Session Initiation Protocol Deployment in Ad-Hoc Networks: a Decentralized Approach

Simone Leggio, Jukka Manner, Antti Hulkkonen, Kimmo Raatikainen

Department of Computer Science
University of Helsinki, Finland
Email: {simone.leggio, jukka.manner, antti.hulkkonen, kimmo.raatikainen}@cs.helsinki.fi

Abstract

Ad-hoc networks constitute a peculiar computing environment, characterized by the lack of centralized support from pre-existing network entities. Applications and protocols designed for centralized environments must be adapted for use in ad-hoc environments. For example, the baseline Session Initiation Protocol (SIP) strongly relies on the presence of an infrastructure, the SIP servers, and cannot therefore be deployed as it is in ad-hoc networks. This paper proposes a solution that enables devices in ad-hoc networks to use SIP functionalities in a decentralized way. Particularly, we embed a limited set of SIP server functionalities in the end devices to allow distribute session management for SIP end devices, without network support.

I. Introduction

Ad-hoc networks have been subject of research since the '70s. Initially the purpose of the study was for military applications, e.g., for quickly setting up communication in battlefields or where no support from infrastructure was available. Recently, developments in the capabilities of mobile devices (laptops, PDAs, mobile phones) and new wireless networking technologies, such as Bluetooth and IEEE 802.11 WLANs, have pushed the research towards other fields of interest as well. Ad-hoc networks formed by mobile devices, mobile ad-hoc networks (MANETs), are gaining a growing interest due to their flexibility and low deployment costs. MANETs can be formed autonomously and on-demand by mobile devices that connect to each other to form an independent communication network without any pre-existing infrastructure support.

A wider diffusion of MANETs will only be possible if applications extended for use in MANETs are developed. MANETs are often networks of peer devices, and very interesting applications are based on a peer-to-peer networking scheme. An attractive scenario could take place in an airport. A user, waiting for her flight to leave, looks for someone else in the MANET willing to begin, say, a gaming or chat session. Another scenario could be a conference presentation or a lecture. The participants connect their laptops in ad-hoc mode to share the slides, an electronic white board is shared and comments can be sent to the group or private comments can be exchanged via instant messages. One of the participants offers through the laptop external connectivity to the Internet, and invite colleagues far away to join the meeting and look at the slides and the minutes.

The common denominator of such applications is that they all require the establishment of sessions before the actual data flow can begin. The Session Initiation Protocol (SIP) [7] has emerged in IETF as the session management protocol, and it was also chosen as the signaling protocol in the 3GPP framework. SIP relies on centralized entities, namely servers, maintained by organizations or service providers. SIP servers handle users' registration, their location, and forward SIP messages to the location(s) where the recipient is reachable.

Networks of peer nodes, created on-demand without support of infrastructured networks such as MANETs are not a feasible environment for the deployment of the baseline SIP protocol. SIP end devices (user agents) in a MANET do not have the means to contact other nodes because the assistance of the servers is missing.

In this paper we propose an architecture for enabling SIP in ad-hoc networks. The main design idea is that SIP user agents have a limited set of SIP server functionalities embedded. The SIP network architecture is made decentralized, as the support that is originally provided by centralized entities is now merged in all the end devices.

Other groups have studied SIP over ad-hoc networks,

too. In [4], the authors suggest the use of broadcast REGISTER messages for spreading users' SIP information in ad-hoc networks. REGISTER messages are included in the broadcast messages used in a given underlying ad-hoc routing protocol. In [1], the focus is in the integration of SIP services with an underlying ad-hoc routing protocol. However, these approaches focus on the integration of SIP and the underlying routing protocol and do not take several SIP-related functionalities, such as efficient registration handling, into account. We instead aim to define a framework for a more generic and flexible deployment of SIP in a decentralized environment, such as ad-hoc networks.

The rest of the paper is divided as follows. Section II explains in more detail the problems of SIP in ad-hoc networks. Section III describes two approaches for operating SIP in decentralized ad-hoc networks. Section IV describes our software implementation forming the architecture for decentralized SIP. Section V provides more implementation details, and shows three scenarios where the architecture has been tested. Section VI gathers our observations about the main issues raised in this paper and presents pointers for future work.

