Ubiquitous Social Cams

O. Gaggi, N. Moretti, C. E. Palazzi
University of Padua, Padua – Italy
{gaggi | cpalazzi}@math.unipd.it

***Abstract*—Web 2.0 is evolving and offering services based on the wide popularity of smartphones and the possibility of gathering data ubiquitously from these mobile devices; this new paradigm is often referred to as Web^2. In this paper, we hence present and discuss a Web^2 application designed to enable users to interact with remote smartphones' cameras to receive a video generated in real time. In essence, users can individuate on a map which mobile cameras are available in a location of interest and request the remote user to generate and send back a short video of the surrounding environment or event. We discuss technical issues related to implementing such a service and the solutions we devised to address them. Finally, we also present experimental results we obtained from a preliminary testbed evaluation that encourages the prosecution of this work.**

Keywords-Camera Sharing; Mobile Application, Smartphone; Web Squared

I. INTRODUCTION

The Web 2.0 paradigm fostered the creation of applications based on collective knowledge and intelligence. Systems such as YouTube, Facebook, Twitter, Wikipedia, Amazon, Foursquare, Flickr and Picasa allowed users to share a huge amount of information in real time. One of the fundamental principles of Web 2.0 is that the service improves its quality with the growth of the number of people using it [1]. A little part of value is explicitly added to Web 2.0; rather, Web 2.0 systems allow the aggregation of users' data by default as an effect of the normal application uses. Therefore, users can use selfish Web 2.0 applications for their own purposes but they automatically add a collective value.

The Web is now moving from Web 2.0 toward different possible evolutions [2]-[4]. One of the most interesting is certainly represented by *Web^2* (*Web Squared*), which derives from the exponential growth of the Web based on the integration between the collective intelligence of Web 2.0 and the use of new interconnected, sensor-equipped mobile devices [5]. Indeed, any object can be associated with some data (location deriving from the GPS, properties associated to its barcode, etc.) creating what is called the *information shadow* of the object. Through the Web^2 paradigm, sensors on mobile devices (e.g., smartphones) allow the combination of the real world with objects' information shadow so as to generate new information and foster innovative services.

The Web^2 paradigm is based on the mobile revolution, which has made smart mobile devices ubiquitously present in our cities (and pockets). Popular technological devices such as smartphones, tablets and netbooks have digital sensors allowing to easily determine their position and to retrieve useful position-related information from the Internet. Furthermore, they have cameras and microphones which enhanced the development of new form of interaction with the Internet collective intelligence.

Even if potentially powerful, the ideas at the basis of Web^2 need to be supported by practical solutions and implementation. This requires the testing of the technological devices in order to verify the feasibility of these paradigms with current technology.

We present *Kweekpeek*, a Web^2 camera sharing application that enables users to interact with the smartphone cameras of other remote users. Different from other similar applications (e.g., Ustream [6]), Kweekpeek allows video consumers to request in real time a live video streaming from a specific location. In essence, Kweekpeek keeps track of the position of registered smartphones through GPS (of course the anonymity of the data provider has to be preserved). Users can check on a map which mobile cameras are available in a location of interest and request to generate and send back a short video of remote environment or event.

The main contributions of this work are hence both the presentation of a novel Web^2 proof-of-concept application and the discussion of related technical criticalities and solutions.

The rest of this manuscript is organized as follows. Section II describes two possible case studies. Kweekpeek and its development are explained in Section III and Section IV, respectively. Section V discusses preliminary experiments we run and Section VI concludes this paper.

II. CASE STUDIES

We briefly discuss two possible scenarios where a camera sharing application could be employed.

Scenario #1: Leisure. Humans are social beings; they generally enjoy the interaction with each other [7]. For instance, they tend to choose shops, bars, clubs and restaurants that are popular. To avoid the disappointment of having chosen the wrong destination to go out, people would like to use an application able to provide them a glance of the remote location they are interested in through a short video generated by somebody already there. Similar, people may want to take a look at concerts or sport matches to see in real time, for instance, the supporters scenography. These are two simple but meaningful examples; there is a huge number of possible cases where people may be interested in receiving a short video from a remote location related to leisure activities [8]-[11].

