OghmaSip: Peer-to-Peer Multimedia for Mobile Devices

Raimund K. Ege
Department of Computer Science
Northern Illinois University
DeKalb, IL 60115, USA
ege@niu.edu

Abstract—**Mobile devices are rapidly being accepted as primary vehicle to consume multimedia content. Capable smart phones with high-speed next-generation Internet connectivity are becoming common place. Peer-to-peer content delivery is one way to ensure that sufficient data volume can be efficiently delivered. However, the openness of delivery demands adaptive and robust management of intellectual property rights. In this paper we describe a framework and its implementation to address the central issues in content delivery: a scalable peer-to-peer-based content delivery model, paired with a secure access control model that enables data providers to reap a return from making their original content available. We describe our prototype implementation for the Android platform that uses the session initiation protocol (SIP) for peer communication.**

Keywords-multimedia sharing; peer-to-peer content delivery; session initiation protocol

I. INTRODUCTION

High bandwidth Internet connectivity is no longer limited to reaching PCs and laptops: a new generation of devices, such as netbooks and smart phones, is within reach of 3G/4G telecommunication networks. Smart phones have ushered in a new era in omni-present broadband media consumption. Services such as iTunes, YouTube, and FaceBook are popularizing delivery of audio and video content to anybody with a broadband Internet connection.

In this paper, we describe a framework for multimedia content delivery that is based on peer-to-peer file sharing. Peers communicate with messages according to the session initiation protocol to discover each other and exchange data. We describe the implementation of a video player application for the Android platform that delivers video in a secure and managed way.

Delivering multimedia services has many challenges; the ever increasing size of the data requires elaborate delivery networks to handle peek network traffic. Another challenge is to secure and protect the property rights of the media owners. A common approach to large-scale distribution is a peer-to-peer model, where clients that download data immediately become intermediates in a delivery chain to further clients. The dynamism of peer-to-peer communities means that principals who offer services will meet requests from unrelated or unknown peers. Peers need to collaborate and obtain services within an environment that is unfamiliar or even hostile.

Therefore, peers have to manage the risks involved in the collaboration when prior experience and knowledge about each other are incomplete. One way to address this uncertainty is to develop and establish trust among peers. Trust can be built by either a trusted third party [2], or by community-based feedback from past experiences [3] in a self-regulating system. Other approaches reported in the literature use different access control models [4] [5] that qualify and determine authorization based on permissions defined for peers. In such a complex and collaborative world, a peer can benefit and protect itself only if it can respond to new peers and enforce access control by assigning proper privileges to new peers.

The broader goal of our work is to address the trust in peers which are allowed to participate in the content delivery process, to minimize the risk and to maximize the reward garnered from releasing data in to the network. In our prior work [9] [15], we focused on modeling the nature of risk and reward when releasing content to the Internet. We integrated trust evaluation for usage control with an analysis of risk and reward. Underlying our framework is a formal computational model of trust and access control. In the work reported here, we focus on the implementation aspects of the framework, especially the use of the Session Initiation Protocol (SIP).

Our paper is organized as follows: the next section will elaborate on how the data provider and its peers can quantify gain from participating in the content delivery. It also explains our risk/reward model that enables a data source to initially decide on whether to share the content and keep some leverage after its release. Section III describes our prototype architecture that uses the session initiation protocol to establish a community of peers to share content. No central tracker manages a database of peer and trust information, but rather peers maintain a distributed database. Peers can serve both as source and as consumer of data. Section IV introduces our prototype client for the Android platform and its implementation in Java. Data is exchanged using the Stream Control Transmission Protocol (SCTP) and is secured using a PKI-style exchange of public keys and data encryption. The paper concludes with our assessment of how peer-to-peer systems can shed their freewheeling image via sensible access control additions.

II. Qualifying the Value of Multimedia

It is amazing at what rate multimedia data is introduced to the Internet and consumed. Almost any kind of multimedia data has value to somebody. Releasing it to the Internet carries potential for reaping some of the value, but also carries the risk that the data will be consumed without rewarding the original source. In addition to the cost of creating the original multimedia data, there is also a cost associated with releasing the data, i.e., storage and transmission cost.

