Implementation of the Diameter-based Cx Interface in the IP Multimedia Subsystem

[1]Siniša Tomac, [1]Marko Sikirica, [2]Lea Skorin-Kapov, [1]Maja Matijašević
[1]University of Zagreb, FER, Unska 3, HR-10000 Zagreb, Croatia
[2]Ericsson Nikola Tesla, R&D Center, Krapinska 45, HR-10000 Zagreb, Croatia
Tel. +385 1 612-9757, Fax +385 1 612-9832
E-mail: {sinisa.tomac|marko.sikirica|maja.matijasevic}@fer.hr, lea.skorin-kapov@ericsson.com

The Diameter protocol was initially developed by the Internet Engineering Task Force (IETF) as an Authentication, Authorization, and Accounting (AAA) framework intended for applications such as remote network access and IP mobility. Diameter was further embraced by the Third Generation Partnership Project (3GPP) as the key protocol for AAA and mobility management in 3G networks. The paper discusses the use of Diameter in the scope of the IP Multimedia Subsystem (IMS) as specified by 3GPP, with special emphasis on its use on the Cx interface between the Call Session Control Function (CSCF) and the Home Subscriber Server (HSS). The goal of this work was to implement basic Diameter functionality corresponding to the Cx interface. The paper compares a number of open source implementations of the Diameter Base Protocol, and provides the rationale for choosing the Open Diameter solution for implementation purposes. Experiences regarding installation, configuration and implementation of basic authorization functionality using Open Diameter are discussed. The resulting implementation is verified in a laboratory testbed.

I. INTRODUCTION

Evolution of the 3rd generation network architecture is driven, among other factors, by the requirement to provide a rather fast, flexible and cost-efficient way of introducing new services for operators, as well as third-party service and content providers. The IP Multimedia Subsystem (IMS), as specified by the 3rd Generation Partnership project (3GPP), represents the key element for supporting ubiquitous service access to multimedia Internet services, with adequate support for Quality of Service as well as advanced, service-differentiated charging [1]. Initially specified by 3GPP/3GPP2, the IMS standards are now being adopted by other standards bodies including ETSI/TISPAN. For the purposes of Authentication, Authorization, and Accounting (AAA) and mobility management in 3G networks, 3GPP has adopted the Diameter protocol [2], developed by the Internet Engineering Task Force (IETF). This paper discusses the use of Diameter within the scope of the IMS.

The paper is organized as follows. Section II briefly describes IMS, its functions and interfaces, and the role of the Diameter protocol as applied to the Cx interface. Section III provides an overview of publicly available open source implementations of the Diameter protocol, while Section IV describes the selected implementation, Open Diameter, in more detail. Section V describes our implementation of basic Diameter functionality corresponding to the Cx interface, as an extension to the existing Open Diameter implementation. Section VI concludes the paper.

II. ROLE OF DIAMETER IN IMS

The IMS is based on a horizontally layered architecture, consisting of three layers, namely, Service Layer, Control Layer, and Connectivity Layer. Service Layer comprises application and content servers to execute value-added services for the user. Control layer comprises network control servers for managing call or session set-up, modification and release. The most important of these is the Call Session Control Function (CSCF). Connectivity Layer comprises routers and switches, for both the backbone and the access network.

A. IMS functions

A somewhat simplified IMS architecture is shown in Figure 1. As mentioned earlier, one of the key functions in the control layer is the CSCF. In this paper, we focus on the interface between the Home Subscriber Server (HSS) and the CSCF. The HSS serves as the main data storage for user related information, such as IMS user profiles (including location), security and registration information, access parameters, and application server profiles.

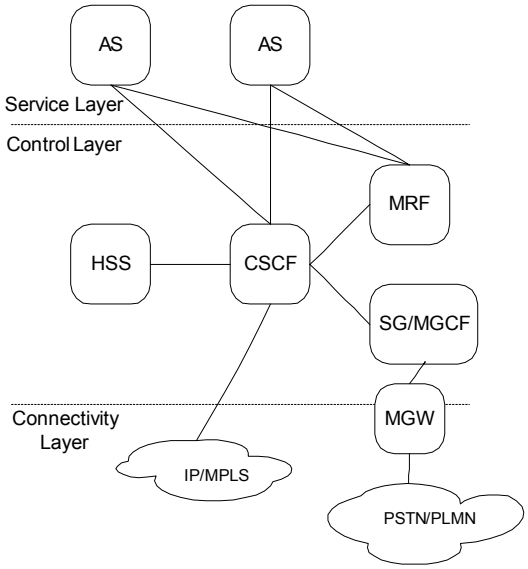

Figure 1. The IMS architecture

The CSCF may serve three different purposes, as the Proxy CSCF (P-CSCF), the Interrogating CSCF (I-CSCF), and the Serving CSCF (S-CSCF).