II. Problem Statement of Decentralized SIP

The architecture of the Session Initiation Protocol (SIP) [7] is based on centralized entities. Two logical elements play a key role in the architecture, *registrar* and *proxy servers*. Registrars are the SIP entities where SIP users register their contact information once they connect to the SIP network. In a basic registration scenario, a SIP user agent communicates to its registrar server (the registrar IP address is usually preconfigured) the SIP user name of the user(s) using the device, referred to as SIP *address of records* (AOR) for that user, and the addresses where the user is reachable. Usually, contact information is stored in the form of IP addresses or resolvable names, but other kinds of contact information, such as, telephone numbers can also be used.

An association between a SIP AOR and a contact address is called a *binding*. SIP registrars exploit an abstract service, called *location service*, and return the bindings for the SIP AORs falling under their domain of competence to the SIP entities issuing a binding retrieval request.

Proxy servers are needed because SIP users cannot know, in the majority of the cases, the current complete contact information of the callee but only its AOR. SIP presupposes that the AOR (SIP user ID) of the party to contact is known in advance, analogously to what happens when sending instant messages or e-mails. A basic SIP session involves the calling user agent contacting the calling side proxy server, which in turn will forward the message to the proxy server responsible for the domain of the called user agent. The called side proxy server retrieves from the called side registrar (i.e. utilizes the location service) the bindings for the called user and eventually delivers the request to the intended recipient.

Registrars and proxies are logical entities, and it is not an uncommon configuration for them to be co-located in the same node. Usually, user agents have a preconfigured outbound proxy server where all the outgoing requests are sent and through which all the responses to the issued requests, or new requests, are received.

The described architecture is clearly not applicable to MANETs, as they are dynamic networks formed by peer nodes while proxies and registrars are fixed, static and centralized entities. As a result, the SIP protocol as it is cannot be deployed in isolated ad-hoc networks. SIP users in MANETs cannot reach other parties, as they do not have support from proxy servers, and cannot be reached by other nodes, as there are no SIP registrars where they can register their contact information.

III. Decentralized SIP

One solution for running SIP in ad-hoc networks could be to elect a node as the registrar in the ad-hoc network; newcomers retrieve from it the bindings of the other nodes. However, this approach presents several disadvantages in terms of fault-tolerance and scalability.

Our solution is to embed a set of the basic functionalities of a SIP proxy and registrar server in every mobile node forming a MANET. Registrar functionalities are needed to let devices use the location service, while proxy facilities are needed for logically accessing the location service. We refer to this approach as *decentralized SIP*. A major advantage of decentralized SIP is that embedded server functionalities do not prohibit the device to function also according to the baseline protocol specification. Users do not need to have two different software stacks implemented in their device and can use their SIP clients transparently in MANETs as well as in infrastructured networks.

SIP operations in MANETs can be broadly divided into two steps:

1) Discovery of users currently available in the network
2) Initiating and managing sessions with them

User discovery can be generic or targeted to find the contact address of a specific individual, e.g., a user in the contact list. Generic discoveries are needed because when a new node joins an ad-hoc network, it usually has no idea of the identities of available users. Initiating a SIP session is not possible if at least the AOR is not known in advance. The following subsections discuss two methods for discovering users' bindings in ad-hoc networks.

A. Fully Distributed SIP

Fully distributed SIP (dSIP) means that the basic operations of retrieving user contact information are performed using SIP methods, although adapted for use without centralized servers. To use dSIP, devices must first register in the ad-hoc network and communicate their presence to all the other nodes. Registering in the ad-hoc network ensures that all nodes have means to contact the newcomer. Distributed registration is done by broadcasting a SIP REGISTER message.

The term broadcast is here used in a general scope. Actually, there are three ways how the REGISTER can be propagated. Link-layer broadcast is feasible in an environment where all the nodes are connected with each other at link layer, e.g., a conference room. Alternatively, nodes can send a multicast REGISTER, addressed to the multicast address 224.0.1.75 registered for SIP. All the nodes that have registered to the multicast address receive the message. This approach requires the presence of an underlying multicast ad-hoc routing protocol, but it is not tied to any specific one. This approach is more suitable for bigger multi-hop ad-hoc networks, where nodes are not all directly connected with each other. A limit on the size of the network is given here on the necessity of efficiently handling the multicast tree. A last alternative would be selective flooding. REGISTER messages are forwarded by the nodes receiving them only if they have not been already forwarded yet and to the neighbors with which it has not been exchanged yet, and only if a hop count for maximum number of forwards has not been reached. In the rest of the discussion, broadcast is meant in the general way.