For example, consider the life of a typical "viral" video found on a popular social media site: the video is captured via a smartphone camera (maybe even accidentally), then is uploaded to the social media site, discussed (i.e., "liked" and "friended"), and viewed by a large audience (measured in millions of hits). The video taker is rewarded with fame, rarely gets a monetary reward, the entity that is getting rewarded is the social media site, which will accompany the video presentation with paid advertising.

Let us first recap our model (described also in [1]) to asses risk and reward, by quantizing aspects of the information interchange between the original source, the transmitting medium and the final consumer of the data. Our emphasis here is on the reward quantity, rather than on how trust in peers affects the outcome.

In a traditional fee for service model the reward *"R"* to the source is the fee *"F"* paid by the consumer minus the cost *"D"* of delivery:

$$R = F - D$$

The cost of delivery *"D"* consist of the storage cost at the server, and the cost of feeding it into the Internet. In the case of a social media site, considerable cost is incurred for providing the necessary server network and their bandwidth to the Internet. The social media site recovers that cost by adding paid advertising on the source web page as well as adding paid advertising onto the video stream. The site's business model recognizes that these paid advertisings represent significant added value. As soon as we recognize that the value gained is not an insignificant amount, the focus of the formula shifts from providing value to the original data source to the reward that can be gained by the transmitter. If we quantify the advertising reward as *"A"* the formula now becomes:

$$R = F - (D - A)$$

Even in this simplest form, we recognize that *"A"* has the potential to outweigh *"D"* and therefore reduce the need for *"F"*. As the social media site recognizes, the reward lies in *"A"*, i.e., paid ads that accompany the video.

Mediation frameworks can capture the mutative nature of data delivery on the Internet (see also our prior work [8]). As data travels from a source to a client on a lengthy path, each node in the path may act as mediator. A mediator transforms data from an input perspective to an output perspective. In the simplest scenario, the data that is fed into the delivery network by the source and is received by the ultimate client unchanged: i.e., each mediator just passes its input data along as output data. However, that is not the necessary scenario anymore: the great variety of client devices already necessitate that the data is transformed to enhance the client's viewing experience. We apply this mediation approach to each peer on the path from source to client. Each peer may serve as a mediator that transforms the content stream in some fashion. Our implementation employs the stream control transmission protocol (SCTP) which allows multi-media to be delivered in multiple concurrent streams. All a peer needs to do is add an additional stream for a video overlay message to the content as it passes through.

The formula for reward can now be extended into the P2P content delivery domain, where a large number of peers serve as the transmission/storage medium. Assuming "*n*" number of peers that participate and potentially add value the formula for the reward per peer is now:

$$R_p = \sum_{i=1}^{n} (F_i - (D_i - A_i)) - F_p$$

D_i and A_i are now the delivery cost and value incurred at each peer that participates in the P2P content delivery. F_i is the fee potentially paid by each peer. F_p is the fee paid to the data source provider. Whether or not the data originator will gain any reward depends on whether the client and the peers are willing to share their gain from the added value. In a scenario where clients and peers are authenticated and the release of the data is predicated by a contractual agreement, the source will reap the complete benefit.

In our model, we quantify the certainty of whether the client and peers will remit their gain to the source with a value of trust. Trust is evaluated based on both actual observations and recommendations from referees. Observations are based on previous interactions with the peer. Recommendations may include signed trust-assertions from other principals, or a list of referees that can be contacted for recommendations. Our model enables an informed decision on whether to accept a new peer based on the potential additional reward gained correlated to the risk/trust encumbered by the new peer.

III. Peer-to-peer Architecture

Our prototype of peer-to-peer multimedia delivery aims to deliver multi-media content from a source to a large number of clients. We assume that the content comes into existence at a source. A simple example of creating such multimedia might be a video clip taken with a camera and a microphone, or more likely video captured via a smartphone camera, and then transferred to the source. Likewise the client consumes the content, e.g., by displaying it on a computing device monitor, which again might be a smartphone screen watching a Internet video. We further assume that there is just one original source, but that there are many clients that want to receive the data. The clients value their viewing experience, and our goal is to reward the source for making the video available.

In a peer-to-peer (P2P) delivery approach, each client participates in the further delivery of the content. Each client makes part or all of the original content available to further clients. The clients become peers in a peer-to-peer delivery model. Such an approach is specifically geared towards being able to scale effortlessly to support millions of clients without prior notice, i.e., be able to handle a “mob-like” behavior of the clients.

