Towards Service Interworking among Ad-Hoc Networks and the Internet

Simone Leggio* Sanna Liimatainen† Jukka Manner* Tommi Mikkonen‡ Jussi Saarinen‡ Antti Ylä-Jääski†

*University of Helsinki
Department of Computer Science
P.O.Box 68, FIN-00014 University of Helsinki
Email: leggio,jmanner@cs.helsinki.fi

†Helsinki University of Technology
TML Laboratory
P.O.Box 5400, FIN-02015 HUT
Email: sos,anttiyj@tml.hut.fi

†Tampere University of Technology
Department of Information Technology
P.O.Box 553, FIN-33101 Tampere
Email: tjm,saarin24@cs.tut.fi

***Abstract*— Ad-hoc networking is a promising technology for new services and applications. Ad-hoc networks constitute an interesting computing environment, characterized by the lack of centralized support from pre-existing network entities. Practically all research so far in the area of service and session management has focused on centralized Internet-based solutions. In this paper, we develop service discovery, session management, and the specific security measures needed to offer services in a wireless ad-hoc network. We chose SLP and SIP as our core protocols, and extend their functionality to support decentralized operation. Furthermore, we present a scheme to define the security measures needed based on services and users.**

I. INTRODUCTION

Data services are emerging into various wireless networks. In cellular environments — like the future 3G networks — centralized architectures are commonly deployed to enable service delivery and service management. This means that services are typically assumed to be handled by servers that are managed by the telecommunication operators.

There is another important parallel wireless trend that is currently happening; wireless local area networks are deployed in enterprises, in public places, and at home. Also in these networks, both network level services and service delivery are typically enabled by centralized servers. However, in addition to the centralized architectures, wireless local area ad-hoc networking will emerge when end-user devices can establish ad-hoc communities without any infrastructure. Then, peer-to-peer systems complement the traditional client-server architecture in ad-hoc network environments.

Ad-hoc networks are networks that form spontaneously between nodes in the vicinity of each other. The network topology changes as nodes join and leave the network, or move around within the network. As there are no stationary nodes, all services in the network must be provided by the participating nodes. This is a challenging task, since the availability of services and users changes with the node movement. Moreover, the ad-hoc network is typically formed using a wireless communication technology, which makes it difficult to prohibit third parties to listen to, and take part in, the communications. This, together with the dynamic network topology, makes the network more vulnerable to attacks than traditional wired networks.

In this paper, we specify and develop middleware services that enable seamless service interworking among wireless ad-hoc networks, cellular networks and the Internet. We further divide the middleware services into three subareas: session management, service discovery, and authentication and authorization mechanisms. We explore how to take advantage of various technologies in heterogeneous networks to establish ad-hoc communities among groups of people, and to enable discovery and use of services. The design of the system architecture is based on the following requirements:

- The solution must cope with the unannounced appearance and disappearance of any node at any time.
- Due to the characteristics of wireless links, the protocols must be able to cope with lost and corrupt messages, and be conservative in their use of the wireless bandwidth.
- The solution must be based on extending existing protocols and implementations.
- It must be possible to run the services in secured mode, even without access to external AAA servers.

To meet the first requirement, all services are fully distributed: all nodes are equal and carry with them the minimum functionality needed to make use of specific services in an ad-hoc network. To fulfill the second and third requirements, we compared several existing solutions and chose the most suitable ones that work in different types of networks, and whose bandwidth consumption can be tuned. To meet the last requirement, we specified an access control module that uses three security levels: none, authentication and confidentiality. All services should offer these three levels, or if a level cannot be implemented, simply deny service on that level.

This work is being carried out in the SESSI project (Seamless Service Interworking in Heterogeneous Mobile and Ad-Hoc Networks). The project is divided into two phases. In the first phase, we have developed the discussed middleware for an isolated ad-hoc network that has link-local scope. All nodes are inside the radio transmission range, and, thus, can hear each other directly, and no IP routing is needed. IP connectivity related issues, such as IP address auto-configuration and name resolution, were left out of scope since there is already a lot of work in this area (see, e.g., [1], [2], and [3]). In the second

phase which is currently underway, we broaden our designs and extend the link-local scope, e.g., to support gateways to external networks, such as the Internet. This would allow making use of new services, setting up sessions with users on the Internet, and verifying user security credentials with external AAA servers. Limiting our scope to link-local networks in the first phase was deemed justified when we were looking for concrete real-life scenarios, where ad-hoc networking could be an interesting solution. Interesting scenarios could be, e.g., meetings in a room, lectures and presentations in a lecture hall, and cafés.

