Integrating P2PSIP into Collaborative P2P Applications: A Case Study with the P2P Videoconferencing System BRAVIS

Ronny Klauck
Computer Networks and Communication Systems Chair
Brandenburg University of Technology Cottbus, Germany
Email: rklauck@informatik.tu-cottbus.de

Michael Kirsche
Computer Networks and Communication Systems Chair
Brandenburg University of Technology Cottbus, Germany
Email: michael.kirsche@tu-cottbus.de

***Abstract*—Collaborative applications, such as videoconferencing systems, allow communication partners to interact and communicate anytime and anywhere over the Internet. Today, such applications require central servers for various functions. Even decentralized Peer-to-Peer (P2P) systems still require servers for the invitation and localization of users. The Session Initiation Protocol (SIP) is usually used for videoconferencing systems as a signaling protocol with its registration and proxy servers. This paper introduces a use case where these central SIP servers are replaced in a collaborative application with the help of the decentralized P2PSIP protocol. In the paper, we evaluate different available P2PSIP candidates and describe the integration of a chosen candidate into the P2P videoconferencing system BRAVIS. We present new insights on the practical appliance of P2PSIP for real life collaborative P2P applications along with general remarks for the integration and improvement of P2PSIP.**

I. Introduction

Today's life is unthinkable without anytime and anywhere communication, collaboration, and cooperation over the Internet. Videoconferencing is one of the possibilities to enable interaction between people around the globe. While server-based conferencing systems are already well established, collaborative videoconferencing applications following the Peer-to-Peer (P2P) principle without the need for servers are still quite rare. However, the available P2P videoconferencing systems still rely on central servers for the invitation process and localization of communication partners. The Session Initiation Protocol (SIP) [1] is often used to enable these services. SIP was developed as a signaling protocol for Voice-over IP (VoIP) applications, for example. Current SIP-based videoconferencing systems (e.g. [2] or [3]) need functions of SIP servers, e.g. registration functions provided by a SIP registrar or forwarding functions provided by SIP proxies. Drawbacks of such servers are high costs for the deployment and the need for special staff for configuration and administration. This is the reason why SIP-based videoconferencing systems with central servers cannot be used to build small ad-hoc conferences for spontaneous collaboration without infrastructure. Another widely discussed drawback of client-server architectures is the single point of failure aspect. The P2P paradigm, where peers take the role of clients and servers at the same time, is normally used as a common approach to overcome this drawback. P2P systems can scale on demand and offer a high reliability due to their decentralized concept. To empower the original SIP client-server architecture with the P2P paradigm, the Internet Engineering Task Force (IETF) called for a combination of P2P and SIP: Peer-to-Peer SIP (P2PSIP).

P2PSIP combines the advantages of both P2P and SIP. In the P2PSIP architecture [4] peers are used to replace conventional SIP servers. Each peer stores and shares information in the Distributed Hash Table (DHT) and provides storing and routing services in the P2P network overlay. Hence, the main advantages of P2PSIP are: (1) cost efficiency and (2) backward compatibility. (1) There is no need for a special centralized server infrastructure because the functionality of SIP servers is shared across the peers to eliminate the single point of failure. (2) The peers are communicating through normal SIP messages, which allows them to connect to conventional SIP servers that do not need to be modified and can be used side by side with P2PSIP.

Different P2PSIP approaches have been proposed and studied extensively. In this work we present a case study of an evaluation of the different available P2PSIP approaches for the purpose of a practical integration and implementation of a chosen candidate into a collaborative application. The evaluation and implementation is made in accordance with the requirements of an example system: the collaborative P2P videoconferencing system BRAVIS, developed at the Brandenburg Technical University Cottbus, Germany. Through this evaluation and implementation process, new insights on the practical appliance of P2PSIP for existing collaborative applications are gained and presented in this paper.

The paper is structured as follows. P2PSIP and BRAVIS are introduced in Sections II and III respectively. The available P2PSIP candidates are described and evaluated in Section IV. The integration of a chosen P2PSIP candidate into BRAVIS is presented in Section V along with comments for possible enhancements. A practical performance analysis of the working P2PSIP integration is discussed in Section VI. Section VII presents related work while a concluding summary is given together with a short outlook on further work in Section VIII.

II. Introducing P2PSIP

P2PSIP is the combination of the P2P paradigm with the Session Initiation Protocol (SIP), which was originally designed by Henning Schulzrinne and Mark Handley and later on standardized in [1]. SIP, as a signalizing protocol, is commonly used for Internet conferencing, telephony, presence, event notification, and instant messaging. Refer to [1] and [5] for more information on the original SIP protocol components and functions. Current collaborative SIP-based applications (e.g. [2], [3]) typically use a SIP registrar and a SIP proxy for every domain. This causes high costs in acquirement and maintenance of such systems. Additionally these systems need special staff to administrate the server in the domains. For this reason, a traditional SIP communication system cannot be easily and quickly set up in environments like spontaneous conferences or emergency scenarios. Furthermore, such centralized architectures are vulnerable for the failure of the central components (single point of failure). I.e., if the SIP registrar or proxy fails new invitations are not possible and existing proxy connections will break. P2P systems in contrast are scalable and reliable because they offer a decentralized concept of peers and eliminate the single point of failure in their specification. By combining P2P and SIP, participants in a P2PSIP overlay become equal and communication (e.g. locating resources, users, or data) is done in a distributed fashion between the participants without a central infrastructure. The P2P principle also enables a new approach for collaborative applications like videoconferencing systems which can be designed and deployed without the need for special technical infrastructure (e.g. central data dissemination servers). Costs for such systems can thus be reduced.

In a P2PSIP architecture [4] peers are not just simply SIP User Agents (UAs) that allow users to make calls. UAs in P2PSIP also map the whole functionality of SIP servers on every participating peer. Through this combination P2PSIP peers combine and offer more functions than one centralized instance in a conventional SIP environment. Storage and routing service functionalities are covered and provided by each actively involved peer in the overlay. Every P2PSIP peer should speak the P2PSIP peer protocol to enable communication and self-organization of the P2PSIP overlay. The backward compatibility to plain SIP is also guaranteed, since P2PSIP peers communicate through ordinary SIP messages, which allows them to connect to conventional SIP servers while using existing unmodified infrastructure.

