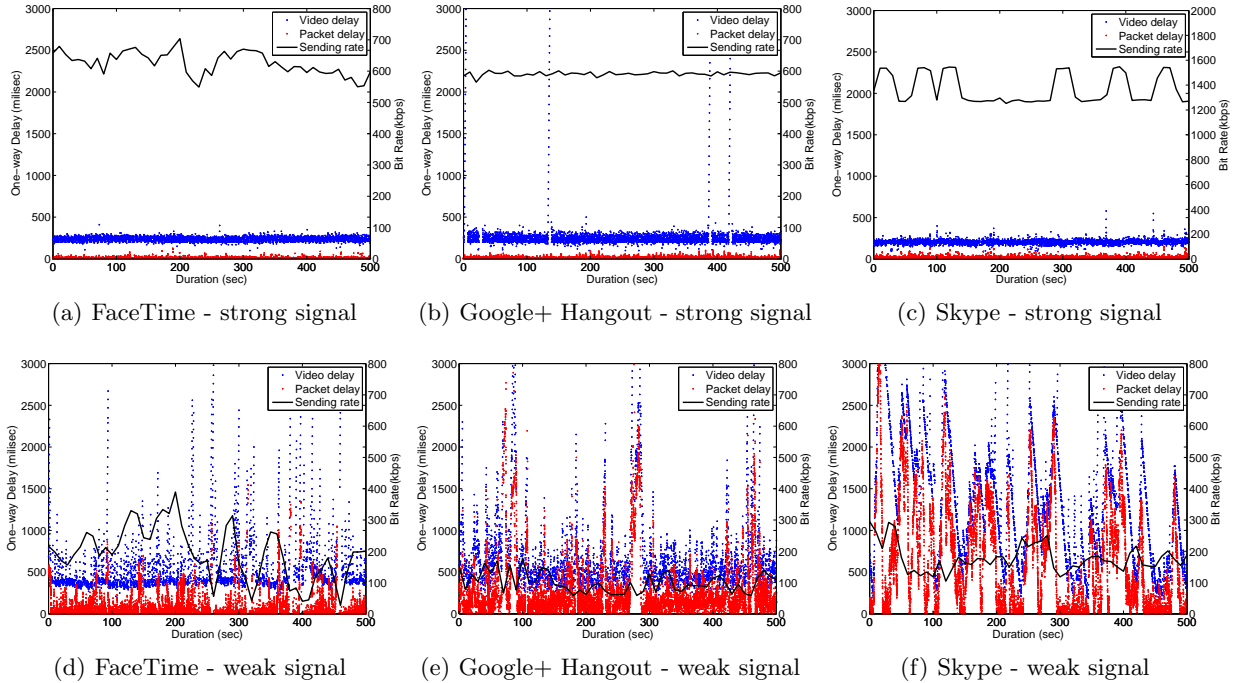
freeze. We then calculate the average loss number over all freezes, and compare it with the average packet loss number in all two-second periods over the entire experiment. Table 7 shows that in most cases, the average packet loss number before a video freeze is significantly higher than the overall 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 7), 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 5.3.