The nature of the source data will dictate the exact details of delivery: for example, video data is made available at a preset quality using a variable-rate video encoder. The source data stream is divided into fixed length sequential frames: each frame is identified by its frame number. Clients request frames in sequence, receive the frame and reassemble the video stream which is then displayed using a suitable video decoder and display utility. The video stream is encoded in such a fashion that missing frames don’t prevent a resulting video to be shown, but rather a video of lesser bit-rate encoding, i.e., quality, will result [7]. We explicitly allow the video stream to be quite malleable, i.e., the quality of delivery need not be constant and there is no harm if extra frames find their way into the stream. It is actually a key element of our approach that the stream can be enriched as part of the delivery process.

In our architecture, peers participate in peer groups. A peer is a network-connected computing device. The purpose of a peer group is to facilitate the dissemination of the multimedia data. Multimedia data, e.g., some video clip, comes into existence at a source. The source tells a single peer about its network location and addressability, i.e., IP and port number. The single peer serves as the “bootstrap” peer, it disseminates the knowledge about the video to the peers in its peer group. The source also advertises the single peer as the “seed” peer on the web. Peers can partake in the video stream either via being told by a peer in their peer group, or by retrieving the “seed” peer from its web advertisement and contacting the “seed” peer and joining its peer group. Peers can do 3 things: (1) they continuously request frames from other peers (the original source is viewed as just another peer) and store them; (2) they may display the frames as video to the user of the peer device; (3) and they make the stored frames available to other peers. Peers don’t have to provide all 3 services. A peer that provides only service (1) and (2) is an “edge” peer, i.e., an end user consumer. A peer that provides service (1) and (3) is a “relay” peer. Relay peers are specifically important for peers that have limited access to the public Internet, i.e., peers behind network boundaries, such as a NAT firewall. In addition, peers stay in contact with each other to continuously update the peer group and source data availability.

Peer communication is achieved via session SIP messages. Each message has a message type and carries a payload. The initial message is of type “peer_join” that a new client peer sends to an existing peer in the peer group. The payload of the message contains the peer’s public key, which will later be used to enable encrypted media delivery. The peer answers with a list of peers that currently make up the peer group. The “ping” message is sent periodically by peers to each other to establish whether they can reach each other: again, a peer that receives a “ping” message answers with its current list of peers in the peer group and its public key. Peers that have answered to a message are maintained as “neighbor” peers and will always by queried first. Another important type of message is “query_media”, which inquires about which media is available and maintained by the peer group. The answer to this message is a list of which peers are able to serve which parts of the available media. The answer also provides communication details such as the IP and port number at which a peer will serve up frames of the media. Every peer constantly monitors the rate of response it gets from the other peers and adjusts its connections to the peers from which the highest throughput rate can be achieved.

Figure 1 shows an example snapshot of a content delivery network with one source, one bootstrap peer, 2 relay peers and one edge peer.

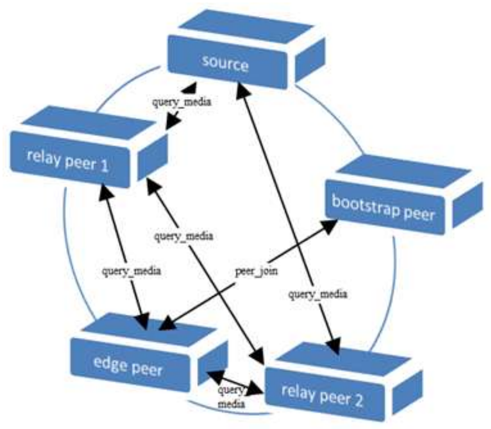


Figure 1. Peer Group with source & active peers

The source is where the video data is produced, encoded and made available. The bootstrap peer knows the network location of the source. Peers connect to the bootstrap peer first and then maintain sessions for the duration of the download: the 2 relay peers and the single edge peer maintain a peer group. The bootstrap peer initially informs the peers in the peer group which source to download from: peer 1 is fed directly from the source; peer 2 joined somewhat later and is now being served from the source and peer 1; the edge peer joined last and is being served from peer 1 and peer 2. In this example, peer 1 and 2 started out as edge peer, but became relay peers once they had enough data to start serving as intermediaries on the delivery path from original source to ultimate consumer.