The P-CSCF is a Session Initiation Protocol (SIP) proxy that acts as the first contact point between the IMS terminal and the IMS network. It is assigned to an IMS terminal during IMS registration. The I-CSCF is also a SIP

proxy, usually located in the home network, at the edge of the administrative domain. Main functions of the I-CSCF are to contact HSS in order to obtain the name of the S-CSCF that is serving the user, and to assign the S-CSCF to the user based on received information received from the HSS.

The S-CSCF is the central node of the signaling plane, the "brain" of the IMS. The S-CSCF is located in the home network and it uses the Diameter-based Cx and Dx interfaces (reference points) towards the HSS to download and upload the user profiles.

B. The Cx reference point

As per IMS technical specifications [3][4], the Cx reference point is located between the S-CSCF/I-CSCF and the HSS, as shown in Figure 2. The Subscription Location Function (SLF) is required in a network in which there is more than one HSS; it provides the mapping between a particular user address and its corresponding HSS. As already noted, the protocol used at the Cx reference point is Diameter. (The unmarked interface between the S-CSCF/I-CSCF and the SLF is Dx, which also uses Diameter.)

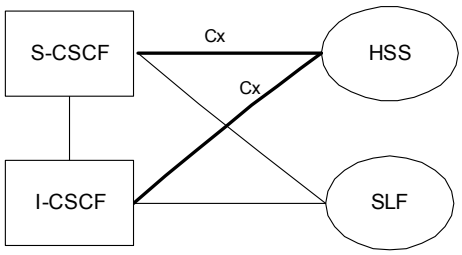

Figure 2. The Cx interface

Procedures in the Cx reference point may be grouped into three areas:

1. Location management procedures
2. User-data handling procedures
3. Authentication procedures

Each group of procedures is briefly described next.

Location management procedures. In location management procedures, the User-Authorization-Request (UAR) command is sent to the HSS whenever the I-CSCF receives a SIP REGISTER request from the P-CSCF. The UAR command contains private and public user identity, visited network identifier, routing information, and type of authorization. In response to the UAR command, the HSS responds with the User-Authorization-Answer (UAA) command. The UAA command contains the name of the S-CSCF assigned to the user. After authorization, the I-CSCF finds an S-CSCF that will serve the user, and it forwards the SIP REGISTER request to the S-CSCF. Once the S-CSCF receives the SIP REGISTER request, it uses the Server-Assignment-Request (SAR) command to communicate with the HSS, and it informs the HSS which S-CSCF will be serving the user. The HSS responds with the Server-Assignment-Answer (SAA) command, which contains the user profile and charging information. Later, when the HSS wants to initiate de-registration it uses the Registration-Termination-Request (RTR) command, stating the reason for de-registration. The RTR command is acknowledged by a Registration-Termination-Answer (RTA) command. If an I-CSCF receives any SIP method other than REGISTER, a procedure for finding S-CSCF uses the Location-Info-Request (LIR) command containing public user identity and routing information. The HSS responds to LIR with Location-Info-Answer (LIA) command, containing the name of the S-CSCF.

User-data handling procedures. During the registration process, user and service-related data are downloaded from the HSS to the S-CSCF via the Cx reference point by using SAR and SAA commands. It is possible, however, for this data to be changed later, during the time while the S-CSCF is still serving the user. To update the data in the S-CSCF, the HSS sends a Push-Profile-Request (PPR) command with private user identity, routing information, and user data. The response to the PPR command is Push-Profile-Answer (PPA) command.

Authentication procedures. In the IMS, authentication relies on a pre-configured shared secret and a sequence number stored within the IP Multimedia Services Identity Module (ISIM) in the User Equipment (UE) as well as in the HSS in the network. To authenticate the user, the S-CSCF sends a Multimedia-Auth-Request (MAR) command to the HSS. MAR contains the private and the public user identities, S-CSCF name, routing information, number of authentication items, and authentication data. The HSS responds to the MAR command with the Multimedia-Auth-Answer (MAA).