We have implemented the middleware in a Linux-based environment, and evaluated the applicability of the schemes in our target environment. The three subareas of our project were implemented as independent modules, where modules can make use of the services of the other modules, e.g., securing a service discovery. The implementations showed that the initial ideas also work in practical computing environments.

This paper describes our research during the first phase of the SESSI project. In Sections 2, 3, and 4 we present our design for middleware supporting service discovery, session management, and security in a challenging ad-hoc network, respectively. In Section 5 we discuss the experiments we conducted to validate our designs. In the final section, we summarize the work, and discuss many interesting issues we plan to study during the second phase of our project.

II. Service discovery

As the overall architecture matured, we conducted an evaluation of different service discovery protocols, which could be used for searching and advertising services. An option should be offered for searching for a specific type of a service, and for services with some specific attributes. Furthermore, we wanted to be able to treat all the peers equally, implying that each peer should be able to act as a client as well as a service provider.

Several candidates, including Bluetooth Service Discovery Protocol (BT SDP), Universal Plug and Play (UPnP), and Jini were reviewed with regard to the functionality they provided and the needs of the project. The aspect that we considered to be of utmost importance is the ability to mutually authenticate both clients of services as well as servers that provide them. In addition to technical aspects of the protocols, also the availability of the protocol source code as the starting point for the experiment was considered important, together with the option that the protocol should be natural to use in an all-IP environment. In the end, we selected the Service Location Protocol (SLP) [4] as the protocol to utilize. As the practical implementation, we have relied on openSLP [5], which has been wrapped with a generic service discovery interface.

A. SLP overview

SLP is an IETF protocol intended for service lookup in the Internet. The protocol implies three different roles; User agents (UA), i.e. clients, use SLP to look for services offered by other parties, Server agents (SA) use SLP to inform its clients about available services, and Directory agents (DA) use SLP to maintain a directory of available services offered by different entities. Each service has a type, which consists of a concrete and an abstract type. The location of the service is expressed with a URL. Moreover, services' attributes can be given as attribute strings. Ordering of messages exchanged by different parties is handled with a stamp referred to as XID, which is increased by one in each new message.

By default SLP operates reactively over UDP. A user can search single services or perform a search for available service types. This operation model ties the used bandwidth to the searching frequency. The UA can send multicast or broadcast service and type request messages if no DAs are available, otherwise unicast is used to communicate with a known DA. Attributes for a certain service can be requested separately with a unicast attribute request message. SAs may also use unicast messages to register and deregister services with DAs.

In standard SLP, it is possible to authenticate the SA that originally created a URL or an attribute list of services. The SA can sign the URL and the lists of the services it advertises. When sending these elements to UAs, SA adds the corresponding signatures to the end of the message. In addition, an SLP message contains a timestamp and a security parameter index (SPI) that identifies the keying material, key length, and the algorithm that is to be used by the clients. The authentication block of SLP is illustrated in Figure 1.

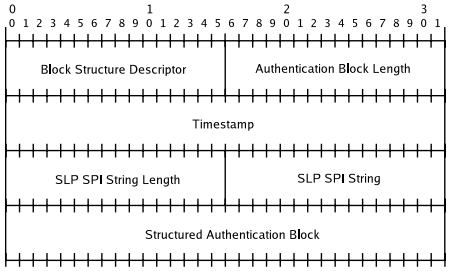

Fig. 1. SLP authentication block

However, SLP does not offer means for protecting the integrity of the most common messages. Therefore, it is possible, e.g., to reuse signed information in faked messages.

B. Modifications needed for Ad-Hoc environment

Since our target environment is a peer-to-peer network, the architecture of SLP must be reconsidered. First, no generic DA can be used in the system, as this would result in increased network traffic and distribution of deprecated service information. Instead, services are always discovered or advertised by peers themselves. Second, in order to keep all peers similar, all of them include ad-hoc capable UA and SA.

Due to the similarity discussed above, it is not enough to be able to authenticate SAs services, but the client of the service may also be authenticated, thus enabling more secure services. This has led us to consider some modifications to core SLP. At least the following security-related issues must be be tackled: no means to connect a service to certain security properties, signatures used for integrity protection are not

sufficient against tampering, no authentication of UA to SA, timestamping relies on clock synchronization, and no secured means to bind messages to a certain sequence (peers may pick whatever XID they wish).

