Screencast in the Wild: Performance and Limitations

Chih-Fan Hsu[1], De-Yu Chen[1], Chun-Ying Huang[2], Cheng-Hsin Hsu[3], and Kuan-Ta Chen[1]
[1]Institute of Information Science, Academia Sinica
[2]Department of Computer Science and Engineering, National Taiwan Ocean University
[3]Department of Computer Science, National Tsing Hua University

ABSTRACT

Displays without associated computing devices are increasingly more popular, and the binding between computing devices and displays is no longer one-to-one but more dynamic and adaptive. *Screencast* technologies enable such dynamic binding over ad hoc one-hop networks or Wi-Fi access points. In this paper, we design and conduct the first detailed measurement study on the performance of state-of-the-art screencast technologies. By varying the user demands and network conditions, we find that Splashtop and Miracast outperform other screencast technologies under typical setups. Our experiments also show that the screencast technologies either: (i) do not dynamically adjust bitrate or (ii) employ a suboptimal adaptation strategy. The developers of future screencast technologies are suggested to pay more attentions on the bitrate adaptation strategy, e.g., by leveraging cross-layer optimization paradigm.

Categories and Subject Descriptors: H.5 [Information Systems Applications]: Multimedia Information Systems

Keywords: Measurement; streaming; wireless networks; experiments

1. INTRODUCTION

Increasingly more, and often larger and less expensive, digital displays have been deployed in homes, offices, schools, shops, and other public/private spaces. For example, several vendors have proposed to make refrigerators and washing machines more intelligent and embed displays in them. On the other hand, the display technology is evolving quickly, and we will soon have elastic displays on newspapers, business cards, or clothes. Such next-generation displays will likely come without associated computing devices, but instead rely on other devices, such as smartphones, for the computation and data hosting. We envision that by that time, the binding between computing devices and displays will be no longer one-to-one but more dynamic and adaptive than what we have today. In fact, a recent study shows that people move across different displays on average 21 times per hour [7], showing the importance of efficient dynamic binding between computing devices and displays.

Such dynamic binding is best realized via *screencast*, which refers to sending the audio/video streams captured on computing devices over wireless channels to one or multiple larger displays in real-time for better viewing experience. Screencast seamlessly addresses the grand challenge of insufficient display real-estate of mobile and wearable devices, and has attracted significant attentions from both the academia and industry. For example, several open-source projects [1, 5] support one-to-one, one-to-many, or many-to-many screencast among smartphones, tablets, laptop/desktop computers. There are also several commercial technologies realizing screencast using dedicated hardware chips and software packages, including AirPlay, Chromecast, Miracast, MirrorOp, and Splashtop. The commercial screencast technologies are more widely deployed, compared to the open-source counterparts. Therefore, the chance for a user to run into a large display compatible with the commercial technologies is way higher. Nonetheless, the performance of the existing screencast technologies has not been studied in the literature, which may be attributed to their closed and proprietary nature.

In this paper, we conduct extensive experiments to quantify the performance of the screencast technologies using a real setup. We control 5 system parameters: frame rate, resolution, bandwidth, network delay, and packet loss rate. We adopt 4 objective performance metrics: video bitrate, graphic quality, end-to-end latency, and frame loss rate. Our in-depth analysis reveals several novel insights and leads to design suggestions to future screencast technologies. To our best knowledge, this paper is the first comprehensive measurement and comparison study on screencast technologies.

2. RELATED WORK

Screencast is one of the real-time remote graphical applications, and the performance of these applications has been studied in the literature. For example, Tolia et al. [8] and Lagar-Cavilla et al. [6] analyze the performance of VNC (Virtual Network Computing), and Claypool et al. [3] and Chen et al. [2] study the performance of cloud games. The quality of commercial screencast technologies has not received attentions in the research community. He et al. [4] conduct a user study on Chromecast with about 20 participants to determine the user tolerance thresholds on video quality (in PSNR), rendering quality (in frame loss rate), freeze time ratio, and rate of freeze events. The user study is done using a Chromecast emulator. Their work is different from ours in several ways: (i) we consider objective perfor-

MM '14, November 03 - 07 2014, Orlando, FL, USA.
Copyright 2014 ACM 978-1-4503-3063-3/14/11 ...$15.00.
http://dx.doi.org/10.1145/2647868.2654991.

