Peer to Peer Media Management for Augmented Reality

Raimund K. Ege

Dept. of Computer Science
Northern Illinois University
DeKalb, IL, USA
ege@niu.edu

***Abstract*—Immersion into rich multimedia is now the norm in today's Internet. Combining multiple streams of textual, audio and media data from a variety of sensors and sources, allow the presentation of a world that is almost "real". Users equipped with portable and wearable devices can become consumers and contributors to an augmented reality. In this paper we describe a general software architecture for an augmented reality immersion network based on crowd sourced media gathering and distribution. We describe a prototypical implementation with cloud and mobile components to establish a secure sharing network, to coordinate and to synchronize the media streams. Joining the content sharing network is subject to peer-to-peer trust management to protect the content and the participants.**

Keywords-Android; augmented reality; multi-media content delivery; securing trust; peer-to-peer systems;

I. INTRODUCTION

What was a personal digital assistant became a smartphone and now an ever-present wearable device: with computing power, display and recording capabilities, and – foremost – with broadband connectivity. The days of just calling and texting on a smart phone are gone: we now watch TV shows, check the world-wide-web, or play games. We can participate in a host of social applications that are rich in multimedia exchange.

Modern mobile devices feature multiple input sensors, like cameras, and advanced geo-location positioning systems that us GPS, cell tower triangulation, compass and accelerometers. Users of such mobile devices are not limited to media consumption, but are allowed to become an active player in the production and sharing of media. The computing power and network connectivity enable the provision of peer-to-peer (P2P) content delivery networks: rather than just down- or up-loading media to one site, media can be shared in such P2P network at a much higher throughput, i.e. no single source bottleneck, and without central control, i.e. big brother registration. The aim of our research is to allow the forming of very large P2P content sharing networks, without central control, but with provisions that instill a degree of trust into the participants.

In this paper we describe architecture for an augmented reality immersion network based on crowd sourced media gathering and distribution. We describe an application framework with cloud and mobile components to establish a secure sharing network, to coordinate and to synchronize the media streams. Joining the peer-to-peer (P2P) content sharing network is subject to trust management to protect the content and the participants.

Possible application scenarios for this technology span from massively online multi-player games to first responder support gear for emergency personnel. In a multi-player game scenario, participants can wander a partially populated game room, that is further augmented with virtually-real objects and events that need to be handled by groups of game players. Each player carries a smart phone which transmits its sensed data (video, location) to others, and receives a coordinated and merged augmented reality view of the game room. In a first responder scenario, support gear collects and transmits sensor data to members of the response team, and receives a coordinated and merged augmented reality of the emergency scenario.

Section 2 gives some background on smartphone technology, P2P content delivery and sharing networks, and security issues such as access control, identity and trust management. We also relate our work to current research. Section 3 discusses how to manage and secure shared content, specifically which elements of security to ensure confidentiality, integrity and availability are available to mobile platforms, with a specific focus on what is available to Android smartphones. Section 4 elaborates on our P2P content sharing model, especially on how our approach defines and gauges trust, and how such trust is maintained, secured and shared in a central-server-less P2P environment. Section 5 outlines our prototype implementation with Java peers, including peers running on Android smartphones. The paper concludes with some lessons we learned and our future perspective.

II. BACKGROUND

Personal digital assistants have come a long way in recent years. The current crop of smartphones is just a stepping stone in the advance of digital devices that enable individuals to compute and connect. Wearable connected

 ISBN: 978-1-61208-404-6

computing devices, such as wristwatches and even eye glasses (Google GLASS) are available. The focus is shifting from computing and storage capabilities on these devices to connectivity and multi-media input and output components.

Connectivity capabilities are typically wireless and include high-bandwidth cellular (4G, LTE) and WLAN (IEEE 802.11) connections, plus lower-bandwidth near field connections (Bluetooth, NFC, etc.). Transmission rates in the multi megabits per second range and latency rates in the sub millisecond range are currently quite standard. Multi-media I/O components include high-definition screens and video cameras, high-fidelity speakers and microphones. Plus components to determine device location, position, and attitude: GPS, accelerometers, compass, etc.

