CloudMediate

Peer-to-peer Media Aggregation for Augmented Reality

Raimund K. Ege
Department of Computer Science
Northern Illinois University
DeKalb, USA
email: ege@niu.edu

***Abstract*—Augmented Reality strives to produce a world composed from virtual models and multiple multi-media streams, and enables complete immersion into the result via modern mobile hand-held or wearable devices. In this paper, we present a conceptual media aggregation framework that is flexible, powerful and scalable to identify, establish and manage connections to media stream source (virtual and real) and produces an adaptable world view, which is then made available to consuming devices with attitude feedback. The framework is structured into three layers: presence, integration, and homogenization layers that work together in a peer-to-peer (p2p) manner to facilitate the delivery of multimedia data. Each layer features cloud-based mediator components, each mediator transforms an input stream into an output stream. This mediation process is context-aware, adaptive and dynamically structured.**

Keywords-augmented reality; virtual reality; peer-to-peer systems; multi-media content delivery.

I. INTRODUCTION

The proliferation of mobile and wearable devices has enabled access, at least on a physical level, to a multitude of disparate but often related information, while scaling geographical barriers. This information, in the form of multimedia streams is produced, stored on and accessed from various kinds of heterogeneous devices. Our goal is to select suitable input streams, correlate and combine them into a virtual world. The virtual world can then be made available to clients, again rendered onto suitable mobile and wearable devices.

Fig. 1 shows our overall idea of how CloudMediate operates. The left side of the figure symbolizes the multitude of potential input streams. While audio and video streams are most common, our approach allows arbitrary streams of data from any sensor. The right side of the figure shows consumption of streams by mobile devices. The common smartphone might be one example of such a display device. Wearable devices, such as headsets and virtual glasses are the target of our approach. Both sides are connected by a cloud-hosted network of intermediaries that normalize, correlate and combine input streams into consumable output streams.

We use the term "mediator" to describe these intermediaries. We chose this term in analogy to the "Mediator" behavioral pattern that address the responsibilities of objects in an application and how they communicate [1].

In this paper, we describe a framework for peer-to-peer media aggregation for augmented reality that features a three-layer architecture for multimedia mediation. The paper is organized as follows: Section 2 presents some related work and briefly covers some differences and similarities between our architecture and existing ones. Section 3 describes our overall architecture and each layer and what functions are performed therein. We also discuss the different classes of mediators in those layers. Section 4 covers the CloudMediate web-based control interface that allows peers to register and sign up with their multi-media stream. The section also introduces our mobile application for portable and wearable devices to allow a peer user to become immersed in the resulting augmented virtual reality. We conclude with an outlook to our future work.

II. BACKGROUND

Virtual Reality devices are emerging rapidly in the market place. Every major vendor of systems and hardware has introduced mobile and wearable gadgets to support virtual and augmented reality. From simple holders for smart phones, to headsets from market leaders such as Samsung and Google (even eye glasses, e.g., Google GLASS). Software to enable these devices is becoming more commonplace. The

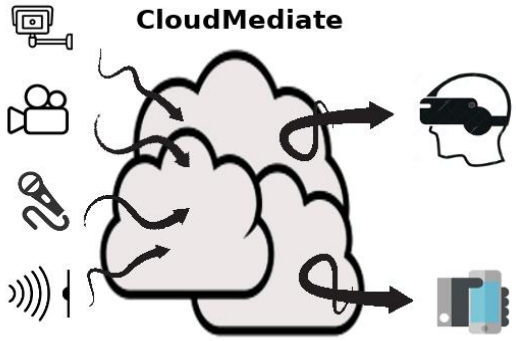

Figure 1. CloudMediate Structure.

 ISBN: 978-1-61208-529-6

application developer kits are becoming ever more powerful to harness the dynamic features of these devices.

