StreamSmart: P2P Video Streaming for Smartphones Through The Cloud

Alessandro Gaeta, Sokol Kosta, Julinda Stefa, and Alessandro Mei
Department of Computer Science, Sapienza University of Rome, Italy
Email: surname@di.uniroma1.it

***Abstract*—Thanks to their power, the many sensors they embed, and their inherent connectivity to Internet, smartphones are certainly becoming the primary source of multimedia content and the main tool for content sharing.**

In this demo, we analyze the complexity of real-time video streaming among smartphone users. Firstly, we show that the traditional solution—a unique server receiving and dispatching all devices' content—suffers from scalability issues. Then, we present StreamSmart, a distributed system for real-time video streaming of smartphones, that leverages a virtual P2P network of smartphone software clones on the cloud. In StreamSmart, the captured content is forwarded from the sharing device to its own cloud clone, that in turn forwards it to the clones of other users. These latter transmit the content to the respective devices and, at the same time, contribute to further spread it to other possible clones in the network. We show that the StreamSmart system is highly scalable, responsive, and fault tolerant.

I. Introduction

Sharing multimedia content has become a crucial feature of our digital social life. Platforms like Facebook, Instagram, Dropbox, Picasa, and so on, all feature a *share* button in their system. Mobile devices like tablets, but most importantly, smartphones, are undeniably playing a main role in all this.

In this scenario we consider the following setting: Alice is at a concert, capturing a video with her smartphone, and wants to share it in real-time with some of her friends. Clearly there are many ways to achieve this. The first and most obvious one, is to use a central server that takes care of managing all users and distributes the video to all of Alice's friends. This traditional solution suffers from the obvious limitations of a single-server platform when managing multiple users (scalability, single point of failure, etc.). For example, recently, the famous streaming software Ustream[1], had to refund many boxing fans because their server was overloaded and the users could not watch the match[2]. An alternative solution is to build a smartphone P2P network, similarly to Skype [1], that uses NAT and firewall traversal techniques to interconnect the traditional computers. This solution is not feasible for two reasons: First, creating a P2P network of smartphones is very hard to get; second, Alice's smartphone would be highly overloaded. Recently in [2], the authors propose MicroCast, a system where a group of smartphone users in proximity of each other, and interested to watch the same video, cooperate to

[1]https://play.google.com/store/apps/details?id=tv.ustream.ustream
[2]http://www.mmajunkie.com/news/2013/01/ustream-exec-apologizes\-for-invicta-fc-failure-blames-record-demand-for-issues

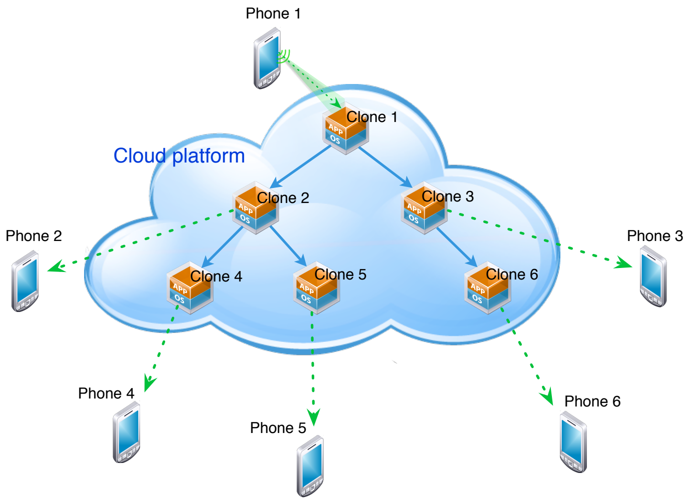

Fig. 1. The C2C architecture used for video streaming.

improve the video streaming experience. In this system, every smartphone uses simultaneously two network interfaces: the cellular interface to connect to the video server and the WiFi interface to connect to the other phones. Through a scheduling algorithm, each phone decides which parts of the video to download, and through a P2P connection distributes them to the others. Even though this is an improvement with respect to previous solutions, the assumption that users are in proximity of each other is very strong.

In this demo we propose to leverage the cloud so to achieve StreamSmart, an efficient and fault tolerant real-time P2P streaming system for smartphones. StreamSmart makes use of the C2C platform [3], [4], [5]. In this platform, every smartphone is associated with its own personal clone on the cloud, that can be used for computation offloading [6], [7], or data backup [8]. In addition, C2C interconnects the smartphone clones in a P2P fashion exploiting the high-speed network of the cloud. This allows the C2C platform to enable, not only computation offloading, but also *communication offloading* between smartphones: A large file that is to be sent from smartphone A to many other smartphones can be uploaded to clone A on the cloud. Then, any other smartphone will seamlessly download the file through clone A, without involving the real device A in the process.