Consider the Google Maps application on a smartphone: the smartphone acquires its location via GPS, sends the location to a Google server. The server responds with appropriate map data which is displayed on the phone. As the user moves, the map data is updated. Such a simple example of augmented reality can be further improved by adding real-time traffic data from traffic sensors. Moreover, data gathered from other smartphones can augment the display with a multitude of other useful data, as in the Waze (www.waze.com) navigation application. Augmenting map data with imagery from satellites and street based cameras (Google Street View) is already common practice. Adding video and other sensor data from nearby smartphones is the logical next step.

Augmented reality provides a live view of a physical, real-world environment. It can be direct or indirect. Its elements are supplemented or augmented by computer-generated input from sensors such as sound, video, graphics or location data. While this field of research has quite a long history [1], only recently has the computing and bandwidth capabilities enabled truly wide acceptance [2] [3]. Key elements of such crowd-sourced augmented reality are real-time coordination of sensor data and establishment of authenticity and trust in the participating peers. Coordination of the data is achieved via ever precise location information, coupled with attitude references. Current locating sensor and accelerometer technology has shrunk and is available in state of the art smartphones.

Access control, trust and digital rights management is essential. Access control is common place in many applications. A server maintains a database of user and account information. A user gains access to the system by providing a user id with additional security information, typically a password. Once authenticated, the user is "trusted", i.e. is allowed to participate in the system's mission. The information stored by the server can include the users past history of participation, which in turn can be used to augment the level of trust in the user. Other users might contribute to the trust evaluation by submitting feedback on others. The level of trust might determine the level of participation a user is allowed, e.g. users with a low level of trust might be able to consume content, while users with a high level of trust might be able to contribute media.

Many modern systems outsource their central access control to an external provider. In a centralized system central access control makes sense: OpenID [4] is an example. OpenID providers maintain identity information and allow users to choose which and when to associate information with their OpenID that can be shared with sites they visit upon request. With OpenID, password information is passed to the identity provider which verifies and then confirms the identity of a user.

Peer-to-peer systems lack a central authority: peers need to collaborate and obtain services within an ad hoc environment where little trust exists. All peers collectively have to manage the risks involved in a collaboration: incomplete knowledge and little prior experience is the norm. Typical approaches address this uncertainty by developing and establishing trust among peers. Trusted third party systems [5] or self-regulating systems with community-based feedback [6] are ways to build trust.

In today's collaborative and complex world, a peer can both protect itself and at the same time benefit only if it can adjust and react to new peers dynamically and enforce access control via flexible and proper privileges. Management of trust helps minimize risk and ensures the network activity of benign entities in distributed systems [7].

Many secure content delivery systems focus on digital rights management, especially for peer-to-peer and mobile systems. Several schemes have been introduced: OMA DRM [8] – promulgated by the Open Mobile Alliance industry consortium – attempts to standardize a framework to secure media for mobile devices. Public key infrastructure (PKI [9]) style certificates are employed that contain and authenticate public keys to protect media. While we also use PKI, our intention is to go further in that we do not want to require absolute certainty of access right, but rather allow building of graduated trust which enables graduated access control to digital media.

In our prior work we focused on how trust can be quantified [10], and how trust can be managed securely [11] by peers who participate in a P2P content sharing network. In this paper we combine these approaches and add the dimension of combining multiple peers' perspectives into one augmented reality.

III. SECURING AND MANAGING SHARED CONTENT

The key to successful sharing of content is its security. While sharing implies to let others consume content, it has to be done in a safe and secure environment. The conventional CIA triad, i.e. confidentiality, integrity and availability, also applies in the mobile content sharing context. Shared media must not be consumed by un-authorized peers, it must stay confidential to only authorized peers. Shared media should not be altered, its integrity must be preserved. And the media, plus the data needed to make access decisions must be available to authorized and trusted peers.

 ISBN: 978-1-61208-404-6

Means to ensure confidentiality include encryption algorithms and protocols which are readily available on the Android platform. Android Smartphones are used daily in ecommerce apps and applications which necessitates support for all common security standards. The underlying Java system provides a rich provider architecture [12] to enable key management, key exchange, symmetric and asymmetric encryption, block and stream ciphers. Android customizes the implementation of the Java architecture via the "Bouncy Castle" [13] implementation. And of course, computing power is amply available on today's multicore smartphone systems.