Multimedia data requires special attention to throughput, timeliness and other quality of service factors. There is a need for architectures to deal with buffering and the intermittent connection associated with mobility. Our approach to enabling high quality access is to build a layered framework of mediators [2]. Lower-layer mediators connect to the actual data sources, while higher-layer mediators provide a logical schema of information to applications. Mediators are typically employed in a situation where the client data model does not coincide with the data model of the potential data sources. They are facilitators that search for likely resources and ways to access them. They provide a mapping of complex models to enable interoperability between client and source(s). Many mediator systems have been proposed for a variety of applications, a major problem often encountered is how to seamlessly query and integrate data from heterogeneous data sources [3].

In peer-sourced augmented reality systems, the management of the multi-media source and establishment of trust is essential [4]. In our prior work [5] [6], we investigated the authentication of participants in peer-to-peer networks, the establishment and management of trust, and the use of such media sources in building content management systems. An important lesson was that while modern mobile devices are compute-capable, cloud-based components add additional heft and authority to a seamless and smooth creation of a truly immersing virtual and augmented reality experience [7][8][9].

III. MEDIATOR ARCHITECTURE

We approach the task of delivering multi-media streams from their producers to the consumers be decomposing it into small steps according to an overall architectural framework. The architectural framework features three layers: Presence, Integration and Homogenization as shown in Fig. 2.

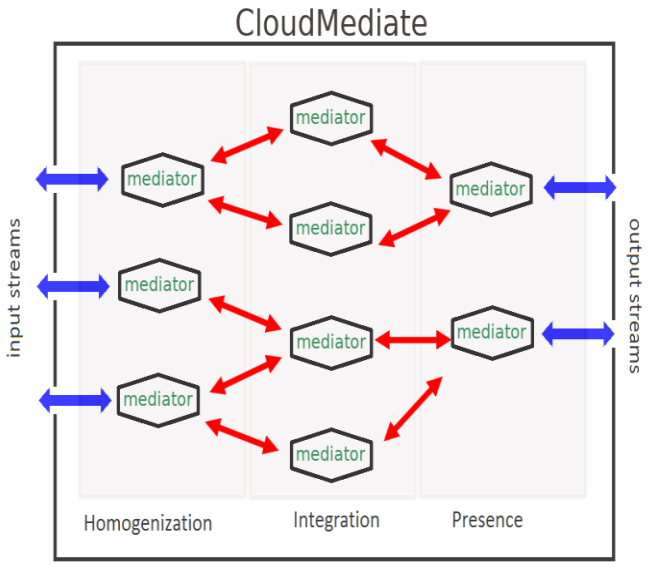

Figure 2. Layers of Architecture.

The Presence layer is closest to the device that consumes the output stream. The Homogenization layer is closest to the device that produces the input stream. The integration layer correlates Presence and Homogenization mediators.

Each layer is composed of several cloud-based mediators. Different classes of mediator are employed within each layer. Fig. 3 shows the top of the mediator class hierarchy.

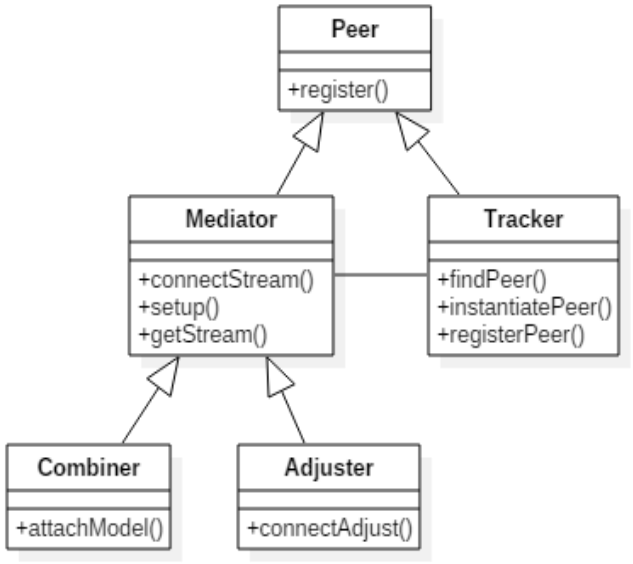

Figure 3. Mediator Class Hierarchy.

Each mediator has an input and an output side and transfers and negotiates on three kinds of information; the schema of the data stream, the data stream itself, and some quality of service (QoS) information specific to the stream. The nature of mediation varies from a simple combination of input streams, to correlation of virtual and augmented streams, and up to reformatting of a stream based on attitude information.