C. Diameter Protocol

Diameter is an authentication, authorization and accounting (AAA) protocol developed by the Internet Engineering Task Force (IETF). It is based on an earlier IETF's AAA protocol called RADIUS (Remote Authentication Dial-In User Service), widely used for dial-up PPP (Point-to-Point Protocol) and terminal server access. Extending the functionality of RADIUS, Diameter is designed to provide AAA services for a range of access technologies, including wireless and Mobile IP. The Diameter specifications consist of the Diameter Base Protocol [2], Transport Profile, and applications such as Mobile IPv4, network access server, credit-control, and Extensible Authentication Protocol (EAP).

The Diameter Base protocol is utilized for negotiating capabilities, delivering Diameter data units, handling errors, and providing for extensibility. On the other hand, the Diameter application defines application-specific functions and data units. Diameter is an application layer protocol. Transport protocols to carry Diameter messages include Transmission Control Protocol (TCP) and Stream Control Transmission Protocol (SCTP). For securing the connection, Internet Protocol Security (IPSec) and Transport Layer Security (TLS) are applied.

Diameter is a peer-to-peer protocol, meaning that any Diameter *node* may initiate a request. The three types of nodes are *clients*, *servers*, and *agents*. Clients are generally edge devices of a network which perform access control. A Diameter agent provides relay, proxy, redirect, and

translation services, while Diameter server handles the AAA requests for a particular domain, or realm. Message routing is based on the network access identifier of a particular user.

As to data structure, in each Diameter node there is a peer table, which contains a list of known peers and their corresponding properties. Each peer table entry is associated with an identity and can be either statically or dynamically assigned. It includes a relative priority setting, which specifies the role of the peer as primary, secondary, or alternative. The status of the peer relates to a specific configuration of the finite state machine of the peer connection, called the Diameter Peer State Machine. As a part of message-routing process, Diameter realm-routing table references the Diameter peer entries. All realm-based routing lookups are performed against a realm-routing table. The realm-routing table lists the supported realms, with each route entry containing certain routing information. Each route entry is either statically or dynamically discovered. Dynamic entries are associated with an expiry time and also route entry is associated with an application identifier, which enables route entries to have a different destination depending on the Diameter application. In a Diameter peer table the destination of a route entry corresponds to one or more peer entries.

A Diameter message consists of a Diameter header, followed by a certain number of Diameter attribute-value pairs (AVPs). The Diameter header is composed of fields denoting Command Flags, Command Code, and Application ID. The Command Code denotes the command associated with the message, while the Application ID identifies the application to which the message is applicable. AVPs define the method of encapsulating information relevant to the Diameter message.

III. DIAMETER PROTOCOL IMPLEMENTATIONS

Table I lists four publicly available, open source implementations of the Diameter Base protocol which we reviewed and considered for possible implementation.

TABLE I. Diameter Base Protocol implementations

Name	DISC	Open Diameter	WIRE Diameter	Diameter Charging SDK
Programming language	C	C++	C++	Java
Source code availability	yes (GPL)	yes (GPL)	yes (GPL)	only partially (client yes, emulator no)
Platform	Linux/ FreeBSD	Cross-Platform	Cross-Platform	Platform Independent
Diameter Base Protocol support	yes; full	yes; full	yes; partial (8 out of 10 functions)	vendor-specific extensions

We now briefly describe each implementation and note the features based on which we selected Open Diameter as a basis for our experimental implementation of Cx interface functionality.

A. Diameter Server Client

Diameter Server Client (DISC) is an open source AAA Diameter implementation, developed by the DISC project, (http://developer.berlios.de/projects/disc/). It can be configured to act as either a Diameter Server or a Diameter Client. On the project's Web page, the authors state that DISC enables what they call "a plug-in model" for new applications, meaning that third parties can link their plug-in with the server or the client code and thus provide various services. DISC is written in programming language C and it has been designed for Linux/FreeBSD platform. Since we needed a platform-independent solution, DISC was not applicable for our purposes. Should an attempt to port DISC onto other platforms, such as MS Windows, be made, it would require significant changes in the transport part of DISC.

B. Open Diameter

Open Diameter (OD) is an open source implementation of Diameter Base Protocol developed by the Sourceforge community (http://sourceforge.net/projects/diameter). It is written in C++, and it is platform independent. Supported platforms include Linux, FreeBSD, and MS Windows 2000/XP. OD supports both Internet protocols IPv4 and IPv6.

Open Diameter has all Diameter Base Protocol functions implemented and the source code is available under GNU General Public License (GPL). Some documentation is also available. Functionality of the Diameter protocol is provided to other applications through dynamic-link library files (*.dll files on MS Windows platform).