### 5.2.1 FEC is not efficient enough to recover from bursty losses in wireless environment

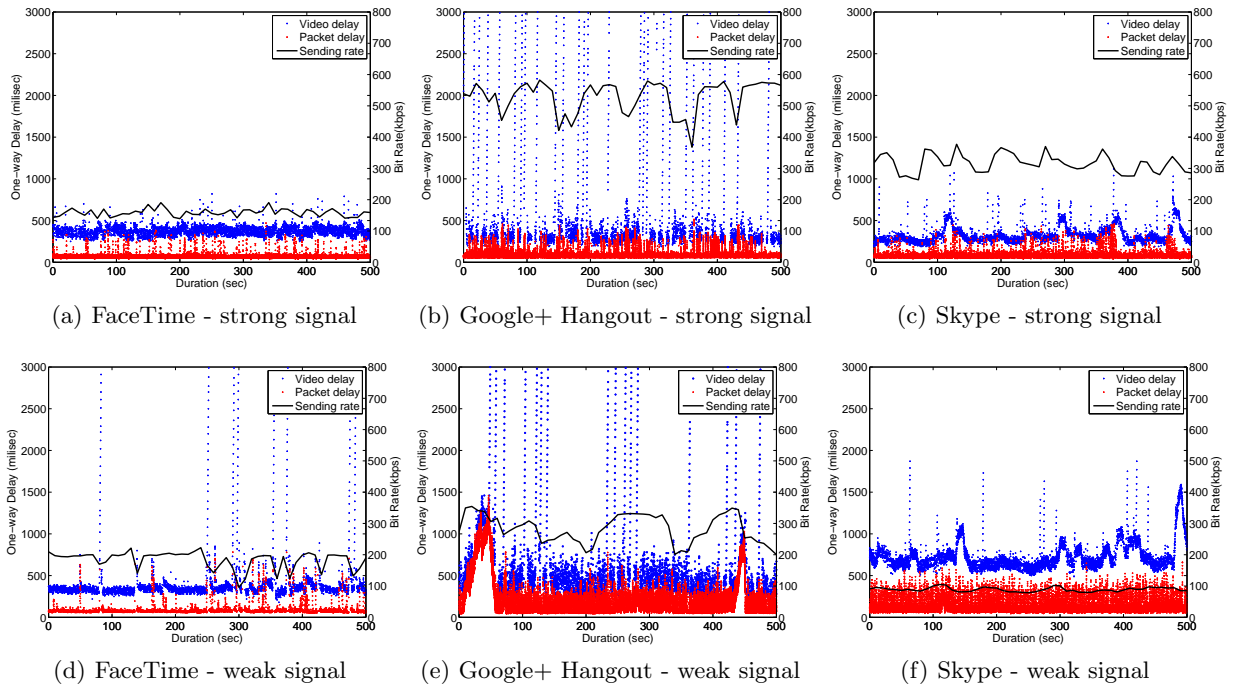
**Table 8: Burst Loss Length**

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 adopts FEC to recover from packet losses. However, it is well-known that FEC is vulnerable to bursty packet losses. Generally, for a FEC code, such Reed-Solomon code, if one FEC block of  $n$  packets contains  $k$  original data packets, FEC protection fails whenever the number of losses experienced by the  $n$  packets is greater than  $n - k$ . One solution is to increase FEC block length  $n$  while keeping a low FEC overhead ratio  $\frac{n-k}{k}$ . This requires each FEC block contains video packets from multiple video frames, especially when the video encoding rate is low. In video conferencing, video frames are generated in realtime, long FEC blocks lead to long FEC encoding and decoding delays. As a result, FEC blocks in video conferencing have to be short. Our previous work [32] has demonstrated that Skype's



**Figure 9: One-way Video Delay for FaceTime Skype and Google+ Hangout over WiFi**



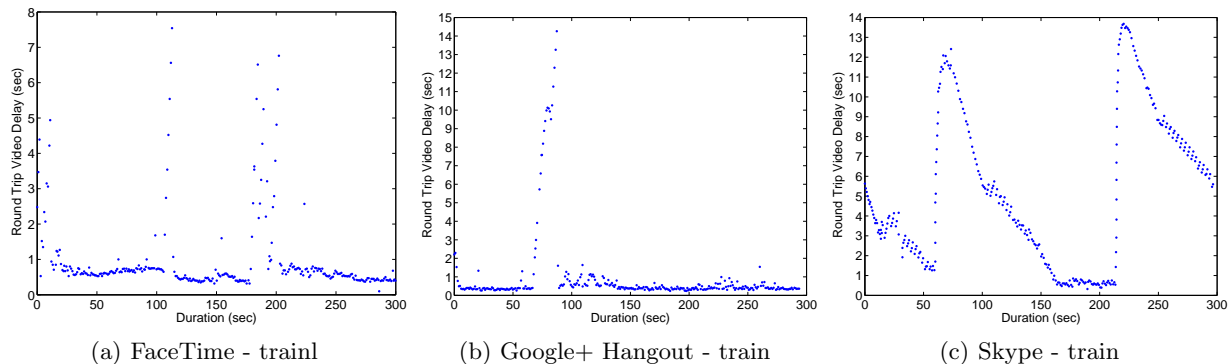
**Figure 10: One-way Video Delay for FaceTime Skype and Google+ Hangout over Cellular**

FEC cannot recover from burst losses injected by a network emulator. Now we investigate the impact of bursty losses in real wireless network environment.