The central class is Mediator: it manages the input/output connections. It is a subclass of Peer, which handles peer setup and registration. A special Tracker peer is responsible for keeping track of all peers. It accepts registrations from other peers and facilitates peer lookup. It can also instantiate new peers to create a path from incoming media streams to output streams through the layers of our framework.

Two subclasses of Mediator are shown here: Combiner is able to attach a virtual reality model to input streams to correlate multiple input streams into a single output stream; Adjuster connects to an attitude stream of an output stream device. As the device attitude changes, it adjusts the perceived attitude geometry of the output stream.

A. *Homogenization Layer*

In our architecture, each homogenization layer mediator is directly associated with an incoming multi-media data stream from a mobile device.

In addition to the actual data sensed, it must be packaged with the exact time of recording. Multiple streams of sensor data are combined into a multi-media stream which

 ISBN: 978-1-61208-529-6

interleaves its content streams plus provides meta-data to ensure their proper sequencing and correlation. It is important that the container format used to wrap the content streams is flexible enough to accommodate not only the stream data but also extensive amounts of reference information used to combine the streams. We are using an extension of the WebM project [10] format. The WebM container format is an open standard and allows us to collate an unlimited number of video, audio, pictures and subtitle tracks into one stream. We add the capability of identifying reference elements at identified points in time and at locations.

Video data is the key stream type captured via video sensors, i.e., cameras, available on the wearable devices carried by a peer. Video is captured as a sequence of video frames. Each frame carries a time stamp as major meta reference data. Equally important is the location of video capture, lens parameters and attitude, i.e., which way the camera points. Our container format allows us to group sequential video frames into video sequences that share a common location. We represent the location with a "Normal Vector".

Fig. 4 shows how a normal vector captures not only the location of a video plane but also its relative position. The normal vector is represented via 2 points: its origin and extent points. Both points are captured in absolute latitude and longitude coordinates. While the distance between origin and extent point of the normal vector is not normally relevant, we use the length of the vector as a guide to the size of the video frames being referenced. A longer vector indicates a larger area shown in the video. We use the length of the normal vector when attaching multiple streams into a virtual reality frame.

Audio data is captured by microphones and sequences into frames that are referenced with a time stamp. While it would be possible to also capture and store directional information, which might be meaningful in the case of a directional microphone, we are able to deduce that information from the normal vector stored for video frames recorded at the same time on the same device. Of course, if the wearable device only records video, then such directional information is not available. We are considering this extension for future work. The audio data with its correlated reference metadata is also wrapped into the same container as the video data.

Any other data, such as gathered from special purpose data sensors, is equally framed and referenced. Examples of such data might be the heart rate of the person wearing the device, the temperature of the surroundings, or movement & acceleration data measured. Our container format allows a free-form type designator that enables sensors of any kind, as long as their sensed data can be digitized and framed.

B. Integration Layer

The mediators that comprise this layer feature multiple incoming streams and usually just one outgoing stream. The task of the mediator is to correlate and combine the input streams into a coherent output stream.

Figure 4. Normal Vector.

The meta information contained in the inputs' container guides the combination. The simplest correlation is for 2 video input streams that is based on the time and location of each stream. The attitude of each individual stream is carried in its normal vector. The output stream contains the adjusted meta information for the combined video.

Our container format also allows the carrying of virtual reality model data. Actually, such data is similar in nature to "real" data, but is derived not from sensors but from virtual reality models of the surroundings that the wearers of the wearable devices inhabit. To allow clear distinction of sub-streams within the container, each sub-stream carries a unique stream identifier, which is correlated to a stream dictionary that holds relevant information about the sub-stream. The complete stream dictionary is embedded into the multi-media stream at regular intervals.

C. Presence Layer

The mediators in this layer are created by the Tracker peer (see Fig. 3) based on a client's request. The primary functions performed in this layer are:

1. Identify and process an input media stream.
2. Connect to location and attitude streams from a peer client.
3. Continuously adjust and re-compute an output media stream to the peer client.