We tackled these problems as follows. To establish the connection between a single service and the required security properties, we used the abstract part of the original SLP service type as a security identifier. In consequence the service type requests were discarded because they could be used to obtain information from the various services in the ad-hoc network. The problems related to integrity and authentication were solved by replacing the structural authentication block with a generic block that can be used in all SLP messages. First, the SPI is replaced by sender's and group's identifiers. Second, the original authentication block was replaced by a signature calculated over the entire message including the receiver's ID. The timestamp problem was solved by replacing the real-time stamping scheme with logical stamps, where each node caches a number of messages from different peers and uses their logical stamps to determine whether a received message is new or a replicated one. The secure binding of messages to sessions, was achieved by adding the receiver's identifier to each message addressed to a particular peer. This, together with XID and the support for message integrity, results in a straightforward yet improved way for clients to manage sessions. The revised authentication block is illustrated in Figure 2.

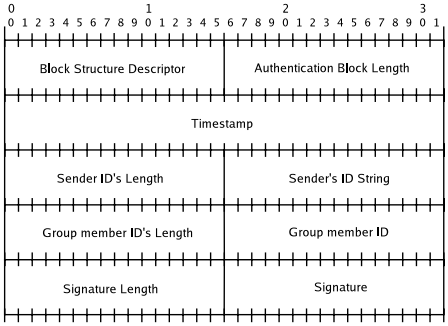

Fig. 2. Improved SLP authentication block

III. Decentralized SIP

The baseline SIP protocol [6] cannot be deployed in ad-hoc networks because it strongly relies on centralized entities, namely the SIP servers. SIP servers are needed for registering users in the SIP network, and for locating the exact address where a user can be contacted. This architecture is unfeasible in ad-hoc networks, which are dynamic networks formed by peers and where no support from pre-existing, centralized entities is available. In order to enable SIP in ad-hoc networks, we decided to embed a small subset of SIP proxy and registrar server functionalities in all SIP end devices. This allows performing in a decentralized and distributed fashion. We refer to this solution as decentralized SIP (dSIP).

A. Overview of dSIP Operations

Decentralized operations mainly affect the way how SIP users are discovered in the network. In ad-hoc networks, users must first discover who is present in the vicinity (user names, in SIP, called Address of Record, AOR), and the IP address of the mobile node the user is available from. After this information, referred to as binding, has been retrieved, SIP sessions can be initiated and managed.

dSIP can operate in two modes, proactive and reactive. In proactive mode, when a node enters the ad-hoc network, it broadcasts a SIP REGISTER message containing the user's binding. The nodes that receive the REGISTER message store the binding of the registering user and reply with an unicast SIP 200 OK message addressed to the registering user's IP address. The message contains the binding for the replying user. After this handshake, in absence of losses, all the nodes in the network know the bindings of each other. The scheme uses refresh messages and timeouts to update the status information of users, and to counter the problems created by lost messages.

In the reactive mode, a node broadcasts the REGISTER message only when the user wishes to initiate a session. The replies to the message are used to build a list of users available in the ad-hoc network. The amount of bandwidth used for this signaling is tied to the size of the ad-hoc network. The overhead of proactive dSIP is further affected by the arrival rate of new nodes, while reactive dSIP is affected by the frequency of session initiations.

An alternative approach to proactive dSIP uses SLP instead of SIP for registering SIP users. A newcomer node sends a broadcast SLP query for all the users running the service "SIP" and for the service attribute "AOR". The nodes reply with an SLP message containing the IP address of the answering node, i.e., where the service is available, and the value of the requested attribute, i.e., the user name. Subsequently, the querying user gets to know the bindings of the other users in the ad-hoc network. However, in this scheme, the binding of the querying node cannot be stored in the replying nodes, as in the dSIP scheme. Thus, each node must periodically send a broadcast SLP query in order to update the list of users available in the ad-hoc network.

Once the server has stored in its cache the bindings of the other users in the network, a list of available users can be communicated to the application, and SIP sessions can be initiated according to baseline SIP operations. The user agent forwards the INVITE message to its outbound proxy server, which in our scheme is located within the device itself. This server, looks up from its internal bindings cache the IP address for the AOR specified in the target of the INVITE, and forwards the message directly to the intended recipient.

B. Software Modules for dSIP

Figure 3 shows the software modules of the implementation, and how they interact with the other modules of the service framework. Our implementation is based on the *oSIP* library and the partysip server [7]. The low level SIP library performs basic SIP-related operations, like message parsing or syntax

checks; it is utilized by the two upper modules. The user agent (UA) module carries out the baseline SIP end device functionalities. The server module enables the deployment of SIP in ad-hoc networks, and acts as the predefined outbound proxy server and registrar for the user agent.