Scenario #2: Safety. Consider a crisis scene in a city, e.g., a street accident or a terrorist attack. In this scenario, it would be useful to provide first responders with real-time pictures/videos of the emergency while still driving toward the crisis area. Devices utilized for this purpose could be security cameras in the area or any other camera-endowed device (e.g., a smartphone) handled by people in proximity of the emergency area. Both commands to activate the device and generated pictures/videos can be sent through the vehicular network directly to the vehicle of first responders, or reach them through the Internet and the cellular network [12]-[15]. In any case, elements that are typically considered of disturbance in emergency situations e.g., people stopping by to see the accident and even take pictures at it, can turn out to be of help.

III. KWEEKPEEK: A CAMERA SHARING APPLICATION

Kweekpeek aims at enabling users to share oneself smartphone camera and to use cameras of other remote users. It represents the union between four different technologies: webcams, social networks, mobile devices and geo-localization. Kweekpeek provides a Web server that shows a map of the geo-localized available devices (see Fig. 1); the user chooses one on the devices on the map to receive video data from the camera of that remote device. In essence, the system embodies a cam social network, made of mobile smartphones registered for the service.

Figure 1. Available cams geolocalized and displayed on map.

Every kweekpeeker (i.e., a system user) can choose any registered smartphone from a map by sending a share request. If the request is accepted the chosen kweekpeeker records his/her surroundings with the camera on his/her smartphone and shares it.

Finally, we are aware that the success of Kweekpeek also depends on features we have partially addressed in this work and are going to deeply investigate in future extensions:

- the system must be multiplatform (to cover the highest number of possible users);
- there must be a set of rules to encourage and reward users to accept cam sharing requests;
- the system must prevent incorrect behaviors such as inappropriate or offensive videos;
- the system have to implement efficient anonymity and authentication policies;
- the system must pay attention to the trade-off between video quality and network usage.

IV. APPLICATION DEVELOPMENT

Kweekpeek is a distributed system in which a user can use his/her phone to capture, send and receive video streams. For this reason, the system is based on a Web server to control and maintain the system. The domain *kweekpeek.com* has been registered for later implementation of the possibility to share webcam streams even directly from the Web site.

The Web site is based on HTTP Web 2.0 Server Meteor [16] that allows push notifications using the Comet Model Long Polling technique that allows a server to hold a client request until data for a response is available instead of having a client periodically sending the same request, with the server responding even if no data is available, until satisfying data are received [17]. Moreover it uses the open source framework concrete5 [18] as Content Management System. The result is shown in Fig. 2.

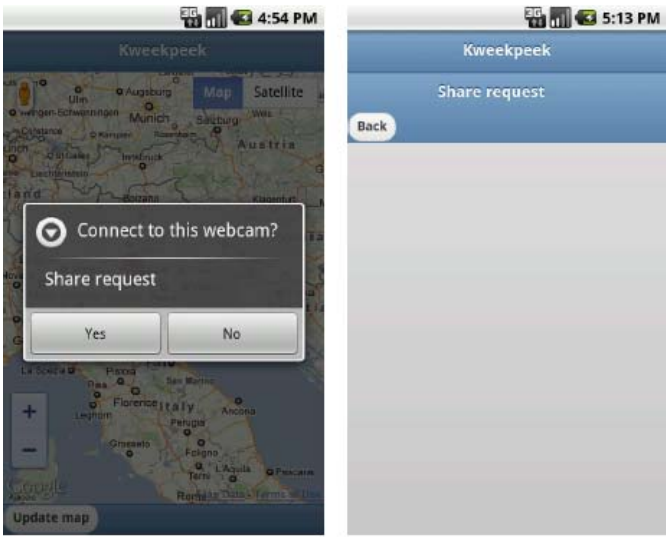

Figure 2. Web site accessed from a mobile device.