Table 1: The Considered Parameters

Parameter		Values		
Content	Frame rate	15 fps	30 fps	**60 fps**
	Resolution	896x504	N/A	**1366x768**
Network	Bandwidth	4 Mbps	6 Mbps	**Unlimited**
	Delay	200 ms	100 ms	**0 ms**
	Packet loss rate	2%	1%	**0%**

Table 2: Specifications of Measured Technologies

Technology	AirPlay	Chromecast	Miracast	MirrorOp	Splashtop
Hardware/SW	Hardware	Hardware	Hardware	Software	Software
Connectivity	AP	AP	Wi-Fi Direct	AP/Internet	AP/Internet
Protocol	TCP	UDP	UDP	TCP	TCP
Proprietary	Yes	Yes	No	Yes	Yes

mance metrics, (ii) we consider multiple screencast technologies, and (iii) we use real setups in the wild for experiments.

3. METHODOLOGY

3.1 Screencast Technologies

We first introduce the considered screencast technologies.

- **AirPlay** is a proprietary protocol designed by Apple. AirPlay supports streaming audio, video, photos, and meta-data over wireless channels.
- **Chromecast** is a digital media player which is capable of directly streaming audio/video contents via Wi-Fi from the Internet or a local network. For screencast, a user can use Google Cast Chrome extension, which uses WebRTC API to cast content from the web browser or desktop to the Chromecast device.
- **Miracast** is a peer-to-peer wireless standard for screencasting over Wi-Fi Direct. Miracast-compatible devices can serve as Miracast senders and receivers. There are also Miracast adapters capable of rendering the screens through HDMI or USB ports.
- **MirrorOp** and **Splashtop** offer pure software solutions, which require the users to install proprietary applications at both the sender and receiver.

3.2 Content Types

We consider 9 content types in the following 3 categories.

- **Gaming**: including first-person shooter, racing, and turn-based strategy games.
- **Movie/TV**: including dialogue movie scene, car chasing movie scene, and talk show.
- **Interactive applications**: including Google street view browsing, slides editing, and web surfing.

For fair comparisons, we record the screens of different content types into videos. In particular, we extract one minute of representative video for each content type and concatenate them into a single 9-minute long video.

3.3 System Parameters

We consider two groups of parameters: content and network, which we believe impose direct and non-trivial impacts on screencast quality. Content parameters are related to the quality of source videos, including *frame rate* and *resolution*. We change the frame sampling rates to generate multiple videos, and set 60 fps as the default frame rate. We also vary the resolutions at 1366x768 (default[1]) and 896x504. For the latter case, we place the video at the center of the (larger) screen without resizing it. This is because we believe image resizing would cause loss of details and bias our results. Network parameters are related to the wireless network conditions. We use dummynet to control the *bandwidth*, *delay*, and *packet loss rate* (packet loss) of the outgoing channel of sender devices. The default bandwidth is not throttled, delay is 0 ms, and packet loss rate is 0%. In our experiments, a system parameter is set to different values while all other parameters are fixed at their default values. The detailed system parameters values are listed in Table 1, with the default values in boldface.

[1]iOS does not support 1366x768, and we pick the closest resolution, 1440x900, instead.

3.4 Experiment Setup

There are several components in the experiment: a sender and a receiver for each screencast technology, and a Wi-Fi AP, which is mandatory for all technologies except Miracast. The specifications of the screencast technologies are summarized in Table 2, and the detailed experiment setups are given below.

- **AirPlay.** The sender is a MacBook Pro running OS X 10.9.2, with a 2.4 GHz Intel Core i5 processor and 8 GB memory, while the receiver is an Apple TV. They are connected to the same Wi-Fi AP before the sender can discover, connect, and stream screens to the receiver.
- **Chromecast.** The sender is a Lenovo ThinkPad X240 notebook running Windows 8.1, with 1.6 GHz Intel Core i5 processor and 8GB memory and the receiver is a Chromecast dongle. The only way for screencasting using Chromecast is by Google Cast Chrome extension. Once the sender is connected to the Wi-Fi AP, it can discover and connect to any available devices on the same Wi-Fi network.
- **Miracast.** We use the Lenovo notebook as the sender. For the receiver, we use a Netgear Push2TV Miracast adapter. Miracast is based on Wi-Fi Direct and supported by Windows 8.1. As long as the receiver is placed within an acceptable range to the sender, Windows 8.1 provides a simple interface for screencasting the desktop to the receiver.
- **MirrorOp** and **Splashtop.** The Lenovo notebook serves as the sender, while a PC running Windows 7, with an Intel Core i7 processor as the receiver. To use these two services, a user needs to create an account, and run the sender and receiver programs on the respective machines. Once both machines are logged in, they can discover and connect to each other.