Given a stream of packets sent out by the source, our packet-matching process identifies whether a packet is received or not. In Table 8, we plot the distribution of loss burst length of five groups of 600-second video

calls in both strong and weak signal. We can see that when the video call is over Cellular, packet loss with burst length larger or equal to 2 happened at least 10 times. As for WiFi, due to higher video throughput, we see more bursty losses. There are several loss bursts with length larger than 10. It is very hard for FEC to recover from those loss bursts. Figure 12 visually





**Figure 11: Round-Trip Video Delay for FaceTime Skype and Google+ Hangout On Moving Subway Skyline**

illustrates the correlation between loss bursts and video freeze for FaceTime under weak cellular signal.

### 5.2.2 Skype’s FEC scheme leads to “vicious circle”

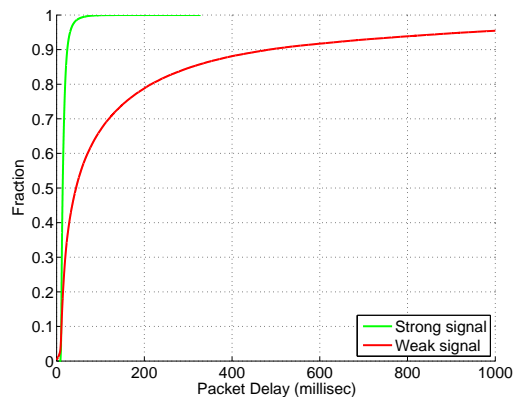
There is another problem with Skype’s FEC scheme, which is more fatal to its video quality. As we discussed in Section 4.3 and 4.4, Skype aggressively increases its FEC ratio as packet loss rate goes up. Meanwhile, Skype doesn’t change its video rate and still tries to take all the available bandwidth. Whenever there is congestion, 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.

In Figure 13, we demonstrate this vicious circle by adding bandwidth constraint to the WiFi router. We start with bandwidth cap of 500 kbps. After the sending rate stabilizes in five minutes, we drop the available bandwidth by 100kbps. Then we plot the video rate, total sending rate, and one-way video delay in the same figure. Before the bandwidth drop, the Skype sending rate is either already higher than the available bandwidth (Figure 13(c)) or very close to it (Figure 13(d)). After the drop, immediately Skype senses a high loss ratio and keeps a high FEC redundant ratio (Figure 13(c)) or increases it (Figure 13(d)). This leads to large video delay oscillations. We did the same experiment with Google+ Hangout and FaceTime. Both of them keep their sending rates below the bandwidth cap and video delay is acceptable most of the time.

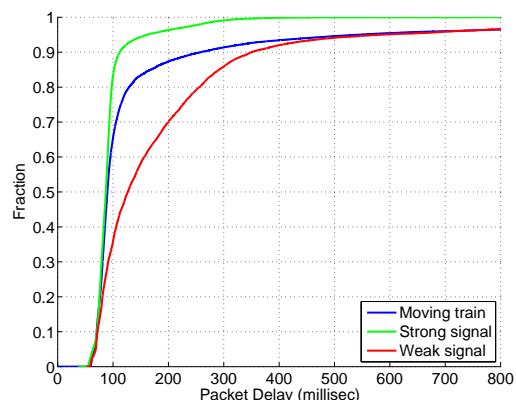
In practice, the available network bandwidth is more dynamic than the constant cap. To test how different systems cope with changing bandwidth, we added both fixed bandwidth constraint (1 mbps) and background TCP connections (total number: 0,1 or 3) to the WiFi router. Figure 14 shows a similar unstable video playback pattern on Skype. After a TCP competing flow was injected, Skype’s video was distorted so much that even the stopwatch reading in the received video

window cannot be recognized by the OCR application. Video freeze can go as high as several seconds. Again, although the video quality of Google+ Hangout and FaceTime are also impacted by the TCP flows, things are much better controlled when compared with Skype.