We also draw on the standard Public Key Infrastructure [PKI] standard. Public and private key pairs are generated for each peer. Public keys are shared, i.e. made available to all peers that participate in the content sharing network. And, of course, private keys are maintained in secret, which enable peers to decrypt and authenticate communications.

IV. P2P CONTENT SHARING MODEL

Peer-to-peer (P2P) is a communications model in which peers communicate on an equal basis with each other. There is no central sever, no peer is a mere client. All peers have the same capabilities: any peer can initiate a communication session.

While all peers share advanced connection capabilities with high throughput and low latency, in our architecture each peer can have all or some of the following capabilities:

1. The peer has video and audio reproduction device, i.e. a suitably-sized display screen and audio speakers. A peer that has this capability is called a “consumer” peer.
2. The peer has several sensors, such as a video camera, audio microphone, location sensors, such as GPS receiver, and attitude indicator, such as an accelerometer. A peer that has this capability is called a “producer” peer.
3. The peer has computing power to merge streams of multi-media, such as combining video, audio, and location data; but also the ability to coordinate multiple video/audio streams together based on precise location and attitude data. A peer that has this capability is called a “mediator” peer.
4. The peer has administrative authority. It can gather and keep information about the available peers, their capabilities and their trust worthiness. A peer that has this capability is called a “tracker” peer.

Each peer also carries a unique identity, which is made known to other peers.

Once a peer is identified, it is a matter of trust whether and to what degree the peer is allowed to partake in the shared media content. The trust value and the peer’s history of relevant transactions are maintained in a container we call “trust nugget”. This nugget contains detailed information on a peer’s participation, such as length and quality of stream transmission, ratio of seed vs. leech behavior, judgments of other stream participants, etc. The nugget content is signed with a special master private key. It can be verified only via the special master public key. This ensures that the trust information maintains its integrity, even as it is shared with peers in the swarm that have lower trust values.

Trust information per peer is maintained by “tracker” peers. The sole requirement for starting a new swarm is the existence of an initial tracker peer that we call the “boot strap peer”. This peer initially creates the master public/private key pair that is only shared with other trusted tracker peers. A trusted peer maintains a database of trust nuggets for all peers in the swarm. Again, initially, only one peer, i.e. the boot strap peer, has such a database, but as other peers attain higher trusted peer status, they can become tracker peers and receive the database. All tracker peers also participate in synchronizing the database to reflect the trust state of the complete P2P network and all its peers. The trust value for a peer is computed from the peer’s history of transactions. The computation is done by a tracker peer whenever a peer reports on another peer. A common scenario is that a peer serves as a producer of media content: it makes the content available to the peers in the swarm. Once a peer has “consumed” the content, the “producer” peer notifies a tracker peer of the peer’s behavior: good or bad. The tracker peer enters a new transaction into the peer’s nugget and signs it with the master private key. Tracker peers are the backbone of our trust model. New peers need to register with one trusted peer which creates a trust nugget for the new peer. The new peer also creates a public/private key pair and submits its public key to the tracker peer.

When a peer acts as a producer peer, i.e. it makes new content available to the swarm; it can set a trust threshold, i.e. a minimum trust value, required for any peer to access the content. Only peers whose trust value meets the threshold can participate. The producer peer also determines the weight of a peer’s participation when computing a peer’s new trust value.

Mediator peers transform streams of media from producer peers into new streams. In effect, a mediator peer combines “consumer” and “producer” behavior. Like any peer, it has to register with a tracker and establish a trust nugget. To “consume” a stream from a producer it must pass the trust threshold, and in turn it will set a trust threshold for other peers to consume its output stream.

Consider the following scenario to illustrate how our model enables shared augmented reality: a user holds a smartphone with forward facing camera, video display, and geolocation sensors. Here the user’s smart phone serves as a producer peer serving a video stream, a location data stream and an attitude data stream. Somewhere else in the cloud is the producer peer serving the virtual reality model of the user’s surroundings. Somewhere else in the cloud is a mediator peer that coordinates and combines the real-time video from the user’s camera with a virtual reality model of the user’s surroundings based on the location and attitude data stream from the user. And finally, the user’s smartphone also is a consumer peer in that its display shows a video stream produced by the mediator peer in the cloud,

 ISBN: 978-1-61208-404-6

We envision content sharing networks with multiple consumer peers, mediator peers, producer peers, all coordinated by tracker peers. A peer can impersonate one tracker personality, e.g. just be a plain consumer peer, but also serve as the all might peer in combining all four peer personality. And of course, a peer can add and shed personalities as the situation and context changes.