We note that some screencast technologies have been realized by multiple vendors, and we cannot cover all implementations in this paper. Instead, we pick a popular implementation for each technology, but our measurement methodology can be applied to other implementations.

3.5 Performance Metrics

We measure the following performance metrics that are crucial to screencast.

- **Bitrate**. The average amount of data per unit time transmitted from the sender to receiver.
- **Latency.** The time difference between each video frame is rendered at the sender and at the receiver.
- **Frame loss rate** (frame loss). The fraction of video frames that are not rendered at the receiver.

- **Video quality** (quality). The video quality rendered at the receiver compared to the original video captured by the sender. Both SSIM and PSNR are used.

3.6 Experiment Procedure

For each technology, we first connect the sender and receiver, play the video with diverse content types at the sender, and measure the four performance metrics. We repeat the experiment several times with different system parameters. To facilitate our measurements, we have added a unique color bar on each frame of the source content as their frame id, which can be readily recognized.

To measure the bitrate used by the screencast technologies, we run a packet analyzer at the sender to keep track of the outgoing packets during the experiments. For measuring the graphic quality, we direct the HDMI output of the receiver to a PC, which is referred to as the *recorder*. The Windows 7 PC is equipped with an Avermedia video capture card to record the videos. To measure the quality degradation, each frame of the recorded video is matched to its counterpart in the source video, using the frame id. Last, we calculate the SSIM and PSNR values. The frame loss rate can also be measured by matching the frames.

To measure the user-perceived latency, we direct the rendered videos of both the sender and receiver to two side-by-side monitors via HDMI. We then set up a Canon EOS 600D camera to record the two monitors at the same time. To capture every frame rendered on the monitors, we set the recording frame rate of the camera to 60 fps, which equals the highest frame rate in our parameter settings. The recorded video is then processed to compute the latency of each frame, by comparing the timestamps when the frame is rendered by the sender and receiver.

4. MEASUREMENT RESULTS

4.1 Overall Performance

We report the results under the default system parameters (see Table 1). Each experiment lasts for 9 minutes 18 seconds, with 33480 video frames. For each screencast technology, we first calculate the bitrate, latency, and video quality of individual video frames rendered by the receiver (i.e., lost frames are not considered). We then compute the mean and standard error across all video frames. We also derive the frame loss rate of each experiment. We plot the results in Figure 1, and make several observations. First, AirPlay and Miracast both lead to high bitrate and low latency, while Miracast achieves much lower frame loss rate. Second, although Chromecast incurs very low bitrate, it suffers from high latency and high frame loss rate. Third, Splashtop and MirrorOp achieve similar frame loss rate and video quality, but Splashtop leads to lower bitrate and latency. Last, screencast technologies other than AirPlay lead to roughly the same video quality in PSNR. Based on these observations, we conclude that: (i) Miracast is the best hardware-based screencast technology and (ii) Splashtop is the best software-based screencast technology under the default system parameters. Last, we mention that we omit the figure of quality in SSIM, because it shows almost identical trends as PSNR (Figure 1(d)).

We next study the performance of these screencast technologies under different conditions. In addition to the default condition, we define a *low workload* condition by reducing the frame rate to 15 fps, which may be sufficient for non-multimedia applications, such as email clients and office suites. We also define a *poor network* condition, by incurring 200 ms network delay and 2% packet loss rate, while capping the network bandwidth at 6 Mbps[2]. For these three conditions, we compute the mean performance metrics, and rank the screencast technologies on each metric independently[3]. We then plot the results in Figure 2. This figure reveals that: (i) Splashtop performs the best and balanced in general, and it is never ranked the last in all aspects, (ii) AirPlay and Miracast perform well in some aspects, but poorly in others, and (iii) Chromecast and MirrorOp result in inferior performance than other hardware-based and software-based screencast technologies, respectively. In summary, Miracast outperforms other hardware-based screencast technologies, and AirPlay suffers from high frame loss rate under low workload and poor network conditions. Splashtop outperforms other software-based screencast technologies.