### 5.3 Impact of Packet Delay



(a) Over WiFi



(b) Over Cellular

**Figure 15: CDF Plot Of One Way Packet Delay**

Wireless networks can introduce long packet delays. In [29], a ping-style measurement show that in New

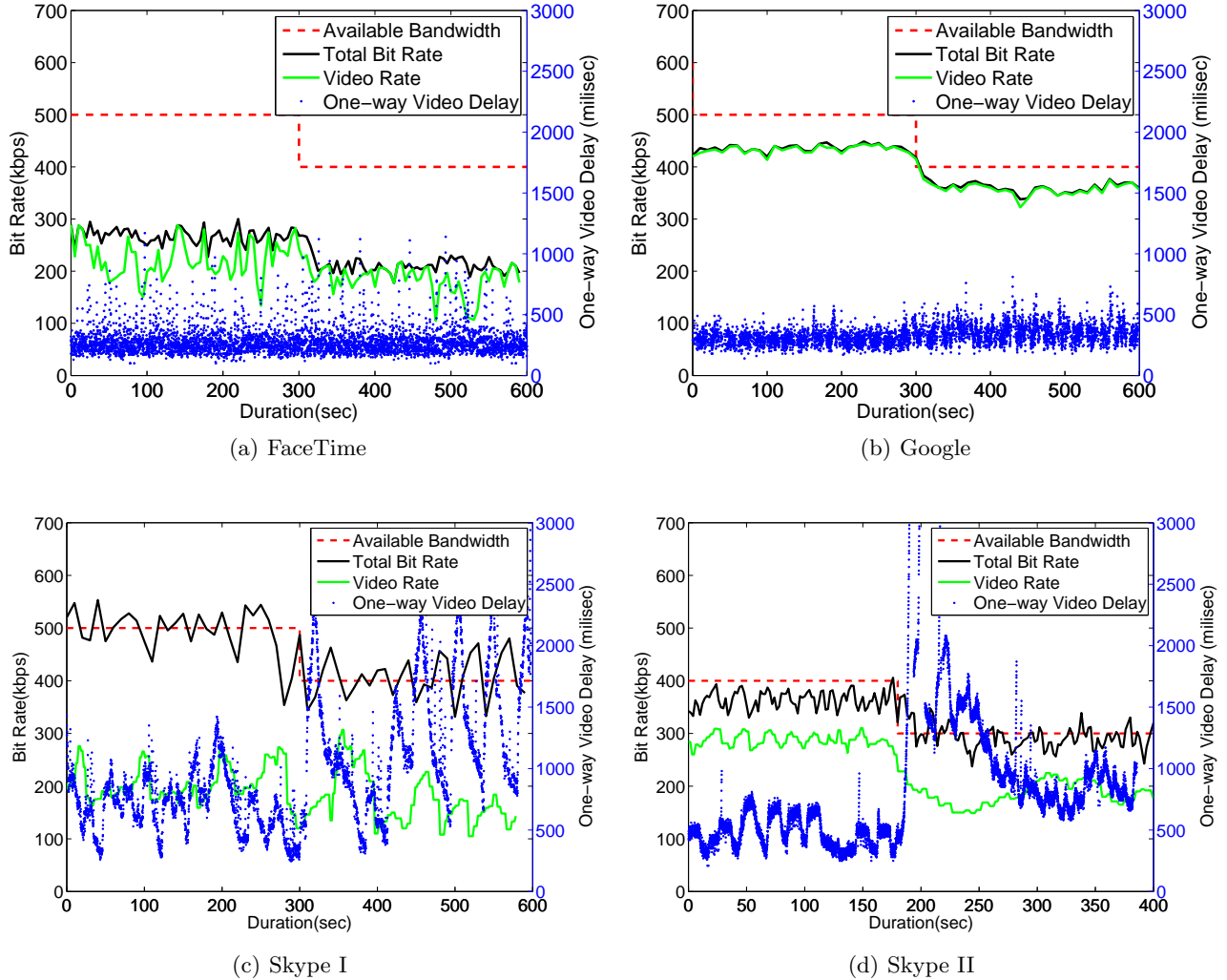


Figure 13: Video Delay Variations After Available Bandwidth Changes

York, NY, the average RTT of WiFi and Cellular is 111.9 and 282.0, respectively, and New York’s RTT performance is already on the upper-middle class among all 15 metro areas. We also measure the one-way packet delay between our video sender and receiver. Figure 15 shows that one-way delay variance could go as high as 400 ms. To cope with this, a buffer at video side must be set and a quite a lot of delay will be introduced.