The Session Management (SM) API decides when ad-hoc functionalities must be used. The operation mode defines the target IP address of the register message. If a registration to a standard SIP server must be done, the API passes the infrastructure network registrar IP address to the UA module. To register to the ad-hoc network, the API passes the loopback address to the UA module. Thus, the REGISTER message is sent within the node to the local SIP proxy, which can then broadcast the message to the ad-hoc network. This scheme allows independence between UA and the local SIP server; they are unaware of the fact that they reside in the same device. The scheme allows using a dSIP device according to standard SIP operations, as the server module is an independent addition to the SIP stack of a user agent.

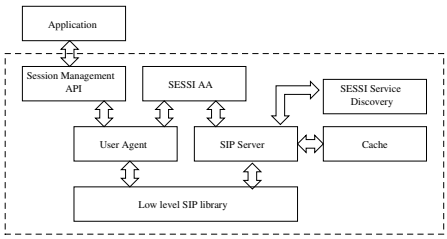

Fig. 3. Software modules for decentralized SIP

The server module can access the Service Discovery service, in order to retrieve the SIP users in the ad-hoc network. When secure SIP operations are needed, both user agent and server interact with the AA module. Mobile ad-hoc networks may be formed by devices with low computing capacities; for such a reason, embedded server functionalities are limited to the most basic possible set, and the SIP server used as basis was lightweight and modular, so only the desired features were added to the implementation.

IV. Security

The SESSI Authentication and Authorization (AA) Module provides credential repository and access control for service location and session management, and generic services such as applications provided by the nodes of the network. Each device in the SESSI enabled network has its own AA module that is independent from the used security mechanisms.

A. Authentication and Authorization Credentials

Unlike the Bluetooth authentication, the SESSI AA module authenticates or authorizes users, not just devices. Each user has a base key pair stored in the AA module. The key pair can be any asymmetric cryptographic key pair that is authenticated in a trustworthy way. For example, the base keys can be fetched beforehand from an external server or they can be added manually by the user that has personally authenticated the key pair and its owner.

In addition to the base keys, other keys and user names for different kind of services, e.g., security parameter index (SPI) and a key for the service discovery, can also be stored in the database either beforehand or automatically in the first contact after the authentication. These service keys are linked to the base key of the user to whom they belong. Thus, the keys form a hierarchical structure where the root key identifies a user and leaf keys are used for different services. This hierarchic structure makes trust management easier for the user.

B. Security Levels and Access Control

The SESSI AA module offers three levels for security: none, authorization, and confidentiality. The first level does not provide any security, and can be used, e.g., for transferring public information from a device to another device. In the authorization security level, the other user is authenticated or authorized. This means that the user is either identified or, if the identification information is not necessary, her right to use the service is verified. In the confidentiality level, all the communication is protected against eavesdropping and the communicating parties are authenticated or authorized.

The SESSI AA module provides two-way access control with three levels of security. Because devices in an ad-hoc network provide services to each other, it is not enough to only define which users can use a certain service. A cautious user may also define which devices can provide the service for her to use. First, like in any traditional access control mechanism, a user can define an access control policy for her device: for each service or resource offered, the user defines who can use the service and the security level access is allowed. The SESSI AA module provides access control also for the other direction: the user defines the services that can be offered to her device, and the security levels required for these services. In addition to the fine grained access policy, the user can define a network specific security level that overrules the service policies if it requires higher security than the service policies. For example, when the user does not trust all parties in a specific environment, she can adjust her security policy by just raising the network specific security level.

C. Structure of the SESSI AA Module

The SESSI AA Module consists of five internal parts as is shown in Figure 4. The credentials are stored in a SQL database. On top of the database, AA manager handles the access control lists and provides functions to manage user credentials and actions like signing, verifying and encrypting the messages. A separate Crypto Service Module provides the cryptographic functions used in the SESSI AA Module. It is based on OpenSSL library. The service location and session management modules, as well as, other SESSI-enabled services and applications, can use the AA module through an API. For the user, the SESSI AA module provides a graphical user interface that can be used to add new credentials or modifying the existing ones.

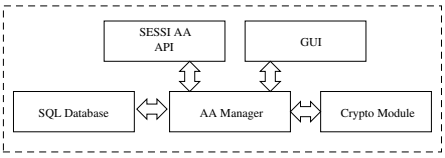

Fig. 4. Structure of the SESSI AA Module

The SESSI AA module helps a user to manage the access control policy of his or her device in an ad-hoc network. It provides one centralized place for all security management. This supports users in the demanding tasks of trust and security management and still allow the user to be aware of her choices. We have implemented and tested a prototype of the AA module for the SIP and SLP implementations.