A. P2PSIP Reference Model

A P2PSIP overlay is a fusion of nodes in a P2P manor with the approach to realize real-time communication with SIP without the usage of centralized servers. The nodes offer therefore a location service on the top of a Distributed Hash Table (DHT) and a transport service for SIP messages between any nodes. The DHT is used to transform Addresses of Records (AoR) to contact Uniform Resource Locators (URLs). The P2PSIP overlay consists of several P2PSIP peers on which the DHT is mapped. Data from all nodes can be stored and read in a very efficient way with the help of the DHT. Every node maintains a local hash table which saves a part of the complete overlay data. Hashing is an algorithm for searching for specific data objects in large data amounts. Hashing uses a mathematical function to calculate the position of an object in a table. Replication of a set of data on more than one peer in a P2PSIP overlay can be used to prevent the loss of data in the case where a peer fails or leaves the overlay without notice. All peers are allowed to offer additional services, e.g. a Session Traversal Utilities for NAT (STUN) relay service. Such additional services can be published with an entry in the DHT, so that other nodes can gain information about available services. The functions and components of these P2PSIP peers are defined in the P2PSIP peer protocol, where Network Address Translation (NAT) traversal is also an element. NAT systems can hinder connections between peers, preventing a direct communication and SIP message exchange between participating peers. Two different solution approaches are currently discussed in the P2PSIP working group, but a specific NAT traversal solution is still in development.

Next to peers, so-called clients can also enter the P2PSIP overlay with the help of normal peers. The IETF P2PSIP working group is currently not consistent with the role of clients because pros and cons of such clients are not fully balanced. Clients can use services offered by peers, but they cannot participate actively in the P2PSIP overlay, i.e. they cannot store data or route messages. An additional client protocol needs to be specified if clients should be enabled inside a P2PSIP overlay.

Fig. 1 shows a P2PSIP overlay with several peers, a client, and a normal SIP UA (light gray). Every peer is responsible for the storage of resources and the routing of messages in the overlay. The normal SIP UA does not communicate directly with the overlay, it uses conventional SIP messages instead. Because this process is not standardized by the P2PSIP working group, individual solutions can be implemented. The client on the right side of the overlay communicates through the P2PSIP client protocol with his assigned peer Q. The SIP proxy of peer P, the SIP redirector of peer R, and the gateway of peer G can be used as interfaces between conventional SIP devices and the P2PSIP overlay. Every peer accepts standard SIP requests and resolves the next hop with the help of the P2PSIP routing services to handle such SIP request in the appropriate way. This process is bidirectional.

Message Flow (example for Fig. 1): When client C’s user wants to initiate a SIP session with user U, the following steps are performed in the P2PSIP overlay [6]:

1) Client C sends a request to peer Q to locate the user U.
2) Peer Q uses a lookup to find the user U in the overlay. As response peer Q gets the answer that the user U can be contacted through peer F.
3) Peer Q sends this response back to client C.
4) Now client C can send a SIP INVITE request to peer F to initiate a SIP session.

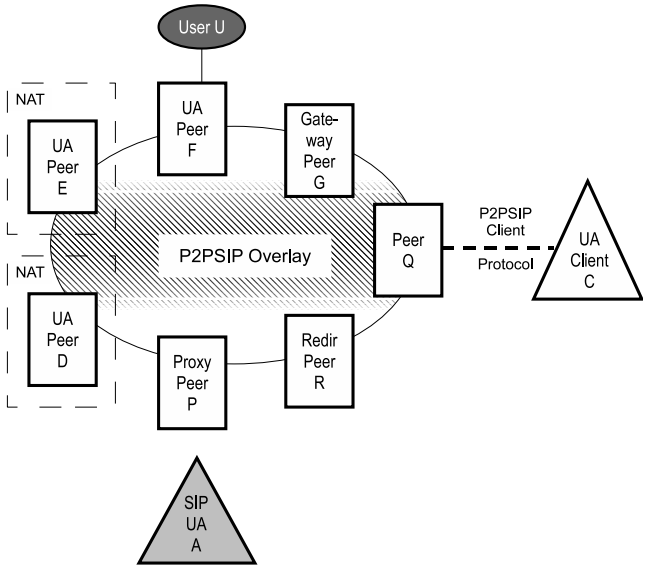

Fig. 1. P2PSIP reference model (based on [6])

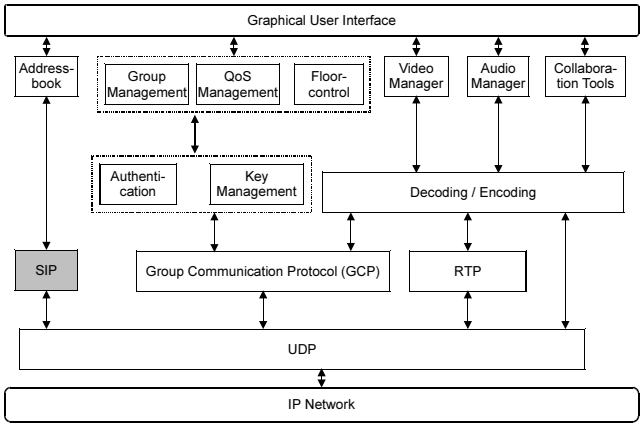

Fig. 2. BRAVIS architecture

III. BRAVIS

BRAVIS (BRAndenburg VIdeo conferencing System) is a P2P videoconferencing system for closed group conferences with up to 16 participants. It was developed at the Brandenburg Technical University Cottbus [3], Germany. We chose BRAVIS as an example system for the presented use case, since it enables cooperations of users through audio, video, and text conferences, and supports collaborative working over the included distributed whiteboard. It is therefore a good example for the class of collaboration-supporting P2P applications.

As a decentralized P2P system, BRAVIS enables a distributed group and Quality of Service (QoS) management. The BRAVIS architecture, as shown in Fig. 2, is split into a data transfer and a signaling part. The data transfer part is responsible for the transmission and distribution of audio, video, and data streams to the conference participants. The signaling part consists of the group manager, the QoS manager and the floor control. These components realize functions like a dynamic join and leave of groups, different collaboration tools (e.g. distributed whiteboard), securing of the closed conference group, or differentiated group view. For more information about the different components and functions of BRAVIS refer to [7].