Table 9 compares the average packet delay within two seconds immediately before a video freeze with the average delay over the whole experiments with weak WiFi/Cellular signal. It is noted that packet delay right before freezes are higher, especially in WiFi.

Table 9: Average Packet Delay Within 2 Seconds Before Video Freeze

Process	FaceTime		Skype		Google+	
	Cell	WiFi	Cell	WiFi	Cell	WiFi
Before freeze(ms)	155.7	172.2	83.4	1384	154.5	622.6
Overall (ms)	94.3	61.7	85.6	274.2	155.7	87.0

To correlate user-perceived video delay with end-to-end packet delay, 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 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

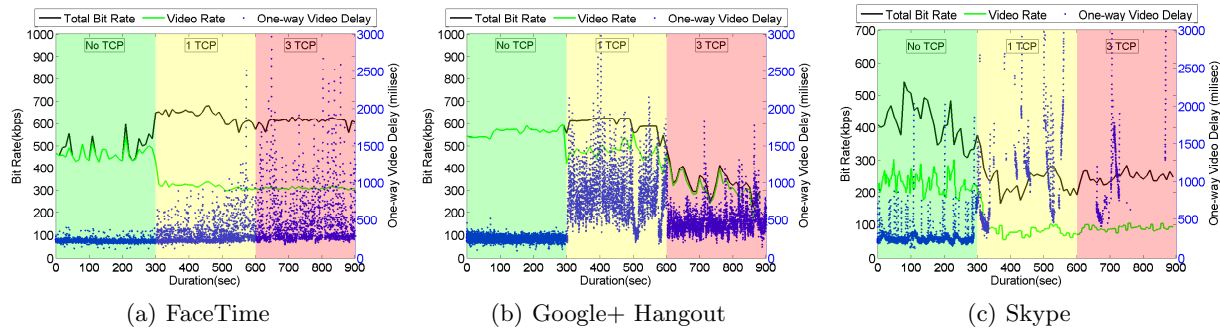


Figure 14: Video Delay Variations When Competing With TCP Flows

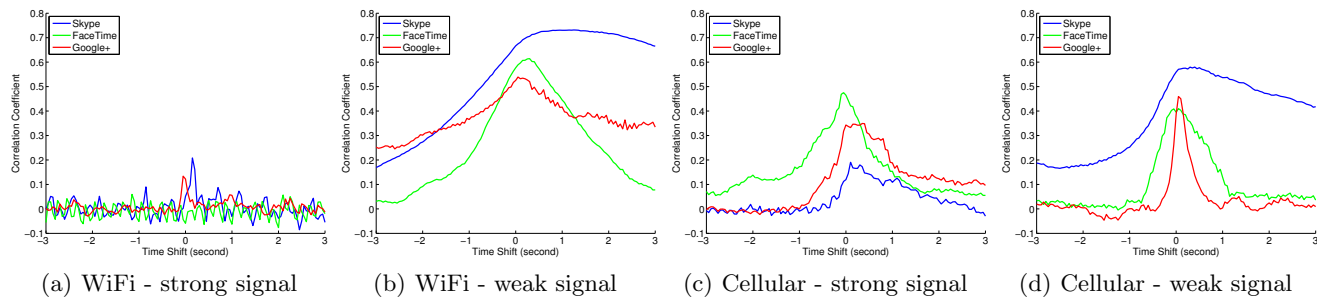


Figure 16: Cross-Correlation coefficient function between one-way video delay and packet delay vs. time shift

$\tau > 0$ . As plotted in Figure 16, 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.