IV. JAVA IMPLEMENTATION

Our implementation has 3 major components: a typical source application, a typical relay peer, and an edge peer to run on a mobile device. All 3 components are implemented in Java. We chose the Android platform to implement a proof-of-concept client for a mobile device. Android is part of the Open Handset Alliance [10]. Android is implemented in Java and therefore offers a flexible and standard set of communication and security features. The communication among the peers within their peer group uses session initiation protocol (SIP) messaging based on the Sip2Peer library [16]. The actual media exchange uses the Java implementation [13] of the SCTP [14] transport layer protocol. In the following we will first showcase the Android client, and then present details of the relay peer implementation.

Figures 2, 3 and 4 show three sample screen shots taken from the Android system. They illustrate our OghmaSIP media app.

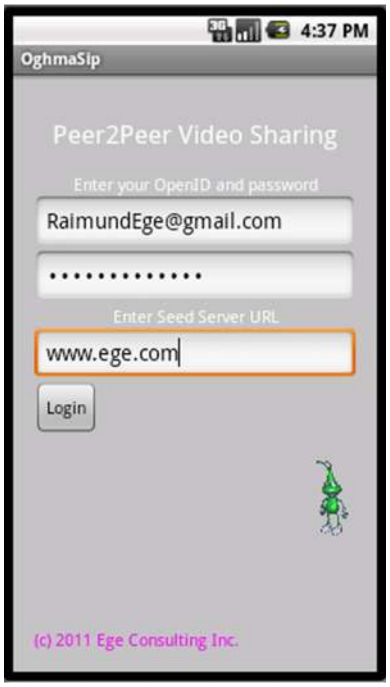


Figure 2. OghmaSip Login Screen

Figure 2 shows the login screen to our OghmaSIP mobile client. It uses OpenID[6] user credentials and allows to establish a connection to a bootstrap peer via a web URL lookup. The client generates a pair of public/private keys and sends “peer_join” message to the bootstrap client.

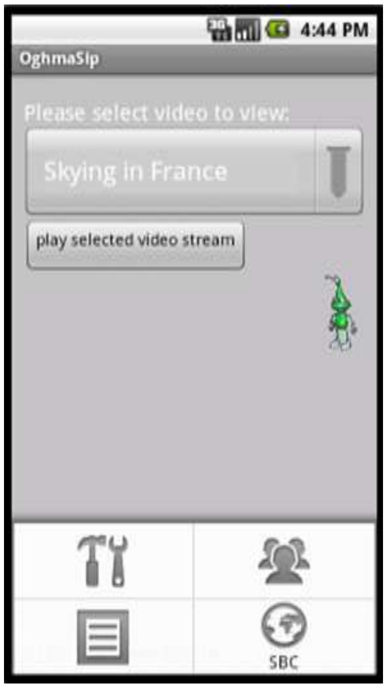


Figure 3. OghmaSip Available Video Streams

Once the bootstrap peer has authenticated the new peer it will respond with a list of available video streams (Figure 3). After the user has made a selection, the screen shown in Figure 4 appears.

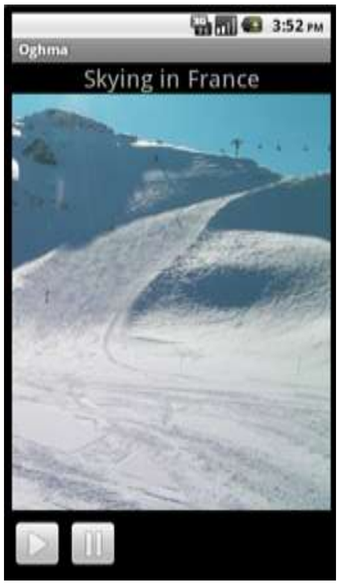


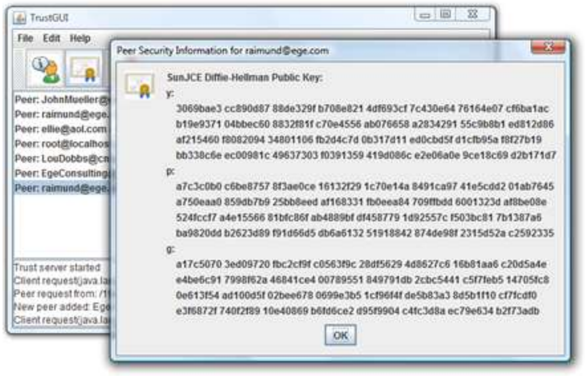
Figure 4. OghmaSIP Video Delivery Screen

Once a sufficient read-ahead buffer has been accumulated, the video stream starts playing on the Android device.

