Pascal Kurtansky, Hasan, Burkhard Stiller[1]
Computer Engineering and Networks Laboratory TIK
ETH Zürich
Zürich, Switzerland
[kurtansky|hasan|stiller]@tik.ee.ethz.ch

Antonio Cuevas
Universidad Carlos III
Madrid, Spain
acuevas@it.uc3m.es

Davinder Singh, Sebastian Zander
Institute FOKUS
Fraunhofer Gesellschaft
Berlin, Germany
[singh|zander]@fokus.gmd.de

Jürgen Jähnert, Jie Zhou
Super Computing Center
University of Stuttgart
Stuttgart, Germany
[jaehnert|zhou]@rus.uni-stuttgart.de

***Abstract*—The design of an extended and generic Authentication, Authorization, Accounting, and Charging Architecture (AAAC Arch.) has been performed within the IST project MobyDick. In addition, this architecture has been implemented to address MobyDick's main objective: to facilitate the deployment of a ubiquitous mobile IPv6-based, Quality-of-Service (QoS)-aware infrastructure through a flexible and evolutionary AAAC Architecture.**

While the AAAC Arch. is based on the DIAMETER protocol, basic concepts developed cover session and services models, user profiles to allow for user mobility and QoS-aware authorization. Based on those basic building blocks for the extended AAAC Arch., the implementation of user registration, service authorization, metering, accounting, charging, and auditing is discussed. The paper closes with the presentation of the two trial sites used and their testbeds.

AAA, AAAC Architecture, QoS, Charging, Auditing, IPv6, DIAMETER

I. INTRODUCTION

Most papers which discuss issues relating to "Beyond 3G Mobile Networks" are considering IP as the final means in the eventual integration of heterogeneous access network technologies. The migration from traditional circuit switched networks towards a packet based wireless IP network infrastructure provokes a big pressure to provide packet-switched voice and data services. This can be regarded as current key drivers for the development of new communication systems and technologies. In the wired Internet and in the future wireless packed-based mobile Internet an economic and security concept, capable of satisfying both operator and user needs, is still an open issue. Mechanisms to describe and define IP services, as well as to measure the usage in a finer granularity are required. Finally an efficient way to charge for the service usage, considering different payment schemes, needs to be developed. Along with the AAA work of Internet Engineering and Research Task Forces (IETF, IRTF), a promising base for these missing functions and mechanisms have been defined. Although the required basic mechanisms are available and widely understood, their efficient and scalable integration with mobility and QoS framework is still a mute point—especially beyond the 3G mobile environment. In order to close this missing gap, within the MobyDick project [8], basic elements and concepts of the IETF AAA have been adopted, extended where required, and successfully implemented. This leads to the MobyDick AAAC Arch. [6], encompassing the facilitation of the deployment of a ubiquitous Mobile IPv6 QoS-aware infrastructure and a prototypical implementation, which is currently being evaluated in the MobyDick field trial.

The remainder of the paper is organized as follows. While Section II introduces the MobyDick enhanced AAAC Arch., Section III presents basic support components for the AAAC Arch., covering the session and services implementation, user profile specifications, and the QoS interactions. Major details of the dedicated functions of metering, accounting, charging, and auditing are discussed in Section IV. The enhanced AAAC Arch. has been utilized in two testbeds in Madrid and Stuttgart and experiences are sketched in Section V. Finally, Section VI draws conclusions.

II. THE ENHANCED AAAC ARCHITECTURE

An important goal of MobyDick is to enhance the AAA Architecture proposed by the IETF and IRTF. The enhanced architecture will allow the use of AAAC Services in a QoS-enabled Mobile IPv6 (MIPv6) environment and will provide charging and auditing functions. The whole AAAC System comprises of AAAC Client, AAAC Server and Auditing System as shown in Figure 1.

Mobile users have to register themselves to the AAAC System before they are allowed to access network resources. Registration request will be sent to the centralized AAAC Server via a AAAC Client located in current point of attachment, i.e. current Access Router.

[1] University of Federal Armed Forces Munich, Information Systems Laboratory, Germany

The main component of the AAAC Client is the DIAMETER Client, which is responsible for accepting registration requests and forwarding them to the AAAC Server. Therefore, the DIAMETER Client runs both User Registration Protocol (URP) and DIAMETER protocol [3].