New conference participants must be invited into a conference to enable closed groups. The invitation process is handled through a SIP invitation with the help of the SIP registrar and SIP proxies, if necessary. The SIP invitation is responsible for the localization of the user and the translation of the globally unique SIP Uniform Resource Identifier (URI) into a specific IP address. With the IP address received from the SIP server after a SIP invitation to the server, the Group Communication Protocol (GCP) starts a group session and creates the P2P overlay. The following communication is done either over the created P2P overlay (e.g. exchange of management data and synchronization between participants) or performed directly between the participants (e.g. media data exchange). BRAVIS is therefore depending on a SIP registrar server and SIP proxies in the invitation phase. This behavior is typical for many collaborative P2P applications. Skype, for example, also relies on a central server [8] for the initial contact phase and the invitation of users.

With the use of a SIP-based invitation, BRAVIS is more of a hybrid P2P system because the current version uses centralized SIP servers to locate users, while the rest of BRAVIS works decentralized. The decision for the SIP infrastructure support was made to allow for an easier commercialization. With the appearance of P2PSIP, BRAVIS can now use a P2PSIP-based localization without centralized infrastructure components, to build a more reliable, decentralized, and scalable architecture for the needs of the future (e.g. ad-hoc collaboration without infrastructure). P2PSIP can also help to eliminate the single point of failure in the localization and invitation process.

IV. Evaluation of Available P2PSIP Candidates

The Internet Engineering Task Force (IETF) collects drafts to realize SIP in P2P environments. The P2PSIP architecture [9] should enable the creation of free and cost effective communication systems for everyone without the influence of higher instances. One of the goals for the inclusion of P2PSIP into BRAVIS is the use of open standards to achieve a wide acceptance of our decentralized videoconferencing system architecture. Most of the proposed P2PSIP drafts [10] are still under development. This section presents an overview of the currently available P2PSIP implementations. Next to the introduction of available candidates, criteria essential for an integration into BRAVIS are introduced in the section. The chosen candidate P2PP is then presented together with the selection reasons.

A. Introduction of Available P2PSIP Candidates

Several drafts of P2PSIP implementations have been proposed to the IETF. Currently these candidates are: Cisco’s P2PSIP project, the P2PP project of the Columbia University,

the SIPDHT2 project, Kademlia dSIP of the University of Karma, Huawei’s P2PSIP implementation, P2PNS from the University of Karlsruhe, and the P2PSIP approach RELOAD from SIPeerior Technologies. These available candidates are introduced shortly in the following paragraphs:

1) CISCO P2PSIP Project: The CISCO P2PSIP project uses a binary P2P signaling protocol called Address Settlement by Peer-to-Peer (ASP). ASP supports Chord as a DHT algorithm, SIP for localization, STUN and TURN services, a security framework on the base of an abstract enrollment server and a protocol extensibility model [11]. The implementation, which is written in C++, seems to be clean and small but a good documentation for the code is not available. Further development also seems to have stopped, since the last code update (Revision 8266) was done in July 2007.

2) Columbia P2PP Project: The Open Source P2P VoIP and IM System OpenVoIP [12] consists of 1000 nodes and runs inside the PlanetLab testbed network on around 300 servers worldwide. The OpenVoIP system uses the P2PP protocol [13]. P2PP supports various DHT algorithms, e.g. Kademlia, Bamboo, and Chord. Routing can be done in an iterative or recursive manner. Additional functions of P2PP are the support for four different hash algorithms (SHA1, SHA256, MD4, MD5) and the support for NAT traversal. OpenVoIP uses STUN [14], TURN [15], and ICE [16] for NAT traversal. Nodes behind a firewall or NAT are restricted in their connections. P2PP allows them to route media traffic and calls through so-called relay peers. P2PP uses the external library PJNATH to provide this relay functionality. PJNATH detects if a node is behind a NAT or firewall at startup. The P2PP implementation is written in the programming language C++ and exists in the version 0.2. The project supports the operating systems Windows and Linux.

3) SIPDHT2 Project: SIPDHT2 [17] uses PCAN [18] as the DHT algorithm. PCAN provides a robust overlay even if most peers are behind a NAT system. SIPDHT2 supports and provides NAT traversal via STUN, data replication, a graphical user interface (GUI) and an overlay simulator. The SIPDHT2 overlay uses SIP messages for administration, e.g. SIP INVITE, to let possible nodes join the overlay. The integrated SIP stack is the Sofia-SIP library. SIPDHT2 offers a good documentation and is written in pure C. A main concern with this project is the strong dependency on external Linux-specific libraries like *avahi* and *glib*. Exportation of SIPDHT2 to other operating systems can therefore be complicated.

4) Kademlia dSIP: The dSIP [4] protocol uses the SIP syntax for signalizing. dSIP provides STUN, TURN, and ICE for NAT traversal through the use of conventional SIP messages. Chord and Kademlia are used as DHT algorithms. The use of SIP messages for signalizing means a higher complexity compared to a binary protocol. This aspect is currently discussed in the P2PSIP working group [19]. dSIP is implemented in the programming language JAVA, leading to a high availability on nearly all operating systems. The implementation is currently limited to SIP INVITE requests and development seems to have stopped (last updates in 2007).

5) Huawei’s SEP Peer and Client Protocol: Huawei’s implementation uses the Service Extensible Protocol (SEP) for communication between P2PSIP peers and maintenance of the DHT service. SEP distinguishes between a peer [20] and a client [21] protocol. According to the P2PSIP specification, peers offer routing and storing services and clients will not offer these services. The client protocol controls the behavior between a client and his allocated peer. Although the code of SEP is written in C++, it is very heavy and complex. Some parts seem to be optimized for the Windows operating system.

6) P2PNS - A Secure and Distributed Name Service for P2PSIP: The Peer-to-Peer Name Service (P2PNS) [22], [23] implements a distributed name service (like DNS) that uses a P2P overlay. The main goal of P2PNS is a secure and efficient SIP name resolution for decentralized VoIP. Currently, only an experimental implementation of P2PNS exists for the OverSim simulation framework [24]. Supported functions of P2PNS are various DHT algorithms (Chord, Koorde, Pastry, Bamboo, Kademlia, Broose), several routing strategies (semi-recursive, full-recursive, source-routing-recursive, iterative, exhaustive-iterative), and security enhancements for the iterative routing.