We also provide a Java desktop implementation of a peer. The typical peer is a "Relay" peer, i.e., it will request media frames from the source, potentially show them locally to a user, and then make these frames available to other peers. Peers that wish to participate in the content delivery must first locate media sources. A peer will start by looking up the bootstrap peer via its web advertisement. Like the mobile client, the typical "relay" peer generates a public/private key pair and sends a "peer_join" message to the bootstrap peer. Figure 5 shows the relay peer's graphical user interface that tracks the peers in the peer group: the center of the screen shows peers that have been accepted into the P2P content delivery network; the bottom of the screen shows a log of access requests from other peers. Overlaid is a popup-screen showing the public key information of a selected peer.

At least one source must exist for the content delivery network to get started. The source first advertises its bootstrap peer. It generates a PKI [11] public/private key pair and transmits its public key to the bootstrap peer. It then stands ready for data requests from clients. If a request from a client peer is received, it looks up the client's public key and uses a Diffie-Hellman key agreement algorithm [12] to produce a session key. The session is then used by the source to encrypt all data that is sent to the client. Peers that become "relay" peers use the same method to encrypt frames as they are sent to other peers.

Our prototype uses the Java implementation [13] of the SCTP [14] transport layer protocol. SCTP is serving in a similar role as the popular TCP and UDP protocols. It provides some of the same service features of both, ensuring reliable, in-sequence transport of messages with congestion control. We chose SCTP because of its ability to deliver multimedia in multiple streams. Once a client has established a SCTP association with a server, packages can be exchanged with high speed and low latency. Each association can support multiple streams, where the packages that are sent within one stream are guaranteed to arrive in sequence. Each source can divide the original video stream into set of streams meant to be displayed in an overlay fashion. Streams can be arranged in a way that the more streams are fully received by a client, the better the viewing quality will be. The first stream is used to deliver a basic low quality version of the video stream. The second and consecutive streams will carry frames that are overlaid onto the primary stream for the purpose of increasing the quality. In our framework we also use the additional streams to carry content that is "added value", such as advertising messages or identifying logos. The ultimate client that displays the content to a user will combine all streams into one viewing experience.



```
Peer: JohnMueller@
Peer: raimund@ege.
Peer: ellie@aol.com
Peer: root@localhos
Peer: LouDobbs@cn
Peer: EgeConsulting
Peer: raimund@ege.

Trust server started
Client request(java.la
Peer request from: /1
New peer added: Eg
Client request(java.la
```

```
SunJCE Diffie-Hellman Public Key:
y:
   3069bae3 cc890d87 88de329f b708e821 4df693cf 7c430e64 76164e07 cf6ba1ac
   b19e9371 04bbec60 8832f81f c70e4556 ab076658 a2834291 55c9b8b1 ed812d86
   af215460 f8082094 34801106 fb2d4c7d 0b317d11 ed0cbd5f d1cfb95a f8f27b19
   bb338c6e ec00981c 49637303 f0391359 419d086c e2e06a0e 9ce18c69 d2b171d7
p:
   a7c3c0b0 c6be8757 8f3ae0ce 16132f29 1c70e14a 8491ca97 41e5cdd2 01ab7645
   a750eaa0 859db7b9 25bb8eed af168331 fb0eea84 709ffbdd 6001323d af8be08e
   524fccf7 a4e15566 81bfc86f ab4889bf df458779 1d92557c f503bc81 7b1387a6
   ba9820dd b2623d89 f91d66d5 db6a6132 51918842 874de98f 2315d52a c2592335
g:
   a17c5070 3ed09720 fbc2cf9f c0563f9c 28df5629 4d8627c6 16b81aa6 c20d5a4e
   e4be6c91 7998f62a 46841ce4 00789551 849791db 2cbc5441 c5f7feb5 14705fc8
   0e613f54 ad100d5f 02bee678 0699e3b5 1cf96f4f de5b83a3 8d5b1f10 cf7fcdf0
   e3f6872f 740f2f89 10e40869 b6fd6ce2 d95f9904 c4fc3d8a ec79e634 b2f73adb
```