The broadcasted REGISTER is processed by the server modules of the nodes in the network. The binding of the registering user is stored and a SIP 200 OK message containing the binding of the replying user is returned to the sending node, unless local policies forbid to send it, e.g., users may not want to reply to people outside their existing contact list. This is a small variation to the SIP logic, where registrars reply to a REGISTER request returning all the contact addresses of the user *sending* the REGISTER. In fully distributed SIP, the bindings of the user *receiving* the REGISTER are returned.

The nodes receiving the REGISTER store the binding of the registering user in their cache. Entries in the cache have a limited validity time, set up either by local policies or by the *Expires* header field that can be contained in the REGISTER message. Nodes, whose registration is going to expire soon, should refresh it by sending a new broadcast REGISTER. Receivers must not reply to refreshed REGISTER requests.

When a user decides to invite a peer to a dSIP session, an INVITE message is built by the caller user agent following baseline SIP recommendations. Every request to users in an ad-hoc network is forwarded by the user agent module to the local server module.

The local proxy module receives the message, looks in the cache for the binding for the specified URI, and sends the INVITE to the contacted user and the SIP session is established. The logic of the SIP protocol has not changed; message exchange is supported by ”intermediary” entities. The main difference is that now the intermediaries are decentralized and embedded in every end device.

If the bindings for the requested target URI are not found, e.g., the binding has expired, the server broadcasts a refresh REGISTER message, specifying the AOR of the user to contact as Request URI of the REGISTER message. Only the node specified in the Request URI, if present in the network, will reply to the refresh message. A targeted REGISTER is also useful if the user suspects that a certain person should be available in the ad-hoc network, yet, no binding exists for a reason or another.

Malicious nodes can send faked registrations, e.g., after they have learned bindings of other users through broadcast REGISTERs, they send a faked refreshed REGISTER request. Thus, mutual authentication schemes between peers are needed before beginning a sensitive session. The lack of centralized trusted authorities makes mutual authentication in MANETs a complex task. A promising solution is adapting the SIP Identity Management enhancements [5]. We will not discuss security issues in this paper, but we plan to address the topic in future work.

B. SIP with Service Location Framework

The support of a service discovery framework is useful in MANETs to give users the possibility to discover people, services, or devices in the network. For example, one of the devices in the network may have Internet connectivity, and offer this service to users in the network looking for the service ”gateway”.

Service discovery can support user discovery in decentralized SIP either by finding out the bindings of users within reach in the ad-hoc network or to discover the IP address of a user by SIP AOR. This latter method is correspondent to exploiting SIP location service with the standard protocol means. We refer to this approach as sSIP.

Various service discovery frameworks can be used to support decentralized SIP. The device needs only an API so that the SIP modules and external applications can issue service discovery queries. The service discovery frameworks that could fit our architecture are (see also [3]), among others, the Service Location Protocol (SLP), Jini, Universal Plug and Play, Bluetooth Service Discovery and Salutation.

We have chosen SLP as the service discovery mecha-

nism, as it is lightweight and also specified by IETF like SIP. With SLP, the location service can be exploited by broadcasting SLP service request messages. The query is for the service SIP and contains the AOR of the user to contact as attribute filter. All devices in the ad-hoc network receive this request and the one that matches the attribute AOR returns the IP address of the service SIP on that host. When the server module receives the response it stores the IP address of the service in the cache.

SLP requests for the service SIP with the attribute filter generically set to AOR can be issued when joining the ad-hoc network. All the nodes hosting the service SIP will return the address of the service and the value of the requested attribute. These steps substitute the registration procedures used in fully distributed SIP, where bindings are received and maintained through periodic SIP REGISTER messages.

IV. Software Architecture

The software architecture of decentralized SIP is comprised of several modules, communicating with each other according to a layered structure. Fig. 1 depicts the structure and the connection between the modules. The lowest layer of the architecture is a low layer SIP library, *osip*, providing basic SIP functions such as message parsing and syntax checks. On top of the low level library, the User Agent and the Server modules are built.