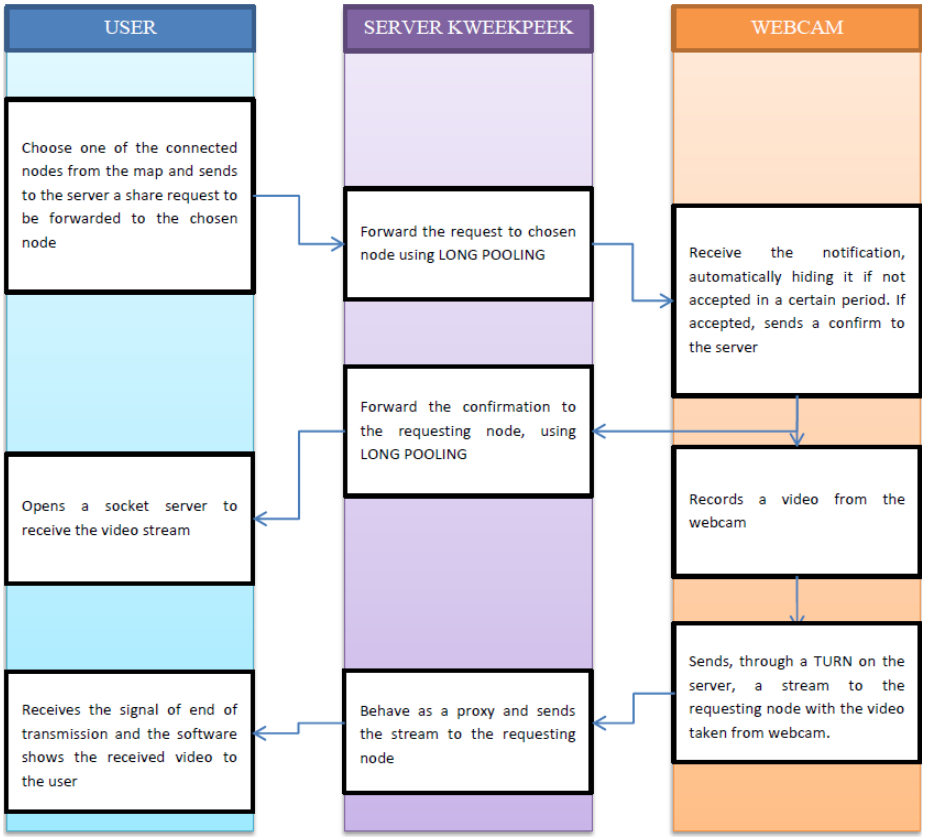

Figure 3. Client-server communication.

The communication between the Web server and the mobile devices is managed by an API package, specifically developed for kweekpeeker, that uses the standard JSON encoding to handle mobile requests. A connection between two nodes is established in 5 steps (see Fig. 3):

1. the user chooses a node from the map depicted in Fig. 1 and send a *share request* to the server;
2. the server forwards the request to the chosen node, using *long polling*;
3. the chosen node receives the notification and, if accepted, sends a confirmation to the server;
4. the server forwards the request to the chosen node, using *long polling*;
5. two connections are separately established between the two nodes and the server using Web sockets.

For all the duration of the video stream, the server behaves as a proxy to the requesting node. On first connection, each smartphone receives from the server an identifier that will be used in all following connections. This identifier will never be revealed to other nodes. The server visualized position of the available cams into a map (see Fig. 1).

The utilization of a central server has been preferred to a P2P solution directly connecting nodes as a server ensures the possibility to address in an easier and more efficient way to provide anonymity and authentication. Moreover, through the utilization of a set of local servers controlled by a central one, the system results scalable while preserving the complete control over the whole system.

As already discussed in Section III, one of the important factors for the success of this project requires that the mobile interface must be available in all platforms, i.e., iOS (Apple),Android OS (BlackBerry), WebOS (HP), WP7 (Microsoft), Symbian and Bada (Samsung), to allow to share the major number of mobile cams. For this reason the client was implemented using PhoneGap [19] an open source mobile framework, that allows to deploy part of the code, independently from the final platforms, using Web standards.