StreamSmart leverages this last aspect of the C2C platform—*communication offloading*—to efficiently spread a videostream.

When Alice wants to stream a video and share it with her friends, she firstly uploads it to her cloud clone (see Figure 1). Then, Alice's clone informs the clones of the users authorized to watch the stream, that in turn forward the notification to the respective real devices. Each authorized user can decide to watch the video immediately, to watch it later by making his clone register the stream, or to simply ignore this streaming. Towards achieving an efficient and real-time system we consider different interconnection configurations among the clones. Our experiments show that by using a balanced tree the StreamSmart system yields very good performance in terms of both scalability and responsiveness.

II. Distributed video streaming for smartphones

The StreamSmart system consists of two sub-systems: The first, that resides on the smartphones, and the second subsystem that resides on the clones. A smartphone either captures a video (if it is the source of the stream) or shows it on the screen. The clones, from the other side, help in distributing it in real-time to the authorized users. We use the H264[3] video format to encode the data, the Real Time Streaming Protocol (RTSP[4]) as network control protocol, and the Real Time Protocol (RTP[5]) with UDP for data transmission.

The sender phone i) captures the video frames from the camera, ii) encodes them to H264 format, iii) encapsulates the encoded data in RTP packets, and finally, iv) sends the data through UDP to its own clone. This clone then, using the C2C connections, distributes the data to the other clones on the cloud, which in turn forward it to the respective real devices. Upon getting the RTP packets from the clone, a receiving phone decodes the H264 data and visualizes the video on the screen.

The heart of StreamSmart is undoubtedly the C2C network. The way the clones interconnect with each other can improve or degrade the overall performance of the system. We implemented and tested two strategies. The first one is a straightforward solution: The clones of the receiving smartphones connect to the clone of the source device, creating, in this way, a star graph. This configuration is analog to the traditional implementation of real-time single server streaming services. As expected, we observed that the clone in the center of the star was highly overloaded: Its CPU utilization exceeds 60% with as less receiving clones as 16.

In the second strategy, the clones create and maintain a balanced tree using a distributed algorithm. Figure 1 depicts an example of the resulting interconnections on a system with 6 clones. The nodes of the tree represent the clones, with the radix being the clone connected to the phone sharing the video streaming. Every clone interested to participate will join the system trying to maintain the tree balanced. Here we distinguish three types of clones based on their behavior: i) The *radix*, that gets the data from the phone and distributes them to its children; ii) The *leaf clone*, which gets the data from its parent and forwards them to the phone connected to it, and iii) The *internal clone*, which gets the data from its parent and distributes them to its children and to the phone connected to it. The CPU utilization of the radix node in this configuration drops down to less than 20% (as opposed to 60% for the star graph configuration), for the internal nodes is lower than 10%, and for the leaf nodes is lower than 5%.

[3]http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

[4]http://tools.ietf.org/html/rfc2326

[5]http://tools.ietf.org/html/rfc1889

An important factor to consider when building real-time video steaming systems is the user perceived video quality. In StreamSmart, this is strictly correlated to the interconnection architecture of the cloud clones: It determines the number of hops between the source of the stream and the "spectator" users. The bigger the number of hops, the higher the probability of packet loss and transmission delay. When the clones are connected as a star graph, the distance between the sender smartphone and every other phone is minimal, while in a balanced tree configuration the distance is variable, depending on the tree level that a clone belongs to.

We estimate the goodness of our system according to the standards proposed in [9]. According to these standards, the packet loss should be no more than 5%, and video latency no more than 4-5 seconds. Our experiments show that the StreamSmart system fully complies with the required standards for tree depth of up to 6 levels. Adding a 7^{th} level makes the users connected to the leaf clones experience a poor video quality. To avoid this, once the tree is full, we force it to grow in width, making the clones stay closer to the radix. In this case, the system starts experiencing problems only when the degree of the nodes becomes very high, that translates in a high overload of the relay clones.

III. Discussion

StreamSmart is a distributed and participatory system. It becomes thus very important that the clones within the system help in spreading the video-stream throughout the network. In this scenario, the following two questions raise naturally: 1) "Should the clones of *non* interested users be involved in distributing a certain content?", and 2) "Can we ensure that the clones participating in the system will behave correctly?".