The User Agent module, formed by the *eXosip* library (higher level SIP library based on *osip*), contains functions used for implementing SIP-based applications. It provides primitives and functions for exploiting higher level functionalities, such as building SIP messages. The server module implements SIP server functionalities, handling operations typical to proxy and registrar servers. It is based on the *Partysip*[1] software. These three modules form the basis of the decentralized SIP architecture; the only module that has been added and modified for enabling decentralized operations is the server (and its cache). UA and server are independent modules and communicate through UDP sockets; this design choice allows interoperability with baseline SIP clients.

The SIP server interacts with a cache, which is the logical correspondent to the SIP location service. The cache contains the bindings of the users currently present in the ad-hoc network. The way the cache is updated depends on the kind of decentralized SIP approach chosen, dSIP or sSIP.

The Session Management (SM) API, built on top of *eXosip*, hides some of the complexity of the lower levels from the application and gives the application support for operating in ad-hoc networks. As an example, the SM API provides methods for initiating, modifying and terminating SIP sessions. SIP-based applications can be built on top of the decentralized SIP modules, using the methods that the SM API makes available. Applications do not need to be aware of whether they are running in ad-hoc networks or not, as all the necessary modifications to SIP messages are carried out by the underlying SM API module.

The SM API decides whether ad-hoc or centralized functionalities must be used. In the first case, the API passes to the UA module the loopback address as the IP address of the registrar. The REGISTER is sent to the local server, which broadcasts it. If communication is performed according to standard SIP operations, the SM API passes to the UA the IP address of the pre-configured external registrar server, and the REGISTER message is sent without involving the local server module. Only the user agent side of the stack is used in this case, which ensures full compatibility with standard SIP implementations. This approach also allows a user to open SIP sessions with nodes inside and nodes outside the ad-hoc network at the same time.

The Service Discovery API allows the proxy module to use a Service Discovery framework, like SLP, for retrieving the bindings. This API can also be used by the application itself for locating other services in the ad-hoc network.

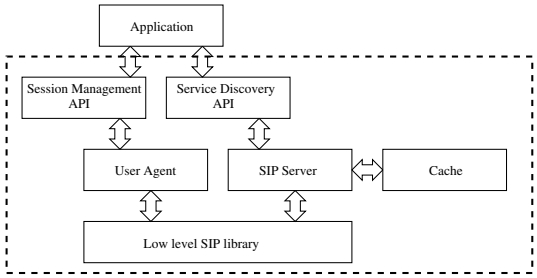

Fig. 1. Components of Decentralized SIP

V. Discussion on the implementation

Decentralized SIP has been implemented on Linux and the following three scenarios tested on laptops connected with IEEE 802.11 wireless cards, running in ad-hoc mode:

1) Session Management with dSIP,
2) Session Management with sSIP, and
3) Session Management between nodes in an ad-hoc network and nodes in an external network

For practical reasons, the tests were performed using a link-local environment, so in this section broadcasting means sending a message to all the nodes listening on the shared medium. Fig. 2 shows the messages exchanged in

[1]Documentation and source code of osip, eXosip, and Partysip server modules are retrievable at `http://www.gnu.org/software/osip/osip.html`