In order to manage requests from a big number of users, the system organizes users into different levels according to the number of requests they have accepted (and served) or asked, i.e., the more requests a user has served, the more requests he/she can made. The system uses the ratio (requests made/ requests accepted) to move a user up and down through different levels. Every level allows a maximum of weekly requests (increasing level by level).

Finally, we ask users that have made a request a feedback at the end of each shared stream share, so that the receiver of the stream can leave a positive or negative mark. In case of multiple negative reviews for requests served by a single node, the system blocks that node for a short period of time.

A. PhoneGap

PhoneGap is a framework that allows the creation of mobile applications from Web applications, i.e., using Web standards like HTML5, CSS3 and JavaScript. PhoneGap does not translate the code into the native language of the platform for which the application is compiled, but encapsulates it together with the Web engine *webkit* so that it can be executed without the help of a browser [20].

Moreover, PhoneGap provides some APIs to access the device features, e.g., accelerometer, wireless connection, media playback, notifications, storage system, etc.

In essence, PhoneGap is a *wrapper*: it requires resource to run the webkit engine in addition to the resources needed for the application itself. On the other hand, it has the advantage of supporting almost all mobile platforms, i.e., Android, iPhone OS, BlackBerry, WebOS, Symbian, WindowsPhone 7 and Bada, without requiring to know in advance which platform will be used, but exploiting all the new features offered by HTML5 and CSS3. In addition, well known JavaScript libraries, like jQuery, can be used without any problem.

Moreover, PhoneGap also permits the integration of native code for unsupported features. In this project, PhoneGap was used to develop the user interface, in particular:

- to integrate the visualization of the available cameras into the Google Maps service;
- to manage the playback of received movies (see Fig. 4);
- to develop menus, settings and information.

B. Android Development

Since the current state of implementation of the PhoneGap framework does not cover all the features offered by our system, the development of Kweekpeek has required the writing of some modules, coded separately for each platform, to manage video recording and data transmission. We discuss here the implementation for the Android platform, which is the only one complete at the moment.

Each time a user accepts a request, he/she has to record a video of the situation in which he/she is present. We set the duration of this video to 20 s, since we noted that this is a good trade-off between the need to show the surrounding in detail and the amount of time asked to the user to serve a request.

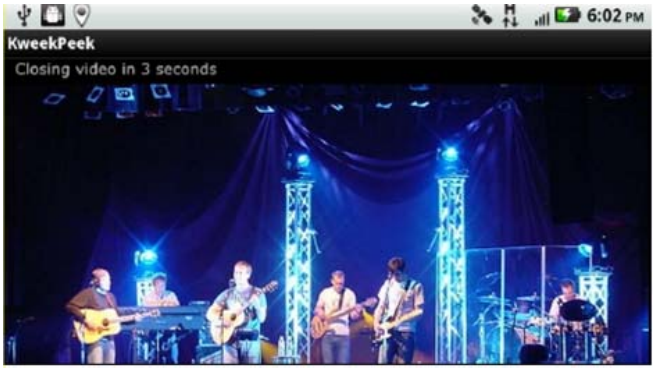

Figure 4. Screenshot of a video shared through Kweekpeek.

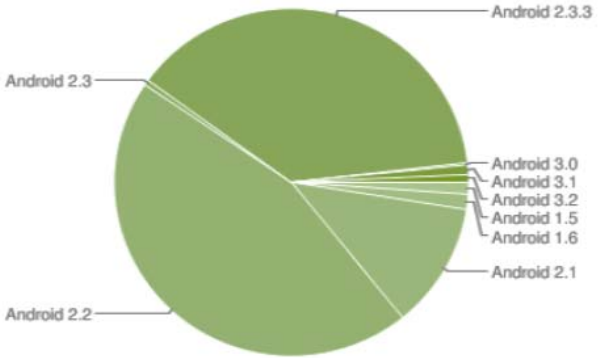

Figure 5. Android OS Fragmentation [21].

Once the recording ends, the user sends the movie to the server. Future developments will provide the possibility, for the user that serves the request, to autonomously choose the duration of the recording.