7) SIPeerior Technologies (RELOAD): SIPeerior Technologies [25] is developing its own DHT protocol called RELOAD (REsource LOcation And Discovery) [26]. The idea was inspired by the protocols dSIP and SoSIMPLE [27], but RELOAD uses a binary protocol for signalizing [19], comparable to the approaches of the STUN protocol. This project uses a commercial license and is still under development.

B. Selection Criteria

We identified several criteria suitable for our use case. To enable a proper candidate selection we used the following criteria, which were also proposed by [28]:

a) License model: The candidate should be published under a Open Source model license like GPL or BSD.

b) Platform independent: The candidate should support common operating systems, e.g. Windows, Linux, or BSD.

c) Programming language: The candidate should be programmed with a system-near and fast programming language, e.g. C or C++. This criteria is important for BRAVIS, since BRAVIS was written in the C programming language in order to ensure a slim and fast implementation.

d) Seamless integration: A seamless and easy integration into the existing BRAVIS architecture, as an example collaborative application, should be possible, so that main principals of the architecture do not need to be changed for the integration of P2PSIP.

e) Automatic configuration: User-driven configuration of the system should not be required. Automatic configuration should be an integral part, so that NAT and firewall systems are automatically detected and traversed, neighbor peers are automatically detected and the initial registration for the P2PSIP overlay is performed autonomously.

f) Heterogeneity support: The candidate should support various peers with different resources. It shall also include

the possibility to adapt to available resources and distinguish between participants with different network capacity and availability constraints. A distinction between nodes and so-called super-nodes is therefore favored.

g) DHT support: The candidate should use an efficient DHT algorithm, to optimize the search and lookup of resources in the distributed overlay. Since different DHT algorithms have different capabilities, e.g. churn suitability, candidates with various supported DHT algorithms are favored.

h) Scalability: A candidate should have a good performance. It should scale well with growing demands and increasing numbers of participants.

C. Selecting a Candidate: P2PP vs SIPDHT2

Coming back to our criteria, only SIPDHT2 and P2PP comply with them. SIPDHT2 seems to be a good choice for developers who want to start a project from scratch because SIPDHT2 uses standard SIP messages for self-administration, which cannot be used by the application. Instead it is possible to send simple text messages from one peer to another using the SIP MESSAGE method. Because SIPDHT2 also includes a predefined SIP stack, developers cannot chose their own stack without getting redundant code. P2PP seems to be the better choice for the BRAVIS architecture: we can reuse the available SIP stack and SIP messages compatible with RFC 3261 [1] will be routed correctly in the P2PP overlay. Another important aspect is that P2PP supports various DHT algorithms, which can be easily upgraded or replaced by new ones in the future. Migrating an application from SIP to P2PSIP should be easy with P2PP because we can reuse the SIP stack and keep the concept behind the migrating application. P2PP has another advantage: it uses an automatic NAT detection based on the STUN, TURN, and ICE protocols, while SIPDHT2 tries to find a way around the firewall or NAT by simultaneous session establishment (works like symmetric RTP [29]). SIPDHT2 uses a lightweight protocol, called XPP [18], which is implemented in C. This is an advantage especially for applications on mobile devices, which have only limited resources. An overview of these aspects is given in Table I.

D. Introducing P2PP

P2PP is an application layer protocol [13] where participating nodes are represented by a human user. The application layer protocol supports structured or unstructured peer protocols to form an overlay. In P2PP, nodes can act in a peer or client mode: a peer is an active member in the overlay and provides storage and routing services to other nodes in the overlay. A client does not offer such services, but it can simply use the services of other peers in the overlay while communicating with a STUN relay peer. A STUN relay peer also offers NAT traversal services to clients. CPU, memory, uptime, and network connectivity are some criteria which determine whether the node can act as a peer or a client.

The protocol stack of a P2PP node consists of three layers: an application, an overlay, and a transport layer. The application layer defines an Application Programming Interface (API) to the overlay layer which allows access to functions like joining, leaving, or searching for peers or objects. The overlay layer includes mechanisms for routing, overlay maintenance, replication, NAT traversal, and storage management. The transport layer supports transmission of messages over an unreliable (e.g. UDP and DTLS) or a reliable (e.g. TCP or TLS) communication channel. For unreliable transport, P2PP provides an ACK-based hop-by-hop reliability mechanism.

V. Integrating P2PSIP into BRAVIS

We take several aspects from the comparison of P2PP and SIPDHT2 into consideration: First of all, we want a solution for BRAVIS in which we can reuse the already integrated SIP stack oSIP [30]. Second, we want to avoid modifications in the architecture of BRAVIS and the basic design principles. Through these aspects we want to show and prove the possibility of a seamless migration of current SIP-based collaborative P2P applications to server-free versions with the help of P2PSIP, eliminating most of their current drawbacks. We also want to use the special features that P2PSIP approaches like P2PP bring along (e.g. automatic NAT detection, support for several DHT algorithms).

A. BRAVIS System Environment

BRAVIS uses SIP messages to register a user at the localization service through a SIP registrar and to invite users to conference sessions with the help of a SIP proxy server. These are the central components that we want to replace with the P2PSIP implementation. The BRAVIS architecture is designed in a modular fashion [31], so that code parts can simply be exchanged or upgraded. All modules work in one process group with one or more related processes. The conference control module exchanges message queues to signal invitations for participants.

B. Integration Considerations

A SIP network contains user agents and servers (e.g. proxy and localization). The SIP registrar stores, changes, and updates SIP URIs in the database of the localization server. This process can be adopted by the P2PSIP DHT. Accessing entries in a DHT is easy and fast by using hashes for each entry (in our case the SIP URIs). Each P2PSIP peer in the overlay will replace the SIP registrar and the SIP proxy servers while providing functions like register, leave of users, and routing of messages. Every participant needs to publish his/her IP address under his/her SIP URI in the P2PSIP overlay so that other users can find him/her. A connection can only be established between users if all conference participants are online and connected to the Internet. This scheme follows the rendezvous principle and no SIP server is needed. A user can request information about the current IP address of a partner from the P2PSIP overlay by using the SIP URI of the partner. With the gained IP address, he/she can build a direct connection and start a communication session. If Alice, for example, wants to talk to Bob then Alice and Bob need to publish their IP addresses in the P2PSIP overlay. This can be done