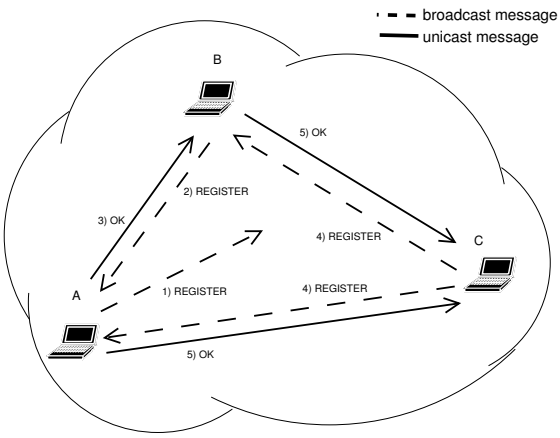

Fig. 2. User discovery with dSIP

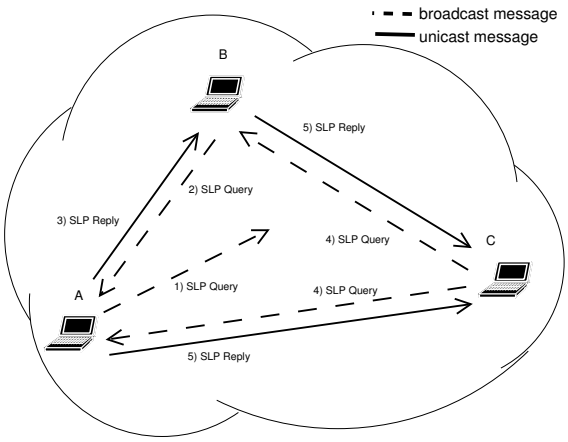

Fig. 3. User discovery with sSIP

dSIP when nodes join the ad-hoc network. Node A is the first in the ad-hoc network, it sends a broadcast REGISTER (1), but receives no reply. When node B arrives, it sends a broadcast REGISTER (2); node A receives it, stores the bindings of B and sends a 200 OK containing its own bindings (3). B knows the binding of A from the 200 OK message. Then a third node C joins. The operations and messages exchanged (4-5) are similar to when B appeared. After the exchange the three nodes know about each other.

Our implementation tries to minimize the number of messages on the air, and thus bandwidth and terminal resource consumption. Each node keeps the ID and sequence number of the received and sent broadcast REGISTERs. Refreshed registrations have the same ID but higher sequence number than the original or the previous ones; in this way nodes can discern whether the REGISTER is original or refresh, and avoid answering with a 200 OK message.

The sSIP scenario, illustrated in Fig. 3, is similar. We in fact enabled the possibility of searching for the service SIP when joining the network. The functions of broadcast REGISTER are made by the SLP request, while SLP reply returns the bindings of the available users. Every node looks for the service *SIP* and the attribute *aor*, and all nodes running that service, i.e., are SIP enabled, will return the address of the service (IP address) and, as an SLP attribute, the SIP AOR of the user.

There is a noticeable difference between the two described approaches: with standard SLP queries it is not possible to store the bindings of the querying user. This means that at the end of the exchanges in Fig. 3 node C knows the bindings of A and B; node B knows the bindings of A (but not of C) and node A knows no one. In general, users that join the ad-hoc network early cannot know the bindings of newcomers.

There are solutions for this problem: nodes may periodically send SLP queries to get also newcomers' bindings, e.g., node A in Fig. 3 would later send a refresh SLP Query and get the binding of node B and C from their replies. We refer to this way of retrieving users' bindings as *Active Service Discovery*. Alternatively, when a node receives an SLP Query from an unknown IP address, it checks whether there is a binding for that address and, if none exists, sends an unicast SLP Query back to the node from which the previous Query was received. Finally, SLP can be enhanced with *Passive Service Discovery* features. If passive service discovery is used, nodes that join the ad-hoc network can advertise their service, without external triggers, sending a broadcast message containing their bindings. Passive service discovery is currently not supported by standard SLP. Active service discovery is easier to achieve, but is more bandwidth consuming than the passive mode, as messages are broadcasted periodically.

Fig. 4 shows a different situation: the messages exchanged when establishing a session between a node in the ad-hoc network and a node in an external network, like the Internet. The node in the ad-hoc network must register its contact information to its registrar, as it would do normally, to be reachable from outside. External users do not need to know that the remote party is inside an ad-hoc network, but only the AOR. To reach other users, the node inside the ad-hoc network must forward all outgoing messages to its predefined outbound proxy server.

A necessary condition for realizing this interworking is that the ad-hoc network has external network connectivity, e.g., through a gateway node. The address of the gateway node can be retrieved with SLP, e.g. by looking for the service "gateway". Another option is for the gateway to advertise its presence in the network by using router

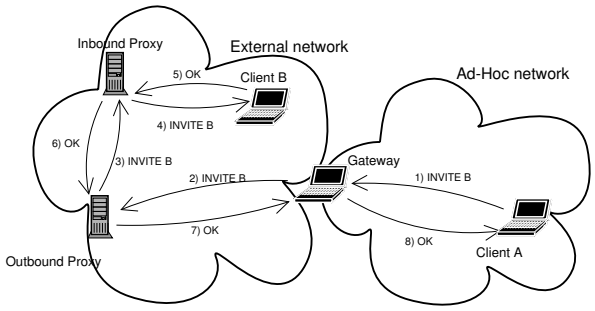

Fig. 4. Interworking with the Internet

advertisement messages [2]. If external connectivity is available, nodes in the ad-hoc network will require a globally meaningful IP address through the gateway.