V. Prototype Implementations

The presented schemes were implemented and analyzed on our own ad-hoc network. We used several Linux-based laptop computers connected to each other with 802.11b WLAN in ad-hoc mode. In this first phase, we studied our distributed SLP and dSIP independently, but coupled our SLP implementation with the security management module.

Each laptop was running a chat application that used the SD module to find nodes in the network and initiate a TCP connection with each of them. The nodes were able to dynamically enter and leave the ad-hoc network. Service discovery for SIP was also tested and proven functional. It offers an alternative operation model where SIP is using the SD module to retrieve SIP user names and then initiates the sessions on its own. The security enhancements have been tested with the chat application. On authentication level only verified users could join the chat and on confidentiality level the exchanged service discovery messages were also encrypted. In addition the logical timestamps prevented replay attacks efficiently.

The dSIP scheme was also evaluated and fulfilled the expectations well. Sessions could be established among users in the ad-hoc network, and complete mutual registrations were archived, in reactive and proactive mode. Moreover, we made a test of an interworking situation, where sessions were established between a user inside the ad-hoc network, and one in an external network. A gateway node provides access to the external world from the ad-hoc network. The internal node used a server in the external network as outbound proxy, and the internal server module was not utilized. The test showed the interoperability of the decentralized SIP stack with the baseline SIP protocol stack. More advanced studies will be performed during the second phase of the project.

Our analysis of use-cases for WLAN ad-hoc networks focused on multiparty communication and groups, and indicated that the groups will probably be some tens of people. The link-local WLAN is suitable for this type of networking. The performance of the distributed SLP and SIP implementations and the AA module were feasible in our tests with respect to delay, latency and control overhead. Our tests included less than 10 nodes, though, depending on the wireless link technology, the system can scale up to hundreds of users.

VI. Conclusions and Future Work

This paper presented the first phase of the SESSI project. The target was to study secure and decentralized service and session management in ad-hoc networks. Our design is based on extensions to SLP and SIP. In addition, the SESSI AA module implements security. The designs were validated with prototypes, and the experiments proved that our schemes were able to fulfill the requirements and goals set for the project—services and users were able to look for each other, and initiate sessions and services. The presented schemes can be employed both in wireless and wired small-scale IP-networks.

We have identified a number of very important and interesting issues to study in more depth during the second phase of our project. A topic of future study is extending the peer-to-peer ad-hoc network to the Internet by, e.g., a peer that opens a connection to the Internet over a 3G service provider, or through a WLAN access point. The peer could then advertise services offered by the fixed network, thus resembling the function of a Directory Agent (DA). However, the required implementation is different from conventional DAs, which are normally used as the only means to find certain services, whereas in our setting, some of the services are known by DAs and some others only by the parties offering them. Therefore, we plan to implement a peer-scaled DA, which can be included in any (or all) peers. The Internet connectivity also allows initiating sessions between users in the Internet and in the ad-hoc network. The challenge is in automatic configuration of the ad-hoc nodes to make use of this connectivity. We also plan to extend the operation of dSIP with the IETF Instant Messaging and Presence framework [8]. Another important issue would be the ability to run SLP and dSIP in a multi-hop environment. Finally, adding support for retrieving external security credentials stored in trusted third parties, such as network operators, to the AA module will make our architecture complete.

Acknowledgments

This work is part of the SESSI project (2644/31/03), funded by the National Technology Agency of Finland, Elisa Corporation, the Finnet Group, and Nokia Corporation.

References

[1] Thomson, S., and Narten, T., IPv6 Stateless Address Autoconfiguration. IETF, RFC 2462, December 1998.
[2] Ceshire, S., Aboda, B., and Guttman, E., Dynamic configuration of link-local IPv4 addresses. Internet draft (work in progress), July 2004.
[3] Jeong, J. P., et al., IPv6 DNS discovery based on router advertisements. Internet draft (work in progress), February 2004.
[4] Guttman, E., Perkins, C., Veizades, J. and Day, M., Service Location Protocol, Version 2, IETF, RFC 2608, June 1999.
[5] OpenSLP web site at `http://www.openslp.org`.
[6] Rosenberg et al., J.,SIP: Session Initiation Protocol, IETF, RFC 3261, June 2002.
[7] The GNU oSIP library at `http://www.gnu.org/software/osip/osip.html`.
[8] Rosenberg, J., A Presence Event Package for the Session Initiation Protocol, IETF, RFC 3856, August 2004.