TABLE I
OVERVIEW OF THE COMPARISON BETWEEN P2PP AND SIPDHT2

Characteristic	P2PP	SIPDHT2
Appliance	Useful for integration or migration of P2PSIP in existing projects: Reuse of an already integrated SIP stack	Useful for new projects and developments: Define a new signalizing concept without a separated SIP stack
NAT traversal	Automatic NAT detection and switching between client and peer mode: Clients use a peer to bypass NATs	Simultaneous session establishment to bypass firewall and NAT systems: Uses no real NAT detection
Additional features	Support for several DHT algorithms: Individual choice of a DHT like Chord, Kademlia, or Bamboo	Interesting for mobile applications: Efficient usage of resources which are very restricted on mobile platforms in their battery life, processor power and memory
Programming language	C++ with objects: Realize complex applications	Pure C: Fast and clean source code for small applications

automatically, when both users are joining the P2PSIP overlay. In the next step the user agent of Alice can look for the IP address of Bob by submitting Bob’s SIP URI. After the request to the P2PSIP overlay is answered with a response message, which includes Bob’s IP address, both users can establish a direct connection. Fig. 3 depicts this example.

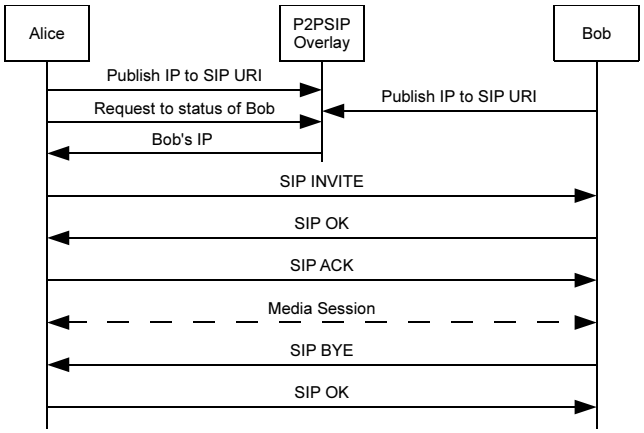

Fig. 3. Message flow for the replacement of SIP with P2PSIP

We used the P2PP project in version 0.2 [32] for the integration of P2PSIP functions into BRAVIS. Version 0.2 of P2PP has an integrated support for STUN, TURN, and ICE, realized through the external library PJNATH in version 0.8. Therewith BRAVIS can reuse its implemented SIP functions, while some of them are replaced with equivalent P2PP functions in order to get rid of the SIP servers. The two main SIP methods INVITE and REGISTER will be exchanged; more details on this are given in the next subsection and in Table II.

TABLE II
OVERVIEW OF REPLACEMENT OF SIP METHODS WITH P2PP

SIP registrar / proxy	P2PP overlay
Register / Unregister	Store user information automatically during join/leave in the P2PP overlay
Send SIP INVITE to a user	1. Lookup request to P2PP overlay to get IP and port of invited user 2. Send SIP INVITE message over P2PP overlay directly to user

C. Changes in the BRAVIS Architecture

While planing the integration of P2PP in our example application, we focused on reusing as much of the original BRAVIS structure and code as possible. On the one hand, we did not have to implement a new SIP stack. This enabled us to stay with the obtained principles of our architecture. On the other hand, this project is a good example to show how SIP-based applications can be ported to the usage of P2PSIP. Only small modifications in the signaling layer of the BRAVIS architecture were necessary for the inclusion of P2PP, as depicted in Fig. 4. The signaling layer was extended with the P2PP module (marked dark gray) and the SIP module was placed into the background. The localization process now runs over P2PP instead through SIP REGISTER messages. Only SIP INVITE messages will be routed by the P2PP module. These SIP messages will be created from the SIP module and then sent to an UDP socket where a P2PP listener is able to catch and route them over the P2PP overlay.

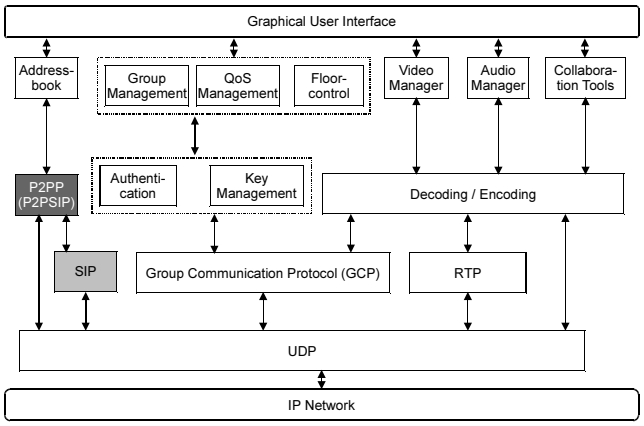

Fig. 4. BRAVIS architecture with integrated P2PP

D. Migration of Collaborative P2P Applications

Generally all changes we made to the BRAVIS architecture followed this simple rule: Remove all SIP functions for registration with the SIP registrar and replace all functions for sending messages over the SIP proxy with functions from the P2PSIP implementation. The SIP location service is built in the P2PSIP architecture via DHT. The P2PP module is

started by an own thread at the startup process of BRAVIS. P2PP makes use of callback functions as design pattern. This enables an access to all interface functions without using Inter-Process Communication (IPC) or a shared memory. No other modifications were made to the P2PP module. It is running out of the box. SIP-based applications can be migrated to P2PSIP, following our way because the applications can reuse their own integrated SIP stacks and they can easily start P2PP as running process in the background.

E. General P2PSIP Specification Problems

During our work with P2PSIP and the various P2PSIP candidates we came across several flaws in the specification of P2PSIP. We summarize these flaws in this subsection and give general advice on possible solution approaches.

The IETF given specification of the P2PSIP architecture describes exactly how the P2PSIP overlay should work, but it does not specify which bootstrapping or security mechanism should be used. The reason is that the P2PSIP working group [19] only concentrates on describing which format of messages should be used for the signaling process in the overlay.