The recorded video stream is then compressed using the codec H264 on MPEG4 format. The current system provides 20 s of video at 15 fps with a resolution of 320 x 240 px. In this way the size of the video is generally less than 1MB.

We did not adopt a tool like Sipdroid [22] to generate a *real-time streaming* of recorded video for its compatibility only with Android 2.3 or higher, whereas many available devices still use previous versions of this platform (see Fig. 5). Furthermore, we realized that the average connection speed is insufficient for the video codec required by Sipdroid, while H.264 allows better performance, i.e., video of low quality (but clearly understandable) and low bandwidth.

To further reduce bandwidth consumption, the generated video has been compressed before transmission. Clearly, the higher the compression ratio, the lesser the bandwidth required, but also the more computational and energy power is needed. Current standards for vide compression such as H.264 and VP8 provide both efficient compression and low bit rates. In our experiment we preferred H.264 for its lower bandwidth usage and better video quality compared to VP8 [23]. Moreover, H.264 is the default codec for video recording in Android, thus simplifying Kweekpeek implementation.

Finally, the current position of each node on the Web site is maintained updated with two Web services which send and receive push notifications about nodes positions.

V. Experimental Assessment

To test its feasibility, Kweekpeek was implemented and tested on a Motorola Milestone 2 A953 with Android v2.3.4 operating system, 1 GHz clock rate and 512 MB of RAM. As anticipated, even if the device allows a maximum video recording resolution of 1280 x 720 px, the experiments were run using a video recording resolution of 320 x 240 px encoded with H.264 to reduce the video size.

The fps rate that has to be generated also depends on the video purpose. For instance, when a video is created for communication via sign language, it is recommended to have at least 21 fps, whereas 5 fps would be enough to perceive audio

and video synchronization in regular situations [24]. In our experiments we have chosen to use 15 fps as this represent a possible tradeoff solution between to ensure low bandwidth requirement without compromising the video quality. However, nothing impedes to use different settings for the fps or to let users chose their preferred configuration.

Tests provided a positive feedback on the deployment of Kweekpeek: the application actually allows to identify smartphones with cameras on a map and to receive back videos from the chosen remote location. Video duration can be modified; clearly the video duration impacts on the video size. In our experiments, we have considered different video size durations (i.e., 10 s, 30 s, 60 s, 150 s) and recorded 10 different sample videos for each of them. For each of these videos, we have measured the resulting video size and corresponding statistical values. We report measured values, classified by video duration, in Table I. Video sizes go from a minimum of *circa* 390 KB (for a 10 s video) to a maximum of *circa* 6334 KB (for a 150 s video). Both are acceptable values for Internet transmission.

TABLE I. VIDEO FILE SIZE (KB) DEPENDING ON ITS DURATION

	Video Duration			
	10s	***30s***	***60s***	***150s***
Min	390.06	1178.50	2358.71	5896.73
Max	444.35	1284.83	2541.97	6333.94
Avg	412.00	1227.35	2445.45	6179.28
St.dev	14.94	44.13	89.91	245.06

VI. CONCLUSION AND FUTURE WORK

We presented and discussed a Web^2 application, named Kweekpeek, designed to enable users to individuate and use smartphones' cameras to receive a video from a remote location. Kweekpeek keeps track of the position of registered smartphones through GPS data (of course the anonymity of the data provider has to be preserved). In essence, users can check on a map which mobile cameras are available in a location of interest and request the remote user to generate and send back a short video of the surrounding environment or event.

Beside, architectural extensions discussed in Section III, in the future we also aim at improving the client application in several directions:

- improving the usability of the user interface;
- allowing the user to save the videos received in chronological order;
- adding requests auto-acceptance functionality for permanent webcams.

ACKNOWLEDGMENT

Partial financial support for this work is provided by the MIUR/PRIN ALTER-NET and the UNIPD/PRAT Web Squared projects.

REFERENCES

[1] T. O'Reilly, What is Web 2.0? www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-Web-20.html?page=1

[2] T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web", Scientific American, 279, 2001.