To answer the first question, there are several things to be taken into consideration: (a) Privacy requirements of the source—the higher it is, the less likely the source wants the video to be spread through clones of unknown people; (b) Constraints for the link creation among clones—If they are bootstrapped by, e.g., a user's social network, it certainly becomes very hard for clones of non-friend users to interconnect to each other.

To answer the second question, the selfishness of the users, and the respective clones, has to be considered. In this case, mechanisms like [10], or Nash Equilibrium protocols like [11] can be exploited to incentive users and respective clones to behave correctly.

As future work we are exploiting other interconnection architectures within the C2C network, with the aim to improve the overall quality of the system. Recent studies show that the degree of separation between nodes within social networks is roughly 4 [12], making the social graph a perfect candidate for our real-time video streaming architecture. In this configuration, with high probability every pair of phones would be close to each other. In addition, each clone would not only send data to multiple clones, as in the tree configuration, but would also receive data from multiple clones. These two simple improvements could alleviate the transmission delay and help mitigate the packet loss effect. Nonetheless, we are aware that the social-network has its own pitfalls. Probably, the most worthy to mention is the one deriving from the so called *hubs*—popular nodes that present a high degree within the network. These nodes would have to handle many users, and as a consequence, would be highly overloaded. A possible way-around is to limit the number of clones served by a hub to a maximum number of neighbors. When selecting the clones to serve, it is more preferred that a hub gives precedence to neighbors with low degree—those with high degree will eventually find another path from which to get the video.

IV. Demo setup

We demonstrate our StreamSmart prototype using four Samsung Galaxy S Plus devices and 32 Android-X86 software clones deployed on Amazon's EC2 public cloud. The first group of 16 Android clones will be connected as a star graph, while the second group of 16 Android clones will be connected following the balanced tree algorithm. To each C2C configuration we associate two Samsung phones.

The user (the demo presenter or a possible volunteer) will use one of the phones (per each configuration) to stream a video through the C2C system. Meanwhile, all the clones will automatically participate to the video streaming, getting the video and waiting for phone connections. The user will use the other phone (per each configuration) to watch the video, alternating the clones the phone connects to. During the demonstration we will show, in real time, the CPU overload of the clones as well as the video quality of the receiving phone.

Acknowledgments

This work has been performed in the framework of the FP7 project TROPIC IST-318784 STP, which is funded by the European Community. The Authors would like to acknowledge the contributions of their colleagues from TROPIC Consortium (http://www.ict-tropic.eu).

References

[1] S. A. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," in *IEEE infocom*, vol. 6, 2006, pp. 23–29.

[2] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou, "Microcast: Cooperative video streaming on smartphones," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012.

[3] S. Kosta, V. C. Perta, J. Stefa, P. Hui, and A. Mei, "Clone2Clone (C2C): Peer-to-Peer Networking of Smartphones on the Cloud," in *5th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '13)*, 2013.

[4] S. Kosta, V. C. Perta, J. Stefa, P. Hui, and A. Mei, "CloneDoc: Exploiting the Cloud to Leverage Secure Group Collaboration Mechanisms for Smartphones," in *Proc. of IEEE INFOCOM 2013*, 2013.

[5] M. V. Barbera, S. Kosta, J. Stefa, P. Hui, and A. Mei, "CloudShield: Efficient anti-malware smartphone patching with a P2P network on the cloud," in *Proc. of IEEE P2P 2012*, 2012.

[6] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading." in *Proc. IEEE INFOCOM 2012*, 2012.

[7] M. V. Barbera, S. Kosta, A. Mei, V. C. Perta, and J. Stefa, "CDroid: Towards a Cloud-Integrated Mobile Operating System," in *Proc. of IEEE INFOCOM 2013*, 2013.

[8] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing," in *Proc. IEEE INFOCOM 2013*, 2013.

[9] T. Szigeti and C. Hatting, *End-to-end qos network design*. Cisco Systems, 2005.

[10] W. S. Lin, H. V. Zhao, and K. R. Liu, "Incentive cooperation strategies for peer-to-peer live multimedia streaming social networks," *Multimedia, IEEE Transactions on*, vol. 11, no. 3, pp. 396–412, 2009.

[11] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "Bar gossip," in *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX Association, 2006, pp. 191–204.

[12] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna, "Four degrees of separation," in *Proceedings of the 3rd Annual ACM Web Science Conference*. ACM, 2012, pp. 33–42.