1) Bootstrapping: Most P2PSIP implementations from the IETF, e.g. dSIP, SIPDHT2, or P2PP, use a so-called bootstrap peer. This is a special peer in the overlay with a static IP address under which joining peers can request IP addresses of peers in their neighborhood. This is a simple solution for the bootstrapping process, but it creates a new centralized instance in the overlay. If the bootstrap peer fails, new nodes cannot join the overlay because IP addresses of neighbor peers can no longer be requested. In a P2P environment all peers should be equal and provide the same services, but the bootstrap peer creates a special centralized instance with an outstanding function that provokes the already mentioned single point of failure. A centralized bootstrap peer is thus again a hidden server that should be avoided in P2P and P2PSIP environments.

The P2PSIP working group proposes a multicast discovery mechanism and a node cache as alternatives to the centralized bootstrap peer. Research on finding a decentralized bootstrapping process [33] showed that both operations are not adequate. The problem is that most routers in the Internet are not multicast-ready and a node cache makes only sense if IP addresses will not change. Home users of videoconferencing systems, for example, will get a new IP address assigned by their Internet Service Provider (ISP) each time they connect to the Internet. Therefore entries in a node cache become obsolete after a short amount of time. For BRAVIS with its inconstantly joining and leaving peers, this means that we need another approach for bootstrapping or that we need to use more than one bootstrapping process side by side to avoid a dependency on central instances in BRAVIS.

The Dynamic Domain Name System (DDNS) [33] could provide a solution for the bootstrapping problem. DDNS provides a name system for P2P applications that works like the Domain Name System (DNS) for the Internet. The idea is that peers in the overlay can dynamically change corresponding IP addresses to host names for P2P services. Joining peers can use the host name to a P2P service to determine IP addresses of active peers in the overlay. If a joining peer does not find an IP address under the requested host name then this peer is the first participant of a new overlay. The peer should then store its IP address in the name service. All peers in the overlay check regularly if the assignment of the IP addresses to the host name is still valid. If this is not the case an active peer needs to store its IP address in the name service. This approach has the advantage that it can be done automatically by the peers and needs no further interaction from users. Relying on the Internet's domain name system is practical because the DNS, as the backbone of the Internet, is sufficiently reliable. If DNS stops working, most Internet traffic will go down and web applications will stop working properly.

2) SIP Fallback Mechanism: In the case where bootstrapping fails at startup, a SIP fallback mechanism could be used. Applications will then register to the old SIP environment. A problem of this approach is that peers who already joined the P2PSIP overlay can only be found over the P2PSIP overlay and not over the SIP servers, since their IP addresses are not registered at the SIP registrar server. The P2PSIP specification allows that conventional SIP servers talk with the P2PSIP overlay through a gateway or SIP messages. Such a gateway or SIP message functionality can be used to spread the IP addresses of joined peers around the overlay and around the conventional SIP registrar infrastructure. Another advantage of this fallback mechanism is that it enables a soft migration of collaborative client-server applications to collaborative P2P applications. Existing SIP-based client-server applications may work without notice in combination with new P2PSIP applications. This enables a step-by-step migration to P2PSIP.

3) Security Aspects: The authentication of users in BRAVIS [34] is secured over the Internet Key Exchange (IKEv2) protocol to save users from malicious or unwanted participants. This security solution only works after a user is located with the BRAVIS invitation system. In a P2PSIP overlay it is possible that malicious peers could disturb the localization by deleting informations or messages, storing wrong information about the user status in the DHT, or simply routing messages to wrong participants. Malicious users could also fake user accounts and prevent real users from getting correct SIP INVITE messages. The whole invitation system of BRAVIS would not work in this case, a solution for the protection of user information inside the P2PSIP overlay must hence be developed.

Another problem is that before a peer can join the P2PSIP overlay, it should obtain a unique peer ID and necessary certificates which will proof the peer's uniqueness and authenticity. The P2PSIP specification does not describe how this process should be handled for peers. Normally, a centralized Certificate Authority (CA) is used, but this central instance is again not useful for P2P systems. The implementation of P2PNS [19] does not rely on centralized security infrastructure components. P2PNS uses a fundamental new solution which provides high security and works fully decentralized. It is based on the Kademlia overlay and extends it with additional

security mechanisms. Unfortunately, P2PNS exists only as an simulation model for the OverSim framework.

Inspired by the decentralized security architecture from the P2PNS projects [19], [23] we want to provide some enhancements for the P2PP project. Both projects have in common that they use Kademlia as a DHT algorithm for the overlay. Through the Kademlia similarity, some ideas can be transfered from P2PNS to P2PP.

If a user wants to read out status information from another user in the P2PSIP overlay, he/she has to send a lookup request to the overlay. Malicious peers could respond with wrong information in this case. A possible solution approach against this spreading of false information could be the provision of various disjunct paths to the goal peer where the requested information is stored. The provision of these short and disjunct paths is handled by the overlay itself [19]. The Kademlia protocol in connection with disjunct paths can increase the success of lookups significantly in a network with malicious peers. P2PP supports the necessary iterative routing out of the box, but iterative routing is two times slower than recursive routing. Therefore the proposed security benefit comes together with a performance drawback.

A malicious peer can manipulate all data that it stores personally. To prevent the delivery of manipulated data by malicious peers, replica to a set of data should be stored on more than one peer. If a peer requests data, it should also get all available replica (or at least a significant number of replica). With the help of a majority decision [19], it is possible for the requesting peer to find out which values are credible and which values might be manipulated. P2PP already contains a function for replication; every time a peer joins successfully, it shares the data of its key room with his new neighbor. What we need to do is to enhance the response function for a lookup in a way that all (or many) replica will be send back to the requesting peer. Second, we need an implementation for a majority decision in P2PP directly or in the application.

Another solution to protect the integrity of content in P2PSIP is using cryptographically generated SIP-URIs [35]. This mechanism is independent from the P2PSIP architecture and is usable for all available SIP stacks. The basic principle is build upon a so-called self-certifying identity that can verify itself without using an external authority (e.g. a central CA). The identity is mapped as a hash of a public key. For a P2PSIP overlay, a possible ID candidate would be the SIP-URI. Further information on this mechanism can be found in [35]. For the SIP-URI, the self-certifying identity uses a hash that looks like a random number or word to a user and cannot be read user-friendly. While the SIP-URI is generated, the user cannot choose his/her own defined SIP-URI. Because remembering an impersonal string is not very common today exchanging these generated SIP-URIs can be difficult.