C. WIRE Diameter

The WIRE Diameter is an open source implementation of the Diameter Base Protocol and Diameter EAP Application, developed by the Wireless Internet Research & Engineering (WIRE) Laboratory at the NTHU Taiwan (http://wire.cs.nthu.edu.tw/WIREDiameter). The software is in part based on OD, but the source code has been modified (support for two functions of Diameter Base Protocol is missing) and it is differently organized.

The WIRE Diameter provides various authentication schemes, including EAP-MD5, EAP-TLS, EAP-TTLS, and PEAP. WIRE Diameter is written in C++, and it is platform independent. Supported platforms include Linux, FreeBSD, and various versions of MS Windows.

D. Diameter Charging SDK

The Diameter Charging Software Development Kit (SDK) is developed by Ericsson, and it intended to support client applications. It is written in Java, and it is platform independent. The software is made available through the

Ericsson Web site at http://www.ericsson.com/mobilityworld/sub/open/technologies/charging_solutions/tools/diameter_charging_sdk. It includes the Diameter Charging API, Diameter Charging Emulator (which emulates the charging server), Diameter Charging Client, and documentation. Due to its intended use for client applications, only the Diameter Charging Client source code is available, and the rest is provided in form of Java class files. The package uses a vendor-specific Service Charging Application Protocol (SCAP), which is based on Diameter Base Protocol.

The Diameter Charging API isolates the core protocol implementation and allows the application to use the Diameter interface with operations that are relevant to the application. The Diameter Charging Client is a reference application that uses the Diameter Charging API. It is used for setting up connections to Diameter server. By using the client, it is possible to insert data, send requests to and receive responses from the (emulated) charging server.

Having considered the implementations listed above, we decided to base our implementation on OD, because it was an open source solution, fairly well documented, and it was under active development and discussion by the community. OD is now described in more detail.

IV. OPEN DIAMETER

This section describes the software architecture of OD. Parts of this text have been taken verbatim from OD documentation.

The Open Diameter API is a session based API, in which each type of Diameter session is being represented by a C++ class. Each session class is derived from a specific AAA state machine framework as defined in Section 8 of RFC 3588 [2]. Session classes handle message transmission, message processing, and event handling. Applications can implement their own AAA functionality by using the appropriate session classes. Figure 3, taken from OD documentation and somewhat simplified, shows the architecture of the OD framework.

In general, session classes may be either client classes or server classes, providing AAA capabilities for clients and servers, respectively. Classes may also be further divided into authentication/authorization classes and accounting classes.

The main difference between client and server classes is in the way they are instantiated. For application classes based on *client sessions*, it is the responsibility of the AAA client application to create and manage the instances of these sessions. For application classes based on *server sessions*, the library is responsible in creating and deleting instances of these classes. Server classes are deleted by using an internal garbage collector, once a server session has completed its execution as defined by its state machine. To facilitate the instantiation of application derived server session classes, the library provides a server session factory that an application may instantiate and register.

Once properly registered, these session factories will create AAA session objects every time a new authorization and/or accounting request arrive. The only criterion for this action is whether the local AAA application supports the application ID advertised in the initial request message.

It should be noted that both client and server session classes only provide Diameter session management. Diameter peer connectivity management is provided within another class called the application class.