[3] S. Ferretti, M. Furini, C. E. Palazzi, M. Roccetti, P. Salomoni, "WWW Recycling for a Better World", Comunications of the ACM, ACM, vol. 53, no. 4, Apr 2010.

[4] T. O'Reilly, J. Battelle, Web Squared: Web 2.0 Five Years On. Web 2.0 summit (2009).

[5] G. Calma C. E. Palazzi, A. Bujari, "Web Squared: Paradigms and Opportunities", in Proc. of the International Workshop on DIstributed SImulation & Online gaming (DISIO 2012) - ICST SIMUTools 2012, Desenzano, Italy, Mar 2010.

[6] Ustream,You're On, www.ustream.tv/

[7] C. E. Palazzi, "Buddy-Finder: A Proposal for a Novel Entertainment Application for GSM", in Proc. of the 1st IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'04), GLOBECOM 2004, Dallas, TX, USA, Nov 2004.

[8] M. Roccetti, G. Marfia, A. Semeraro, "Playing into the Wild: A Gesture-based Interface for Gaming in Public Spaces", Elsevier Journal of Visual Communication and Image Representation, Elsevier, vol. 23, n. 3, Mar 2012.

[9] S. D. Fuhrman, "Beyond Webcams: An Introduction to Online Robots", Presence, MIT Press, vol. 11, n. 5, Oct 2002.

[10] I. Smith, "Social-Mobile Applications", IEEE Computers, IEEE Computer Society, vol. 38, n. 4, Apr 2005.

[11] S. Ahmed, A. Khan, I. Babar, "Monitoring Detection and Security Maintenance using WMS-Webcam Mobile Surveillance", in Proc. of 3rd IEEE International Conference on Emerging Technologies (ICET 2007), Islamabad, Pakistan, Nov 2007.

[12] C. E. Palazzi, "Interactive Mobile Gaming over Heterogeneous Networks", in Proc. of the 5th IEEE/ITI International Conference on Information and Communications Technology (ICICT 2007), Cairo, Egypt, Dec 2007.

[13] M. Roccetti, G. Marfia, A. Amoroso, "An Optimal 1D Vehicular Accident Warning Algorithm for Realistic Scenarios", in Proc. of IEEE Symposium on Computers and Communications (ISCC'10), Riccione, Italy, Jun 2009.

[14] A. Amoroso, G. Marfia, M. Roccetti, "Going Realistic and Optimal: A Distributed Multi-Hop Broadcast Algorithm for Vehicular Safety", Computer Networks, Elsevier, vol. 55, n. 10, Jul 2011.

[15] G. Marfia, M. Roccetti, "Vehicular Congestion Detection and Short-Term Forecasting: A New Model with Results", IEEE Transactions on Vehicular Technology, IEEE Vehicular Technology Society, vol. 60, n. 7, Sep 2011.

[16] Meteor Web 2.0 Server, http://meteorserver.org/

[17] G. Wilkins, Comet Is Always Better than Polling, http://cometdaily.com/2007/11/06/comet-is-always-better-than-polling/ Nov 2007.

[18] CMS Concrete5, http://www.concrete5.org/

[19] Phone Gap Framework, http://phonegap.com/

[20] The WebKit Open Source Project, http://webkit.org/

[21] Gadget Venue, http://gadgetvenue.com/

[22] Sipdroid, Free SIP/VoIP client for Android, http://sipdroid.org/

[23] P. Seeling, F. H. P. Fitzek, G. Erli, A. Pulipaka, M. Reisslein, "Video Network Traffic and Quality Comparison of VP8 and H.264 SVC", in Proc. of the 3rd ACM Workshop on Mobile Video Delivery (MoViD 2010), Florence, Italy, Oct 2010.

[24] J. Scholl, P. Parnes, J. D. McCarthy, A. Sasse, "Designing a Large-Scale Video Chat Application", in Proc. of the 13th annual ACM International Conference on Multimedia (Multimedia 2005), Singapore, Nov 2005.