Based on a peer client request, a suitable mediator is instantiated in the cloud to handle that request. It is connected to input and control streams. The mediation of input stream involves geometric conversions to adjust the spatial location and dimensions to produce a life-like presentation. The rendering of the resulting augmented reality is adjusted to conform to the attitude of output device: this enables a peer client to view the resulting stream in the same geo space as its recording device.

 ISBN: 978-1-61208-529-6

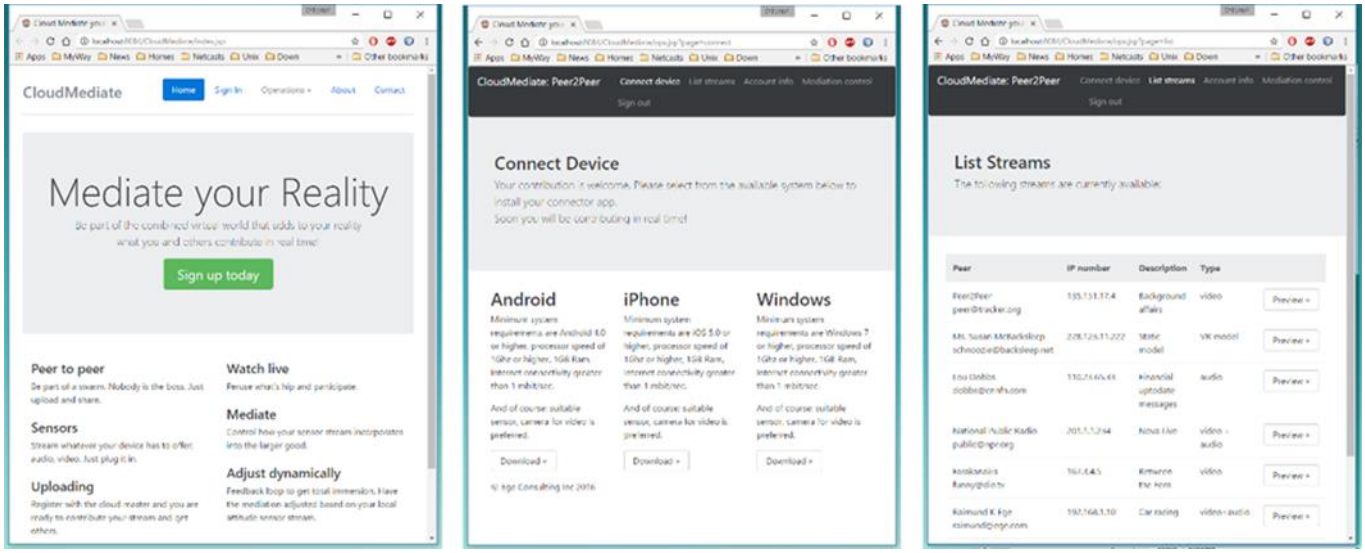

Figure 5. CloudMediate Web-Control Interface.

QoS management is essential to efficiently access pertinent information at the required level of quality. This function attempts to meet the level of quality required by user. The continuous nature of the QoS management is especially important in the event that the client device is mobile. Resources are scarce on mobile devices and the availability of a resource may vary significantly and unpredictably during the runtime of an application. In the absence of resource guarantees, applications need to adapt themselves to the prevailing operating conditions.

IV. PROTOTYPE: CLOUDMEDIATE

The features provided by our architecture framework are illustrated by our reference prototype implementation. It is anchored by the CloudMediate web-based control interface that allows peers to register and sign up with their multi-media stream. Fig. 5 shows screenshots. The left part shows the central welcome and sign-up screen.

Users can sign up and sign in, which then makes the peer-to-peer features available. The following operations become available: “Connect Device”, “List Streams”, “Account Info” and “Mediation Control”. Only control functions are available via the web interface. Actual stream production, connection and consumption and is handled via custom apps that can be downloaded. “Connect Device” (center screen shot) allows users to download a suitable app for their device. While the screenshot shows Android, iPhone and Windows, we currently have only an Android prototype under development. The “List Streams” operation displays all streams that are currently controlled by the CloudMediate system, that is all the input streams offered up by all peer clients, plus the output streams of all active mediators from all three layers of our framework. The “Account Info” operation allows the peer to see its own capabilities. The “Mediation Control” operation allows a peer to adjust parameters and settings of its own streams.