SIP messages could be deleted or wrong routed by malicious peers. This would affect the invitation phase of BRAVIS. One solution for this problem could be SECURE SIP [36] which ensures that SIP messages are transfered in the overlay in the right way. Instead of SIP URIs we need to use SIPS URIs [1] which employ Transport Layer Security (TLS) between each hop pair to validate and secure the connection. In a P2PSIP communication it thus guarantees the authentication of identities that talk directly with each other. This solution needs support for SIPS by the P2PSIP implementation, otherwise there is no routing possible.

VI. Performance Analysis

Our two test environments are the P2PP PlanetLab overlay, which consists of nearly 1000 STUN relay peers, and a local P2PP overlay at the Brandenburg University of Technology Cottbus, Germany, which is composed of two STUN relay peers and one bootstrap peer. The P2PP PlanetLab overlay represents a worldwide distributed network of peers, while our local P2PP overlay represents a local group of peers which is a very common scenario for the usage of BRAVIS in companies, for example. Both overlays use the P2PP version 0.2 with integrated NAT detection support. The P2PP PlanetLab overlay enables the simulation of worldwide invitations for BRAVIS without setting up such a big network for ourselves. The goal of the analysis was to get a relation between the query times of the worldwide to the local P2PP overlay. From the result of the average query time for each overlay, we can say how our improved BRAVIS architecture would scale or be usable for invitations to a video conference spread around the world. We used a modified version of the sample program `p2pmain.cpp` (part of P2PP [32]) to determine the values for each overlay. We also used the `gettimeofday()` function from the programming language C to measure the time of a query.

Our modified test program was behind a NAT system to simulate a typical Internet home user who uses a dynamic IP address and a broadband connection with a router to access the Internet. Another important aspect is that in this constellation we need a STUN relay peer, which is an active member of the P2PP overlay, to interact with the P2PP overlay in client mode. This has the advantage that our values coupled with the `userID` are stored directly in the overlay and not locally because only the peer mode allows the offering of routing and storage services in the overlay. For each overlay, we performed over 10.000 runs. One run consists of the following sequence:

1) Join the P2PP overlay with a random `userID`. Through these actions the IP address and SIP URI will be stored in the overlay. The SIP URI is build of `userId@yourDomain`.
2) After the joining process was successful, send a lookup request for the SIP URI of our `userID`.
3) Save the query time and leave the P2PP overlay.

The average query times for each P2PP overlay are depicted in Fig. 5. These values only apply to the query itself and do not include the response message and the data of the respond. This is caused by the use of callback functions in the P2PP module which cannot be recognized by the C function `gettimeofday()`.

As a result of our performance test, we can say that the P2PSIP implementation P2PP can be used to locate users

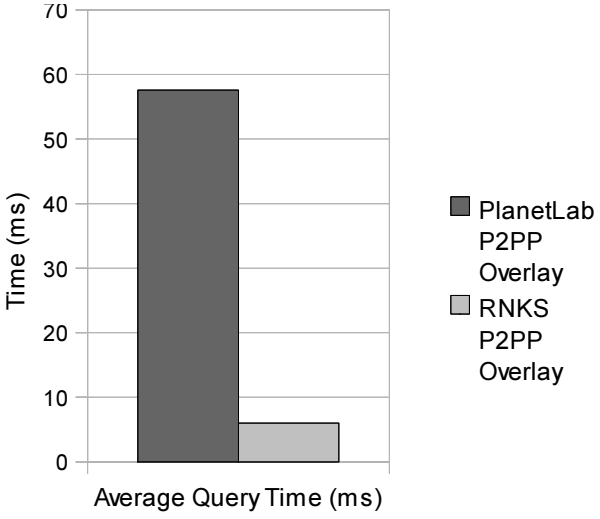

Fig. 5. Performance analysis - P2PP query times: worldwide vs. local

in a decentralized way. Comparing the gathered times from the local overlay to the times from the worldwide PlanetLab overlay, we can see that the local overlay is around 10 times faster because the nearest STUN relay peer is allocated in the local overlay compared to the remote STUN relay peer in the PlanetLab overlay. Getting the nearest STUN relay peer for the P2PP clients is therefore essential to achieve a fast query time in the P2PSIP overlay.

VII. Related Work

There is a large amount of literature available on P2P, SIP, and P2PSIP. Examinations of the different P2PSIP candidates were already referenced in Section IV-A. Further research works in the area of practical appliance of P2PSIP for SIP-based collaborative applications are otherwise limited. The OpenVoIP project [12], an open source P2P VoIP and IM system, has integrated P2PP in OpenWengo-2.2.1 [37], a free VoIP software, to demonstrate how P2PP can work in a real-life application. With this modification, users in the buddy list can be found over the P2PP overlay and media sessions can be established between active users. Supported media session protocols are SIP and the Real-Time Transport Protocol (RTP).

The integration of the P2PP protocol in OpenWengo enables the software to invite users and realize media sessions over a P2P overlay. However this is not very advisable for a closed group communication system like BRAVIS, because media sessions could be monitored by every peer and confidential communication could be revealed. BRAVIS uses its own secure group communication protocol [34] to ensure that only invited and trusted participants get the provided informations. In contrast to OpenWengo, BRAVIS is a real-life application that uses the P2P paradigm for the communication during a media session. We therefore focus on the use of P2PP to forward SIP messages in a P2P manor during the BRAVIS invitation phase.

VIII. Conclusions

This paper presented a use case where a SIP-based collaborative P2P example application was extended with a P2PSIP approach to remove the central SIP servers for localization and invitation. We described an approach on how SIP-based applications can be migrated to server-free P2PSIP-based applications. We introduced the available P2PSIP implementations from the IETF and evaluated them under consideration of various criteria which helped us to select the right implementation for our situation, needs, and given architecture. As a result of the integration process, we acquired a pure P2P application which uses P2PSIP to route and send SIP messages through a P2PSIP overlay. We also described the problems of the current centralized bootstrapping process of the P2PSIP protocol and proposed the integration of a Dynamic Domain Name System (DDNS) to solve this problem.

In addition to the described bootstrapping problem, we showed that several security flaws (e.g. non-encrypted SIP URIs or injection of malicious messages) exist in the P2PSIP specification. We proposed several enhancements with the help of existing approaches like SECURE SIP or self-certifying identities. These enhancements go hand in hand with the security concept of our example application, the collaborative P2P videoconferencing system BRAVIS.