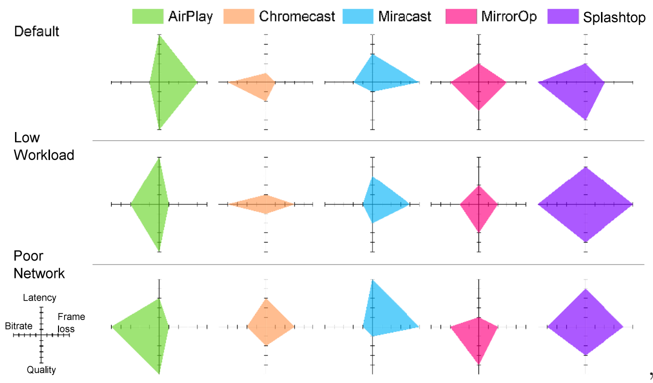

Figure 2: Ranks of different screencast technologies under different conditions. Ticks closer to the origins represent lower ranks (worse performance).

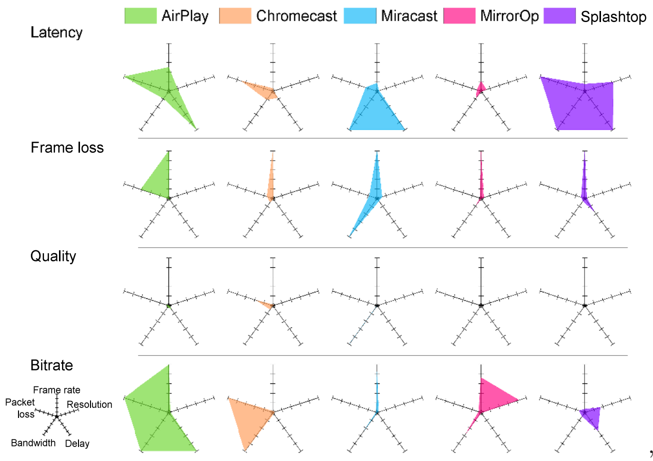

Figure 3: Susceptibility of different screencast technologies under different parameters. Smaller susceptibility (closer to the origins) means less vulnerable to dynamic environments.

[2]We also cap the network bandwidth at 4 Mbps, but several screencast technologies stop working under that constraint.
[3]We use PSNR as the quality metric, but SSIM leads to nearly identical ranking.

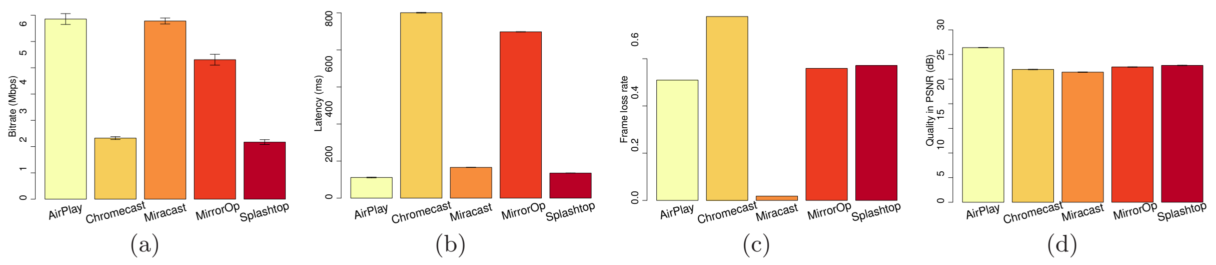

Figure 1: Performance of screencast technologies: (a) bitrate, (b) latency, (c) frame loss, and (d) quality in PSNR. The error bars represent the standard errors.

4.2 Susceptibility Comparison

We perform susceptibility analysis to quantify how much impact each system parameter incurs on each performance metric with different screencast technologies. For each screencast technology, we vary a system parameter while fixing all other system parameters at their default values. We repeat the experiment multiple times, and compute the mean performance of each experiment. For each metric, we then compute the ratio between the range (i.e., difference between the maximum and minimum) and the minimum, which is referred to as susceptibility. If the resulting susceptibility is larger than 1, we set it to be 1. Smaller susceptibility means more stable performance; larger susceptibility indicates higher dependency on some system parameters.