The final component of our prototype framework is our proof-of-concept client peer implementation for the Android platform. Fig. 6 shows the login screen of our Android prototype client peer application. 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.

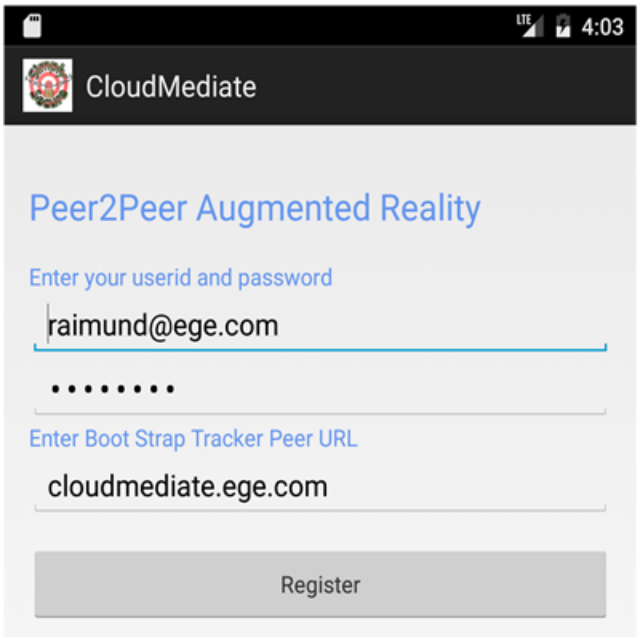

Figure 6. Android App Login Screen.

While the main app activity allows a peer to consume an available stream (see Fig. 8), our prototype also allows a peer to serve up its streams. Fig. 7 shows how a peer connects its input sensors in the mediator network. Any streams that are

gathered by the peer device can be made available. In the example shown, video, audio and location streams are available and can be selected.

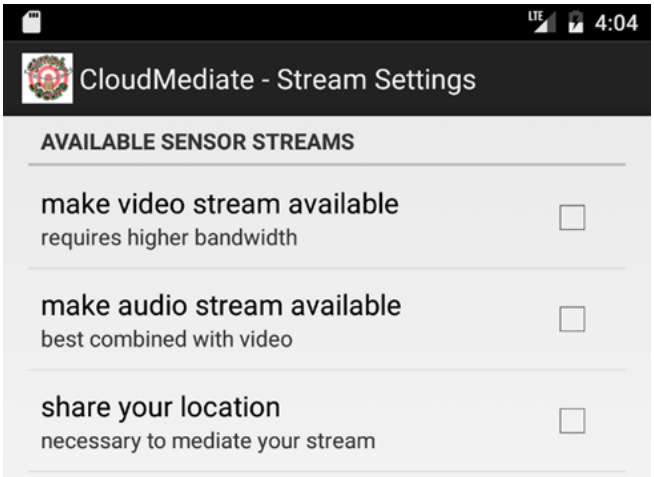

Figure 7. Android App Stream Setting.