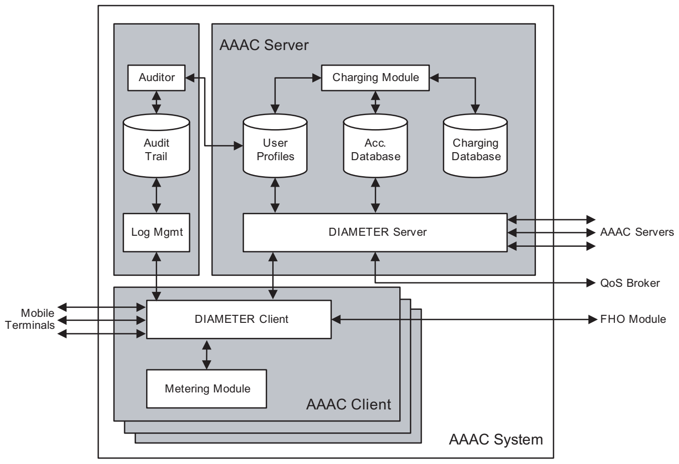


Figure 1. AAAC Architecture for a QoS-enabled Mobile IPv6 Environment.

The main component of the AAAC Server is the DIAMETER Server, which is responsible for authentication, authorization, and accounting based on the user profile information. The accounting data will be stored in an accounting database before being fetched by the charging module to calculate charges for each service usage. The resulted charges will be placed in the charging database for billing purpose. The DIAMETER protocol is also used for communication among AAAC Servers in the case where mobile users are roaming.

Auditing enables further function with respect to evaluation of audit trails to determine whether users and network activities comply to pre-established Service Level Agreements (SLAs).

Figure 2 depicts detail components within DIAMETER Server and Client. Principally, mobility within an administrative domain should be transparent to AAAC System; however in order to allow for AAA context transfer, an interface between DIAMETER Client and Fast Handover (FHO) Module is needed. The AAA context transfer enables mobile users to move to another point of attachment without having to re-register.

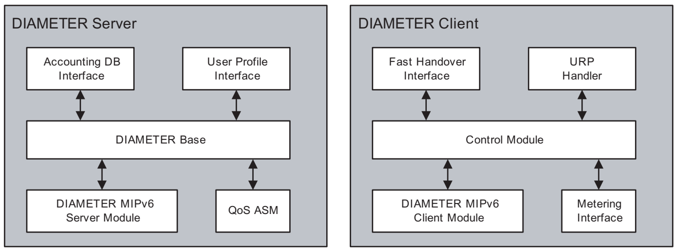


Figure 2. Detail Components of DIAMETER Server and Client.

With the help of well-defined Application Specific Modules (ASMs), an AAAC Server can be adopted to various applications. The MobyDick architecture has an ASM for handling Mobile IPv6 and another for interaction with QoS Broker. In this particular application, the service user is the Mobile Terminal (MT) on behalf of the human user. It is assumed that any authentication and authorization request on behalf of a user may originate from alternate AAAC Servers or from the MT via the AAAC Client.

III. SUPPORT COMPONENTS FOR AAAC

The key supportive components for the enhanced AAAC Architecture include the session model, service definitions, the user profile, and a QoS interaction model, each of which is discussed below.

A. *Session Concept*

The notion of a session ID is introduced to bind together a set of related activities. Each session is associated with a unique session ID and is linked to one user with a unique user ID. The session ID together with the user ID allows a provider to combine accounting data for the different activities and generate a bill.

The AAAC Client creates a new session with a new ID after several events, e.g. a user switches on his mobile terminal (MT) and requests authentication and authorization. Within a running session, a customer can—according to his Service Level Agreement (SLA) which is held in the user profile—use the subscribed services offered by the operator. It is possible that a user has more than one open session. Every authorization is bounded to a certain lifetime, and before the lifetime expires, the MT must perform re-authentication. Re-authentication does not affect running sessions and the corresponding accounting.