V. IMPLEMENTATION FRAMEWORK

In reflection of our content sharing network architecture, our implementation framework provides feature rich components that can be assembled into a peer. At first, a peer has the basic capabilities to establish its identity, its peer properties and to connect to other peers. Then each peer can assume additional capabilities via any of these components:

1. The "producer" component, which gathers data from sensors (i.e. camera, microphone, GPS receiver, accelerometer, etc.) and make them available in stream format.
2. The "mediator" component, which receives data on multiple incoming streams, to produce a combined outgoing stream, which contains the logical coordination of the incoming data. The coordination is based on location and attitude data that is associated with input streams. The coordination can be done in several modes: add, merge, and layered. The add mode simply combines that input streams: this is useful for time synchronized multimedia that does not cause direct interference, e.g. combining audio and video. The merge mode attempts to combine similar-type input streams into an out stream. Time and location data is used, plus an attempt is made to recognize key features that are present in all input streams to correct the location and attitude data. The layer mode preserves the input streams and allows consumers of the output stream to select layers dynamically.
3. The "consumer" component, which receives an incoming stream and renders it onto suitable output devices (i.e. display, speaker, etc.). If the input stream is of layered mode, it also allows selecting one or more layers.
4. The "tracker" component, which accepts registrations from other peers, maintains their trust information and coordinates the available peers and streams available in the content sharing network.

All these components are available in Java, so they can be assembled into a peer that runs on a mobile device, i.e. Android smartphone, or a peer that resides in the cloud.

For our prototype implementation we assembled a set of peer types built from these components:

(1) A set of tracker peers, initially just one: the boot strap peer application; this application runs as a Java application in the cloud and also serves as the control and observation point for our prototype implementation.

(2) A producer peer Java application to submit information about a content stream; multiple instances, i.e. multiple source peers can be introduced into the content sharing network.

(3) A mediator peer Java application that coordinates multiple data streams, and makes it available to other peers.

(4) A consumer peer to run on an Android mobile device. Android is implemented in Java on a Linux base and therefore offers a flexible and standard set of communication and security features.

Figure 1 shows a sample scenario with one producer, one tracker, one mediator, and one consumer peer:

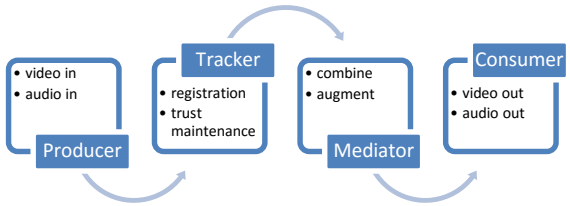

Figure 1: Peer-to-Peer Network

A. *Tracker Peer (also boot strap peer)*

The central component of our architecture is the tracker peer. It maintains a database of all peers and a tracks the collection of data streams that are made available by sources. Our tracker peer prototype presents a display of all peers and streams (see Figure 2).

When a new peer connects to a tracker peer, authentication is achieved via the peer's openID, which is validated the openID provider. If the peer is new, i.e. the tracker peer has no trust nugget for the peer, the new peer must provide its public key and a new trust nugget is created. The peer's public key is later provided to consumer peers who will use it to encrypt content destined for that peer. The top part of the tracker window shown in Figure 1 lists the peers that are currently part of the content sharing network. Each peer is shown with its avatar, its identification and location detail, the level of trust it has achieved so far, and the number streams that the peer is currently participating in. The center part of the tracker window shows a log of peer and stream access activity among the peer that are part of the content sharing network. The lower part of the tracker window shows the list of available media streams. The active stream detail column shows the title and the actual URL used to connect to the stream. The "Trust" column displays the minimum trust threshold that a peer must pass to be allowed to participate in the stream. The "bonus" column list the increment a peer is giving for a successful, i.e. benevolent, participation in the stream production, delivery, and consumption.