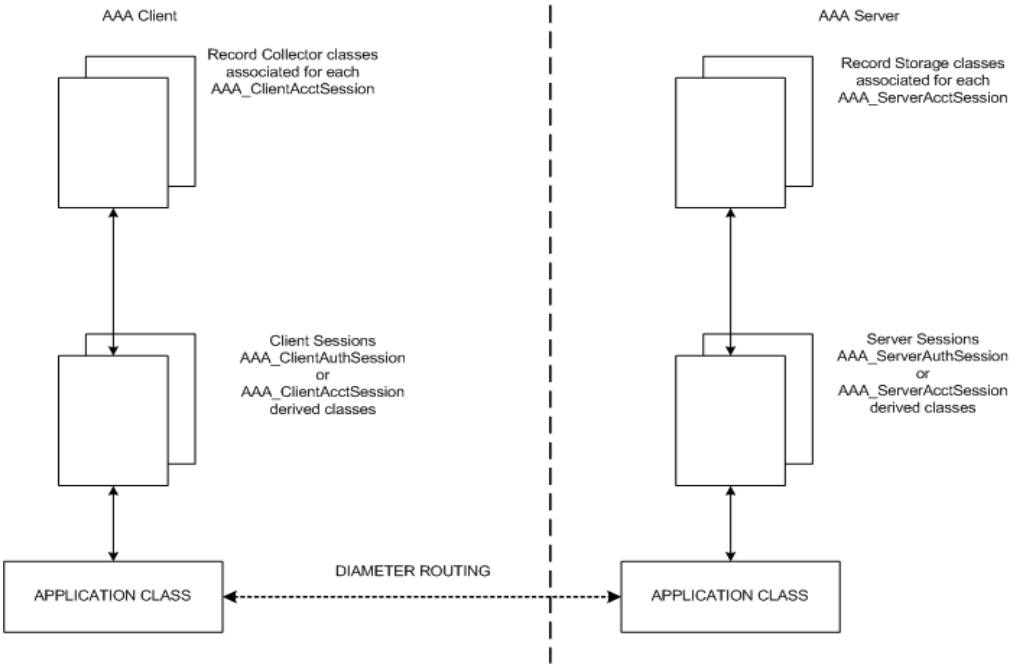

Figure 3. Open Diameter software architecture

This class manages configuration loading, peer connectivity, and AAA message routing. Client session class binds to this application class via its constructor. Server classes are bounded to an application class via the server session factory class which is registered in the application class. By binding to the application class, session classes are able to send and receive messages from the routing platform provided by the application class.

V. IMPLEMENTATION & RESULTS

A. Setting up the Open Diameter

The OD Base packages are available as either "plain source" files, or as source files organized into Microsoft Visual Studio solution (provided by Toshiba research). We used the latter. To compile and use OD libraries support of following API-s and applications is required:

1.) Perl (Active Perl 5.8.7 Build 813)

The Perl language is utilized by some installation scripts. The version we used is available from:
http://www.activestate.com/ActivePerl.

2.) Xerces C++ XML Parser
(xerces-c_2_6_0-windows_nt-msvc_60)

Xerces is a shared library for parsing, generating, manipulating, and validating XML documents. The version we used is available from:
http://xml.apache.org/xerces c/index.html.

3.) OpenSSL library (openssl-0.9.8.tar.gz)

OpenSSL is an open source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and TLS v1 protocols, as well as a full-strength general purpose cryptography library. It is available from http://www.openssl.org.

4.) ACE library (ACE-5.4)

ACE library is an open-source, object-oriented toolkit written in C++ that implements core concurrency and networking patterns for communication software, including event demultiplexing and event handler dispatching, signal handling, service initialization, interprocess communication, shared memory management, message routing, dynamic (re)configuration of distributed services, concurrent execution and synchronization. It is available from:
http://www.cs.wustl.edu/%7Eschmidt/ACE.html.

B. Our implementation

The OD distribution contains several client/server examples, which we used to examine Diameter mechanisms. As a starting point in our development we used the example presenting an authorization application. In terms of specifications, we followed the specifications of the Diameter protocol [2][5] and Cx interface [3][4] provided by 3GPP. Our work included modifying the client and the server classes provided by OD by adding the Cx interface specific Diameter messages – UAR, MAR, and SAR – and building the client and server applications to use the functionality of those classes.

The client and the server code have rather similar structures, up to the point of Diameter session management. The example code included in the OD distribution provided sample Diameter communication between the client and the server and we needed to implement the UAR, MAR, and SAR commands, which, according to the Cx specification, are sent from the client (CSCF) towards the server (HSS).

The OD distribution contains both the server and the client classes to enable a Cx node to operate in a peer-to-peer network. In our application, we implemented the functionality of the Cx interface as if the CSCF acted as a client and the HSS acted as a server. (This could have also been implemented the other way round to have both client/server, i.e. peer functionality on each side.) Figure 4 shows the exchange of messages in our implementation. It may be noted that each message transmission method (i.e. TxUAR) on the client side has its corresponding counterpart on the receiving, server side (i.e. RxUAR). The notation used here is Tx for transmission, and Rx for receiving. Messages are distinguished by their message code, embedded in the message header. The client composes a message with the specific code, and sends it to the server, which then recognizes the message code and initiates the appropriate receiving method.