Through the evaluation and integration of P2PP in BRAVIS, we gathered new insights on the applicability of P2PSIP for collaborative SIP-based applications and showed that a soft migration of client-server based applications to P2PSIP applications is possible. Further work in this area will be the integration of the enhancements, which we proposed in the paper, into the P2PSIP implementation in the videoconferencing system BRAVIS and additional testing in various scenarios.

Acknowledgment

The authors would like to thank the Chair of Computer Networks and Commuication Systems, Brandenburg Technical University of Cottbus, and Dr. Fuwen Liu for his start-up aid and help on various occasions.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “Sip: Session initiation protocol,” IETF, Request for Comment 3261, June 2002.

[2] K. Singh and H. Schulzrinne, “Peer-to-Peer Internet telephony using SIP,” *New York Metro Area Networking Workshop*, Sept. 2004.

[3] “BRAVIS (BRAndenburg VIdeo conference System),” [Online] http://www.bravis.tu-cottbus.de/, 2009.

[4] D. Bryan, B. Lowekamp, and C. Jennings, “dSIP: a P2P approach to SIP registration and resource location,” P2PSIP, Internet-Draft, Feb. 2007.

[5] H. Sinnreich and A. B. Johnston, *Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol*, 2nd ed. John Wiley and Sons, 2006.

[6] D. Bryan, P. Matthews, E. Shim, and D. Willis, “P2PSIP concepts and terminology,” *IETF67*, Nov. 2006.

[7] H. Koenig, D. Rakel, F. W. Liu, and M. Kirsche, “P2P video conferences for closed groups,” *PIK - Practice of Information Processing and Communication*, vol. 30, no. 4, pp. 219–226, Dec. 2007, *(in German)*.

[8] P. Biondi and F. Desclaux, “Silver needle in the Skype - Black-Hat Europe 2006,” [Online] http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-biondi/bh-eu-06-biondi-up.pdf, Mar. 2006.

[9] S. D. Pundkar, “Peer-to-Peer communications over the Internet: An open approach,” Master Thesis, Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, India, July 2007.
[10] IETF, “P2PSIP implementations,” [Online] http://www.p2psip.org/implementations.php, 2009.
[11] C. Jennings, J. Rosenberg, and E. Rescorla, “Address Settlement by Peer-to-Peer,” P2PSIP, Internet-Draft, July 2007.
[12] S. A. Baset, “OpenVoIP: An Open Peer-to-Peer VoIP and IM System,” [Online] http://www1.cs.columbia.edu/~salman/peer/, 2009.
[13] S. Baset, H. Schulzrinne, and M. Matuszewski, “Peer-to-Peer protocol (P2PP),” Network Working Group, Internet-Draft, Nov. 2007.
[14] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, “Session traversal utilities for NAT (STUN),” IETF, Request for Comment 5389, Oct. 2008.
[15] J. Rosenberg, R. Mahy, and P. Matthews, “Traversal using relays around NAT (TURN): Relay extensions to session traversal utilities for NAT (STUN),” BEHAVE WG, Internet-Draft, Feb. 2009.
[16] J. Rosenberg, “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,” MMUSIC, Internet-Draft, Oct. 2007.
[17] “SIPDHT2,” [Online] http://sipdht.sourceforge.net/sipdht2/index.html, 2009.
[18] E. Marocco and E. Ivov, “XPP extensions for implementing a passive P2PSIP overlay network based on the CAN distributed hash table,” Network Working Group, Internet-Draft, Nov. 2007.
[19] I. Baumgart, “New developments in the area of decentralized Voice-over-IP networks,” *PIK - Practice of Information Processing and Communication*, vol. 30, no. 4, pp. 199–205, Dec. 2007, *(in German)*.
[20] X. Jiang, H. Zheng, C. Macian, and V. Pascual, “Service extensible P2P peer protocol,” P2PSIP, Internet-Draft, Feb. 2008.
[21] Y. Song, X. Jiang, H. Zheng, and H. Deng, “P2PSIP client protocol,” Network Working Group, Internet-Draft, Feb. 2008.
[22] “P2PNS,” [Online] http://www.p2pns.org, 2009.
[23] I. Baumgart, “P2PNS: A Secure Distributed Name Service for P2PSIP,” in *Proceedings of the Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008)*, Mar. 2008.
[24] “OverSim Framework,” [Online] http://www.oversim.org, 2009.
[25] “SIPeerior Technologies,” [Online] http://www.sipeerior.com, 2009.
[26] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, “REsource LOcation And Discovery (RELOAD),” P2PSIP, Internet-Draft, July 2008.
[27] D. Bryan, B. B. Lowekamp, and C. Jennings, “SOSIMPLE: A serverless, standards-based, P2P SIP communication system,” *Proc. of the 2005 International Workshop*, June 2005.
[28] K. Singh and H. Schulzrinne, “Peer-to-Peer Internet Telephony using SIP,” *Columbia University Technical Report CUCS-044-04*, Oct. 2004.
[29] D. Wing, “Symmetric RTP / RTP control protocol (RTCP),” Network Working Group, Request for Comment 4961, July 2007.
[30] “The GNU oSIP library,” [Online] http://www.gnu.org/software/osip/, 2009.
[31] D. Rakel, “Media dissemination in the videoconferencing system BRAVIS,” Diploma Thesis, Computer Networks and Communication Systems Chair, Brandenburg University of Technology Cottbus, Germany, 2004, *(in German)*.
[32] S. A. Baset, “P2PP version 0.2,” [Online] http://www1.cs.columbia.edu/~salman/peer/0_2.html, Aug. 2008.
[33] M. Antonovic, “Decentral bootstrapping approaches for peer-to-peer systems,” Student Research Project, University of Stuttgart, 2007.
[34] F. Liu and H. Koenig, “Secure and Efficient Key Distribution for Collaborative Applications,” in *Proceedings of the First IEEE Conference on Collaborative Applications (CollaborateCom 2005)*, Dec. 2005.
[35] J. Seedorf, “Using cryptographically generated SIP-URIs to protect the integrity of content in P2P-SIP,” in *Third Annual VoIP Security Workshop, Berlin, Germany*, June 2006.
[36] U. Trick, *SIP, TCP-IP, and Telecommunication Networks*, 2nd ed. Munich, Germany: Oldenburg, 2005.
[37] “Openwengo,” [Online] http://www.openwengo.org, 2007.