We report the resulting susceptibility in Figure 3, where ticks closer to the origins indicate smaller susceptibility. We make several observations. First, Splashtop and Miracast have relatively low susceptibility in general, except user-perceived latency. In particular, packet loss, bandwidth, and network delay affect their user-perceived latency the most. Fortunately, since screencast senders and receivers are likely to be collocated, severely poor network conditions are unlikely to happen. Therefore, Splashtop and Miracast still outperform over other screencast technologies. Second, the bitrate of AirPlay is extremely susceptible to network conditions and frame rate. That is, AirPlay aggressively adjusts its bitrate to cope with the dynamics of user demands (frame rate) and network conditions. Intuitively, this should give AirPlay a performance edge over other screencast technologies. It is however surprising to see that AirPlay performs the worst in dynamic network conditions (low workload and poor network in Figure 2), compared to other screencast technologies. A closer analysis on why AirPlay's bitrate adaptation strategy does not work well is one of our future tasks. Last, Chromecast and MirrorOp have relatively low susceptibility. In fact, as illustrated in Figure 1, the largest issues of Chromecast and MirrorOp are high latency and high frame loss rate, which unfortunately are not alleviated by better network conditions and lower resolution. Instead, the only system parameter that could ease these two issues is the frame rate. Therefore, Chromecast and MirrorOp are less suitable to higher frame rates (say, 60 fps).

4.3 Summary

We briefly summarize our main findings in the following.

- Splashtop performs the best and balanced in general with low susceptibility except user-perceived latency. The extreme network conditions are less common in screencast setups, where the senders and receivers are connected by one-hop ad-hoc link or via a Wi-Fi AP.
- Miracast performs the best among all hardware-based screencast technologies. Compared to Splashtop, Miracast's performance is slightly inferior, but Miracast adaptors are less expensive than dedicated receiver computers. Hence, Miracast has a better price/performance ratio than Splashtop.
- AirPlay aggressively adapts its streaming bitrate based on user demand and network conditions. It, however, suffers from high frame loss rate, indicating a suboptimal adaptation strategy.

5. CONCLUSION

In this paper, we have presented detailed measurement methodology and conducted the first comparison study on the state-of-the-art screencast technologies. This is the first measurement study using real screencast setups. Our methodology is also applicable to other screencast technologies, which may be developed in the future. We have found that Splashtop and Miracast perform the best. Our experiments also reveal some common shortcomings: (i) technologies other than AirPlay do not adapt the bitrate under different user demands and network conditions, and (ii) AirPlay's bitrate adaptation strategy however results in bad performance under low workload and poor network conditions. Future screencast developers should carefully design the bitrate adaptation strategy, and maybe leverage the low-level wireless characteristics via cross-layer optimization paradigm.

References

[1] S. Chandra, J. Boreczky, and L. Rowe. High performance many-to-many Intranet screen sharing with DisplayCast. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 10(2), Feb 2014.

[2] K. Chen, Y. Chang, H. Hsu, D. Chen, C. Huang, and C. Hsu. On the quality of service of cloud gaming systems. *IEEE Transactions on Multimedia*, 16(2), Feb 2014.

[3] M. Claypool, D. Finkel, A. Grant, and M. Solano. Thin to win? network performance analysis of the OnLive thin client game system. In *Proc. of ACM Workshop on Network and Systems Support for Games*, Nov 2012.

[4] Y. He, K. Fei, G. Fernandez, and E. Delp. Video quality assessment for Web content mirroring. In *Proc. of Imaging and Multimedia Analytics in a Web and Mobile World*, Mar 2014.

[5] C. Huang, K. Chen, D. Chen, H. Hsu, and C. Hsu. GamingAnywhere: The first open source cloud gaming system. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 10(1), Jan 2014.

[6] H. A. Lagar-Cavilla, N. Tolia, E. de Lara, M. Satyanarayanan, and D. O'Hallaron. Interactive resource-intensive applications made easy. In *Proc. of the ACM/IFIP/USENIX International Conference on Middleware*, Nov 2007.

[7] People swap devices 21 times an hour, says OMD, 2014. `http://www.campaignlive.co.uk/news/1225960/`.

[8] N. Tolia, D. Andersen, and M. Satyanarayanan. Quantifying interactive user experience on thin clients. *Computer*, 39(3), 2006.