In Fig. 4 the outbound proxy for client A also acts as registrar. SIP messages addressed to the outbound proxy server are routed through the gateway (1-2) and forwarded to the intended recipient following the chain of inbound SIP proxy and destination user agent (3-4). Replies are routed following the reverse route (5-8).

If no global IP addresses are available in the ad-hoc network, a Network Address Translator (NAT) based solution is needed for routing messages to the ad-hoc network from the external network. The gateway would also act as the NAT for the ad-hoc network. RFC 3581 [6] discusses extensions to SIP for NAT traversal; these can be utilized when the gateway node also operates a NAT. Scenarios where nodes act as super-peers for a subset of nodes in the ad-hoc network are not unrealistic. In a huge meeting hall, a user may provide gateway and NAT services to his company colleagues, while denying the service to non-authorized users. A general problem with NATs is that many protocols and services do not work well when NATs are involved, and often solutions for NAT traversal are protocol specific, e.g., for SIP.

VI. Summary and Future Work

This paper has presented a framework for the deployment of decentralized SIP in ad-hoc networks. Decentralized SIP is needed in network environments where no support from pre-existing infrastructures such as servers, is available. Every device that supports decentralized SIP implements limited SIP server functionalities. By distributing server functionalities among all the nodes in the network, the SIP protocol can be deployed when no centralized servers are available. Decentralized SIP devices can be used also in infrastructured networks, as the architecture is built respecting the logic of SIP operations.

Two methods for retrieving the identities and the contact IP address of users in the ad-hoc network have been discussed. In the first one, dSIP, users that join an ad-hoc network communicate their presence and contact information to other nodes in the network by sending a broadcast REGISTER message. The second method, sSIP, relies on the presence of a service discovery framework, e.g., SLP, for retrieving users' identities in the network.

The main problem with using a service discovery framework is mutual interoperability; devices implementing a framework may not be able to locate devices (and therefore the services they host) implementing a different one. An advantage of sSIP is that it is more bandwidth conservative as SLP messages are binary encoded; conversely, dSIP ensures interoperability as it relies only on SIP. Moreover, it allows mutual knowledge of the users' bindings among nodes in the network.

Work is still needed for improving and refining the architecture. An important issue to consider is how to guarantee authentication of users, when no centralized trusted identity can be used. We'll extend the scheme proposed in [5] to fit the needs of our architecture. We also aim to integrate our framework with SIP-based Instant Messaging and Presence functionalities, as specified by the IETF SIMPLE framework, and study further the issues related to ad-hoc networks connected through a gateway to an external infrastructured network.

Acknowledgments

This work has been carried out as part of the SESSI project, funded by the National Technology Agency of Finland. The authors wish to thank their project partners for their fruitful comments.

References

[1] N. Banerjee, A. Acharya, and S. K. Das. Peer-to-peer sip-based services over wireless ad hoc networks. In *BROADWIM: Broadband Wireless Multimedia Workshop*, October 2004.

[2] S. Deering Ed. ICMP router discovery messages. RFC 1256, IETF, September 1991.

[3] S. Helal. Standards For Service Discovery And Delivery. *IEEE Pervasive Computing*, 1:95–100, July-Sept 2002.

[4] H. Khlifi, A. Agarwal, and J. Gregoire. A framework to use SIP in ad-hoc networks. In *Canadian Conference on Electrical and Computer Engineering. IEEE CCECE*, volume 2, pages 985–988, May 2003.

[5] J. Peterson and C. Jennings. Enhancements for authenticated identity management in the session initiation protocol SIP. Internet draft (work in progress), Internet Engineering Task Force, February 2004. draft-ietf-sip-identity-04.

[6] J. Rosenberg and H. Schulzrinne. An extension to the session initiation protocol (SIP) for symmetric response routing. RFC 3581, IETF, August 2003.

[7] J. Rosenberg et al. SIP: Session initiation protocol. RFC 3261, IETF, June 2002.