Figure 5. Peergroup listing and security info

V. Conclusion

In this paper, we described a framework for new content delivery networks that almost implements access control for its participating peers. We have described a prototype implementation that uses SIP messaging to establish a P2P network, where a group of peers disseminate information on which sources are available to download from, and includes a Java-based client for the Android platform for smart phones. Such P2P content delivery has great potential to enable large scale delivery of multimedia content. Our framework is designed to enable content originators to assess the potential reward from distributing the content to the Internet. The reward is quantified as the value added at each peer in the content delivery network and gauged relative to the actual cost incurred in data delivery but also correlated to the risk that such open delivery poses.

Consider the scenario we described earlier in the paper: a typical “viral” video found on a social networking site: the video is captured on the fly, then uploaded onto the site, stored and transmitted for free and viewed by a large audience. The only entity that is getting a reward is social media site, which accompanies the video presentation with paid advertising. The only benefit that the original source of the video gets is notoriety. Using our model, the original data owner can select other venues to make the video available via a peer-to-peer approach. The selection on who will participate can be based on how much each peer contributes in terms of reward but also risk. Peers will have an interest in being part of the delivery network, much like Facebook and YouTube have recognized its value. Peers might even add their own value to the delivery and share the proceeds with the original source. Whereas in the social media approach the reward is only reaped by one, and the original source has shouldered all the risk, i.e., lost all reward from the content, our model will enable a more equitable mechanism for sharing the cost and reward.

References

[1] Raimund K. Ege. Trusted P2P Media Delivery to Mobile Devices. Proceedings of the Fifth International Conference on Systems (ICONS 2010), pages 140-145, Menuires, France, April 2010.

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

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

[4] E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. In SACMAT ’01: Proceedings of the sixth ACM symposium on Access control models and technologies, pages 41–52, New York, NY, USA, 2001.

[5] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. ACM Transaction Database System, 26(2):214–260, 2001.

[6] OpenID, http://www.openid.net. [accessed September 22, 2010]

[7] C. Wu, Baochun Li. R-Stream: Resilient peer-to-peer streaming with rateless codes. In Proceedings of the 13th ACM International Conference on Multimedia, pages 307-310, Singapore, 2005.

[8] R. K. Ege, L. Yang, Q. Kharma, and X. Ni. Three-layered mediator architecture based on dht. Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 2004), Hong Kong, SAR, China. IEEE Computer Society, pages 317–318, 2004.

[9] L. Yang, R. Ege, Integrating Trust Management into Usage Control in P2P Multimedia Delivery, Proceedings of Twentieth International Conference on Software Engineering and Knowledge Engineering (SEKE'08), pages 411-416, Redwood City, CA, 2008.

[10] Open Handset Alliance, http://www. openhandsetalliance.com/. [accessed November 19, 2010]

[11] Gutmann, P., 1999. The Design of a Cryptographic Security Architecture, *Proceedings of the 8th USENIX Security Symposium*, pages 153-168, Washington, D.C., 1999.

[12] Network Working Group, Diffie-Hellman Key Agreement Method, Request for Comments: 2631, RTFM Inc., June 1999.

[13] java.net – The Source for Java Technology Collaboration, The JDK 7 Project, http://jdk7.dev.java.net. [accessed September 22, 2010]

[14] R. Stewart (ed.), Stream Control Transmission Protocol, Request for Comments: 4960, IETF Network Working Group, September 2007, http://tools.ietf.org/html/rfc4960. [accessed September 22, 2010]

[15] Raimund K. Ege, Li Yang, Richard Whittaker. Extracting Value from P2P Content Delivery. Proceedings of the Fourth International Conference on Systems (ICONS 2009), pages 102-108 Cancun, Mexico, March 2009.

[16] Sip2Peer, SIP-based API for robust connection and communication among peers, http://code.google.com/p/sip2peer/. [accessed May 9, 2011]