## 6. 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; end-to-end video delay is highly correlated to end-to-end packet delay in cellular networks, regardless of the signal strength; while FEC can be used to recover random packet losses, the inability to differentiate congestion losses from random losses can trigger vicious congestion cycles; and conservative video rate selection and FEC redundancy schemes often lead to better video conferencing quality. Insights obtained from this study can be used to guide the design of new solutions that can deliver high-quality video calls in wireless networks.

## 7. REFERENCES

- [1] Apple Inc. Facetime for iphone. <http://www.apple.com/iphone/features/facetime.html>.
- [2] 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.
- [3] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of Internet Measurement Conference*, pages 280–293, November 2009.
- [4] S. A. Baset and H. G. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *Proceedings of IEEE INFOCOM*, pages 1–11, April 2006.
- [5] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi. Tracking down skype traffic. In *Proceedings of IEEE INFOCOM*, pages 261–265, Apr 2008.
- [6] V. Brik, S. Rayanchu, S. Saha, S. Sen, 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.
- [7] K. Chen, T. Huang, P. Huang, and C. Lei. Quantifying skype user satisfaction. In *Proceedings of ACM SIGCOMM*, volume 36, Oct

- 2006.
- [8] L. D. Cicco, S. Mascolo, and V. Palmisano. A mathematical model of the skype voip congestion control algorithm. In *Proceedings of IEEE Conference on Decision and Control*, Dec 2008.
- [9] L. D. Cicco, S. Mascolo, and V. Palmisano. Skype video congestion control: an experimental investigation. *Computer Networks*, 55(3):558 – 571, Feb 2011.
- [10] 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.
- [11] e2eSoft. Vcam: Webcam emulator. <http://www.e2esoft.cn/vcam/>.
- [12] 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.
- [13] A. Gember, A. Akella, J. Pang, A. Varshavsky, and R. Caceres. Obtaining in-context measurements of cellular network performance. In *Proceedings of Internet Measurement Conference*, pages 287–300, November 2012.
- [14] Google+. Homepage. <https://plus.google.com/>.
- [15] S. Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems*, pages 1–6, Santa Barbara, CA, February 2006.
- [16] 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.
- [17] T. Huang, K. Chen, and P. Huang. Tuning skype redundancy control algorithm for user satisfaction. In *Proceedings of IEEE INFOCOM*, pages 1179–1185, April 2009.
- [18] 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 MICNET*, pages 19–24, September 2009.
- [19] 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.
- [20] Microsoft Research Asia. Network Emulator for Windows Toolkit (NEWT). <http://blogs.msdn.com/b/1kruger>.
- [21] Y.-F. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang. Model the impact of frame rate on perceptual quality of video. In *Proc. of IEEE ICIP*, 2008.
- [22] Y.-F. Ou, Z. Ma, and Y. Wang. A novel quality metric for compressed video considering both frame rate and quantization artifacts. In *Proc. of Intl. Workshop Video Processing and Quality Metrics for Consumer (VPQM)*, Scottsdale, AZ, Jan. 2009.
- [23] 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.
- [24] Paul Spoerry. Hidden features in google+ hangouts. <http://plusheadlines.com/hidden-features-googleplus-hangouts/1198/>.
- [25] redsn0w. Homepage. <http://blog.iphone-dev.org/>.
- [26] Renovation Software. Text grab for windows. <http://www.renovation-software.com/en/text-grab-sdk/textgrab-sdk.html>.
- [27] Z. Shafiq, L. Ji, A. Liu, J. Pang, and J. Wang. A first look at cellular machine-to-machine traffic - large scale measurement and characterization. In *Proceedings of ACM SIGMETRICS/Performance*, pages 65–76, June 2012.
- [28] Skype Inc. Skype features. <http://www.skype.com>.
- [29] 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.
- [30] W. L. Tan, F. Lam, and W. C. Lau. An empirical study on 3g network capacity and performance. In *Proceedings of IEEE INFOCOM*, pages 1514–1522, May 2007.
- [31] Wireshark. Homepage. <http://www.wireshark.org/>.
- [32] 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.
- [33] 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.
- [34] 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 INFOCOM*, pages 621–629, March 2012.