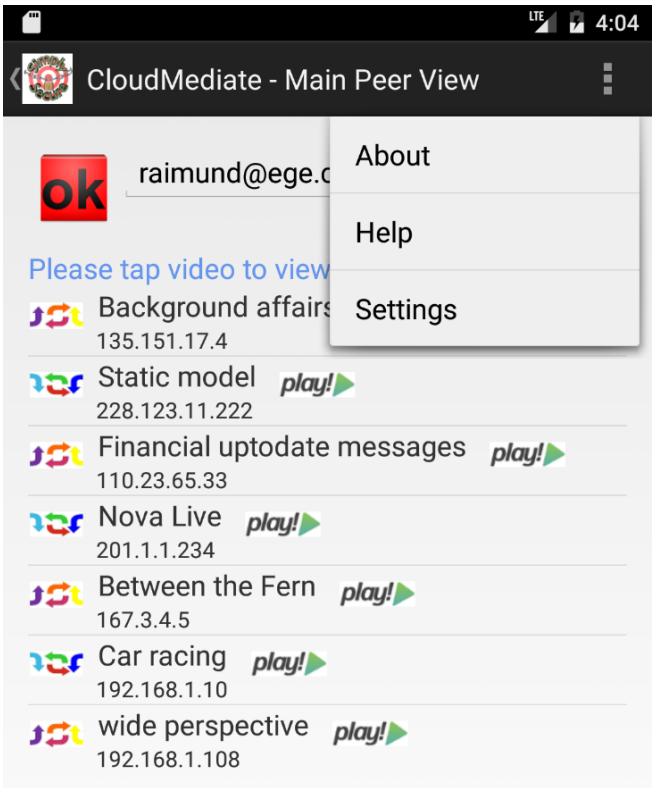

Figure 9. Android App Stream Selection.

The "stream selection" (Fig. 8) screen shows the currently available streams in the prototype application. 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.

Figure 8. Android App Stream View.

The screen capture in Fig. 9 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 allows the user to control the video display.

V. CONCLUSION AND FUTURE WORK

The interchange of data between client and heterogeneous sources requires an efficient and dynamic approach to mediation. The framework described in this paper features three layers of mediators: presence, integration, and homogenization. We gather multi-media sources from sensors and tag them suitable for adequate correlation and mediation. Peers join a content delivery network and establish trust relationships among each other. Cloud-based media streams – mediated based on their associated metadata - are embedded into a reference virtual reality model and rendered into a geo-referenced attitudinal output media stream suitable for a heads-up display.

The advantage of our mediation process is its adaptive and dynamic nature. The framework is designed to uniquely determine how to fulfill each query while taking properties of delivery into consideration. Our mediation architecture is a work-in-progress and there are many research issues that will

 ISBN: 978-1-61208-529-6

be encountered during the course of this project, they include but are not limited to, defining of communication protocols with specific focus on how to deal with real-time data and mobility (e.g., temporary loss of connectivity in mobile devices), security issues involved with the distribution and access of data across a p2p network, and how to intelligently decompose and integrate XML schemas and streams while avoiding loss of information.

Our goal for future research is to enhance the media mediation capabilities of our cloud-based peer components. We envision a rich augmented reality world that is populated by many dynamic input streams.

REFERENCES

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.

[2] R. Ege, L. Yang, Q. Kharma, and X Ni, "Three-Layered Mediator Architecture Based on DHT" Proceedings of the International Symposium on Parallel Architecture, Algorithm and Networks (I-SPAN), Hong Kong, China, pp. 313-318, 2004.

[3] S. Giesecke, J. Bornhold, and W. Hasselbring, "Middleware-Induced Architectural Style Modelling for Architecture Exploration", Proceedings of 2007 Working IEEE/IFIP Conference on Software Architecture (WICSA'07), pp. 44-47, 2007.

[4] S. Aukstakalnis, Practical Augmented Reality, Addison-Wesley Professional, 2017.

[5] R. Ege, "Secure Trust Management for the Android Platform," International Conference on Systems (ICONS 2013), Seville, Spain, pp. 98-103, 2013

[6] R. Ege, "Peer to Peer Media Management for Augmented Reality," International Conference on Networking and Services (ICNS 2015), Rome, Italy, pp. 95-100, 2015.

[7] R. Azuma et al., "Recent Advances in Augmented Reality," IEEE Computer Graphics and Applications (CGA) 21(6), pp. 34-47, 2001.

[8] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Real-Time Detection and Tracking for Augmented Reality on Mobile Phones," IEEE Transactions on Visualization and Computer Graphics, 16(3), pp. 355-368, 2010.

[9] A. Morrison et al., "Collaborative use of mobile augmented reality with paper maps," Journal on Computers & Graphics (Elsevier), 35(4), pp. 789-799, 2011.

[10] The WebM Project - About WebM. http://www.webmproject.org. [retrieved December 19, 2016]

 ISBN: 978-1-61208-529-6