B. *Producer Peer*

The producer peer application is used to submit information about a content stream; multiple instances, i.e. multiple producer peers can be introduced into the content sharing network. In our prototype implementation, we provide a very simple version: a simple dialog that captures a input sensor as media stream and allows to submit the stream information to a trusted peer. Figure 3 shows a simple Java application as producer peer.

 ISBN: 978-1-61208-404-6

Figure 2: Boot Strap Peer

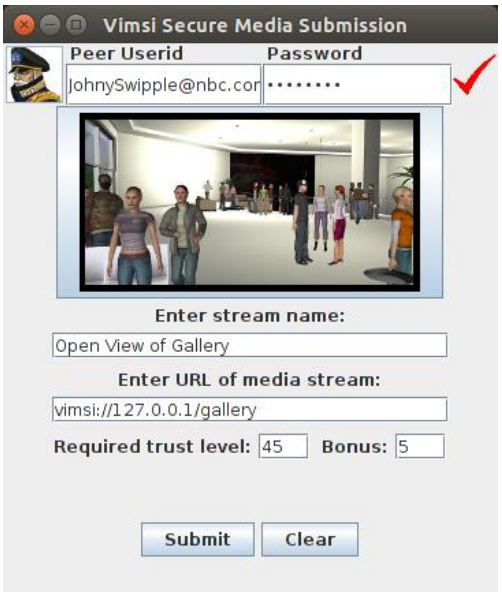

Figure 3: Media Producer Peer

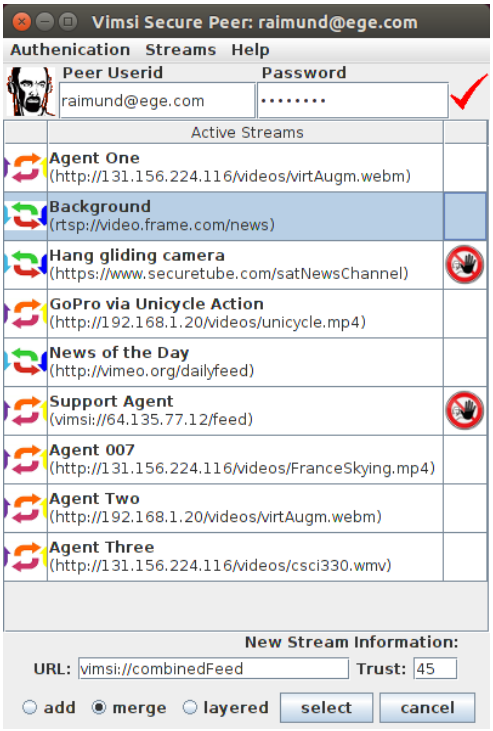

Figure 4: Mediator Peer

C. *Mediator Peer*

The purpose of a mediator peer is to select input streams and coordinate them into an output stream. Figure 4 shows a screen capture of the Java Mediator Peer prototype. Once the peer is authenticated with a tracker peer, it requests a list of available streams. Figure 4 shows all streams that are currently available. Note that some streams are not currently available: they display a “do not touch” symbol to indicate that they require a greater trust value for access. The reason why all stream are displayed, even the ones which require a higher trust value than what the peer currently has, is to give the peer an incentive to first participate in another stream to add the bonus to its trust value. However, only streams can actually be selected for which the peer is currently qualified.

The mediator peer further has to set which mode should be used for the coordination: add, merge or layered; “merge” is selected here to indicate that the selected input streams are

time and location coordinated and merged into a combined output stream. Finally the dialog allows entering a unique name for the combined stream. The new stream has the tag "vimsi" which indicates to other peers that it is a combined stream coordinated by a relay peer.

D. Consumer Peer

The final component of our prototype framework is our proof-of-concept consumer peer implementation for the Android platform. Figure 5 shows three screens: "login", "stream selection", and "stream play" of our Android prototype consumer peer application.