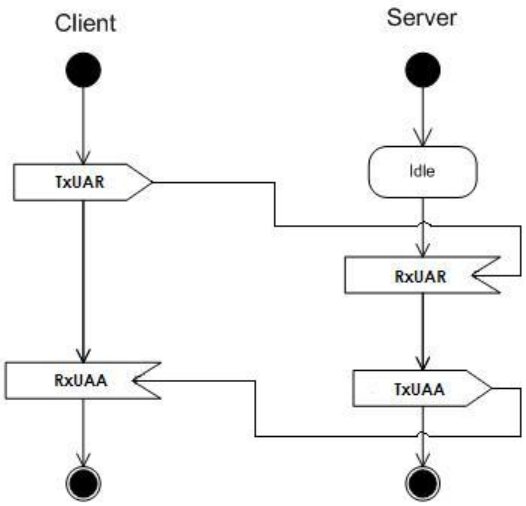

Figure 4. Exchange of Cx specific Diameter messages

Each message type carries some specific information, being coded as AVPs. Thus, it was necessary to implement the method for composing and resolving the message for all types of messages. This included definition of message parts, initialization of message fields, and finally, construction of message body.

Finally, we also extended the OD dictionary (XML file), which is used by the parser part of OD for message identification and validation, with Cx related message type specifications.

The exchange of messages between the client and the server goes as follows. The client issues a request, and then waits. The server, having received the request, parses the message, processes the request and parameters, and, invokes the corresponding response message method, which returns the information required back to the client. Having completed this task, the method returns a status code, which may be used to determine the success of actions performed and to set the application into the adequate state.

C. Testing

We first tested the initial OD implementation and its conformance to the Diameter specification, followed by testing of our extended implementation. Since a more recent release of OD became available in the course of our work, our final implementation was based on OD version 1.0.7-g, which was significantly improved compared to the previous one.

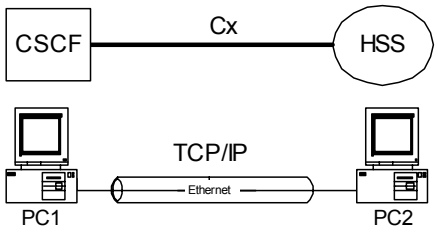

Figure 5. Laboratory setup

The laboratory setup (Figure 5) consisted of two low-end PC-compatible computers, PC1 and PC2, attached to the local TCP/IP network via Ethernet interface. The PC1 served as the CSCF node (running the Diameter client application), and PC2 served as the HSS node (running the Diameter server application). We also needed to configure the applications by editing the XML configuration files which contain network, routing, identity, and some additional configuration data.

We used Ethereal (www.ethereal.com, version 0.10.12) network protocol analyzer to capture the Diameter messages exchanged between the client and server applications. Ethereal was installed on the both the client and the server. It may be noted that the version 0.10.12 of Ethereal worked well, while many problems were encountered in the previous version of the package, regarding proper recognition and dissection of Diameter protocol messages.

The purpose of the test was to establish the client and the server behavior, and study the content of messages exchanged. We activated the packet capturing procedure within Ethereal and then initialized the server and the client applications. Once both applications were properly started, we commenced the message exchange between them. Messages captured confirmed the correct operation on both the client and the server side, but with one notable exception in OD part of the code. Namely, there was no notification of session termination on the server side, which could pose a threat to normal operation in real environment. The OD documentation states that all server side sessions are being properly terminated by the garbage collector. Should this implementation be used as a basis for further development, a notification of that event should be added.

VI. CONCLUSION

With the emergence of new wireless access technologies and new applications envisioned in new generation networks, the need for AAA becomes more pressing. The AAA solution adopted by the 3GPP and 3GPP2 for use in the IMS is based on the Diameter protocol. In this paper, we have studied the Diameter protocol and its application in the IMS Cx interface. We reviewed four open-source implementations of the Diameter protocol, and we used Open Diameter as a basis for implementing the AAA functionality that IMS needs, more specifically, the selected Cx interface functions UAR, MAR, and SAR. The conformance of the implementation to the specification was verified by testing in a laboratory setup. Our further work includes implementation of the remaining Diameter messages for the Cx interface.

REFERENCES

[1] G. Camarillo, M. A. García-Martín, *The 3G IP Multimedia Subsystem: Merging the Internet and the Cellular Worlds*, John Wiley and Sons, Ltd., England, UK, 2004.

[2] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, *Diameter Base Protocol*, IETF RFC 3588, September 2003.

[3] –, *IP Multimedia (IM) Subsystem Cx and Dx interfaces; Signaling flows and message contents*, The 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; TS 29.228, 2005.

[4] –, *Cx and Dx interfaces based on the Diameter protocol; Protocol details,* The 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; TS 29.229, 2005.

[5] J. Loughney, *Diameter Command Codes for Third Generation Partnership Project (3GPP) Release 5*, IETF RFC 3589, September 2003.