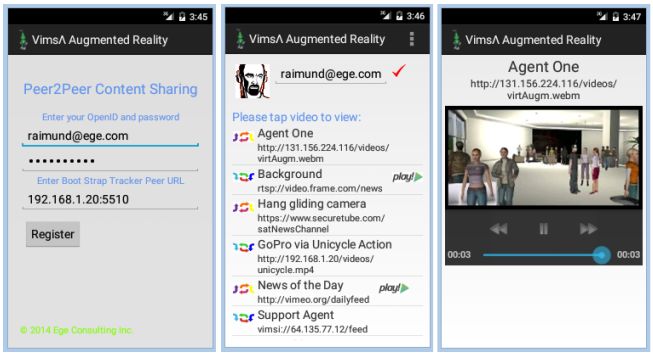

Figure 5: Android Prototype Peer App

The "login" screen allows the peer to authenticate with its OpenID credentials. The user enters userid and password, plus the URL of a boot strap tracker peer. If the peer is recognized into the content delivery network, the tracker peer transmits all available streams to the new peer. The "stream selection" screen shows these streams. As before, not all streams are available to the new peer: only those that display the "play" button can be used by this peer based on its trust level. Once the "play selected video stream" button is pressed, and a sufficient read-ahead buffer has been accumulated, the video stream starts playing on the Android device. The third screen capture shows the video stream being displayed. The video shown here is derived from a scene generated by a virtual reality rendering producer peer. The on-screen control allow the user to control the video display.

VI. CONCLUSION

Our goal was to enable the merging of realities - both real and virtual - into a comprehensive experience to enable life like immersion in real time. In this paper we described a framework for a peer-to-peer based content sharing network where peers collect, augment and share multi-media streams of data.

We introduced a model to gather, manage and use trust information to allow an ad hoc assembly of peers, and demonstrated the feasibility of our approach with a Java-based prototype implementation that includes a peer client for the Android platform. We also showed that the security capabilities of the Android/Java/Linux system are up to par and implementable on today's crop of smartphones.

While our current implementation already allowed the merging of multiple streams, much additional work needs to be done to allow the combination and mediation of real-time multimedia stream. Our next step will be to focus on using virtual reality models as concrete reference and marking points to enable realistic augmented reality worlds.

REFERENCES

[1] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier and B. MacIntyre. Recent Advances in Augmented Reality. IEEE Computer Graphics and Applications (CGA) 21(6):34-47, 2001.

[2] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond and D. Schmalstieg. Real-Time Detection and Tracking for Augmented Reality on Mobile Phones, IEEE Trans. Vis. Comput. Graph., 16(3):355-368, 2010.

[3] A. Morrison, A. Mulloni, S. Lemmelä, A. Oulasvirta, G. Jacucci, P. Peltonen, D. Schmalstieg and H. Regenbrecht. Collaborative use of mobile augmented reality with paper maps, Journal on Computers & Graphics (Elsevier), 35(4):789-799, 2011.

[4] OpenID, http://www.openid.net. [accessed September 19, 2014]

[5] J Y. Atif. Building trust in E-commerce. IEEE Internet Computing, 6(1):18–24, 2002.

[6] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. Communications of the ACM, 43(12):45–48, 2000.

[7] H. Li and M. Singhal. Trust Management in Distributed Systems. Computer, vol. 40, no. 2, pp. 45-53, Feb. 2007.

[8] OMA Digital Rights Management V2.0, http://www.openmobilealliance.org/technical/release_program/drm_v2_0.aspx. [accessed September 20, 2014]

[9] [10] C. Adams and S. Lloyd. Understanding PKI: concepts, standards, and deployment considerations. Addison-Wesley Professional. ISBN 978-0-672-32391-1. 2003.

[10] Raimund K. Ege. OghmaSip: Peer-to-Peer Multimedia for Mobile Devices. The First International Conference on Mobile Services, Resources, and Users (MOBILITY 2011), pages 1-6, Barcelona, Spain, October 2011.

[11] Raimund K. Ege. Secure Trust Management for the Android Platform. International Conference on Systems (ICONS 2013), Seville, Spain, January 2013.

[12] Java Cryptography Architecture (JCA) Reference Guide. http://docs.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html. [accessed September 20, 2014]

[13] The Legion of the Bouncy Castle. http://www.bouncycastle.org/java.html. [accessed September 20, 2014]

 ISBN: 978-1-61208-404-6