TABLE I. QOS DESCRIPTION IN MOBYDICK

Service		**Rel. Prio.**	**Service Parameters**	**Service Description**
Name	***Class***			
S1	EF	1	Peak BW: 32 kbps	Real time Services
SIG	AF41	2a	Unspecified	IP signaling only
S2	AF21	2b	CIR: 256 kbps	Priority data transfer
S3	AF1*	2c	AF11 – 64 kbps AF12 – 128 kbps AF13 – 256 kbps	Olympic service (better than BE: streaming, FTP, ...)
S4	BE	3	Peak BW: 32 kbps	Best effort
S5	BE	3	Peak BW: 64 kbps	Best effort
S6	BE	3	Peak BW: 256 kbps	Best effort

According to the IETF Differentiated Services [1], the Differentiated Services Code Point (DSCP) identifies a particular service in a unique way. When the user wants to utilize one of the above services, he needs to send IP packets into the network marked with the corresponding DSCP of the chosen service. After this marking the QoS Manager (a QoS entity in the Access Router) and QoS mechanisms deployed in the network will be in charge of granting this level of service. All details of these QoS definitions according to Table 1 have to be provided. Further details can be found in the subsection on QoS interactions and in [7].

Each MobyDick operator must have a service definition table stored in its AAAC System. In the most general AAAC environment, each operator could use a different table—i.e. the mapping from a service to a DSCP value might be different in each administrative domain.

The use of these DSCP values in IP packets still focuses on network layer service descriptions and their charging. Application-based charging is currently not supported and can be enabled only, if (a) new service definitions will be used, which can be mapped onto unique IP packet flows or (b) application layer data are utilized and investigated within the MobyDick infrastructure's components.

B. *User Profiles*

A user profile (UP) is defined as a data record of all user-specific data. This includes authentication data, data of the SLA and auditing information. The UP is unique and is initially created, when the user signs his SLA, which allows or disallows the customer to use certain services. The UP is stored in the home AAAC Server (AAAC.h) of that MobyDick operator with whom the customer has established an SLA.

The user-specific information of the UP is needed to properly identify the customer. The next part of the UP contains tariff IDs for each service being offered. These tariff IDs are used by the charging component to select the appropriate tariff function. In addition—for auditing purposes—the UP contains the level of availability guarantee and guarantee of success of registration and session setup. Finally, the UP contains QoS-related information, called the Network View of the User Profile (NVUP). The NVUP contains the DSCPs of those services a user is allowed to use. In the case of a roaming user, the NVUP is transferred to the foreign AAAC Server (AAAC.f). The foreign operator will map the DSCPs to actual services, according to its service table.

C. *QoS Interaction*

The interaction with the QoS System is performed at the registration phase. The NVUP, along with the Mobile IPv6 Care-of-Address (CoA) is transferred to the QoS System. That makes the QoS System aware of the QoS service granted to the user.

The QoS System conforms to the DiffServ Architecture where Access Routers (ARs) perform policing of packets sent by a user's terminal into the network. The AR outsources the policing decision to the QoS Broker, following the COPS outsourcing model [5]. Based on the NVUP, the QoS Broker sends its decision to AR whether to grant service or not. Accordingly the AR will either forward user packets at the specified quality or drop them. Therefore, in roaming scenario the QoS Broker in the foreign domain (QoSB.f) must receive the NVUP via AAAC.f, which gets this from AAAC.h.

The QoS Broker keeps NVUP of a user for a certain lifetime, which is the same as the DIAMETER authorization lifetime. When this time expires, the registration process will be repeated and the NVUP will be dumped again to the QoS Broker. More details of this QoS Interaction can be found in [4].

IV. METERING, ACCOUNTING, CHARGING, AND AUDITING

After having seen, which basic components are underlying the AAAC System's implementation, the metering of service usage, the task of performing the accounting, the charging process, and the auditing scheme are presented in this section in detail.

A. *Metering*

Measuring network traffic has a long history within the IETF. A working group within the IETF, the so called Real Time Traffic Flow Measurement (RTFM) Working Group, developed an architecture for traffic measurement [2]. An IETF RTFM flow is a bi-directional stream of packets between two endpoints, each defined by a set of attribute values, which can be determined flexibly. Via these flows, the notion of a virtual "connection" is introduced to the IP layer. Historically, this architecture was designed for accounting issues and as a base for further charging and billing processes in order to detect and handle resource/service usage more accurately.

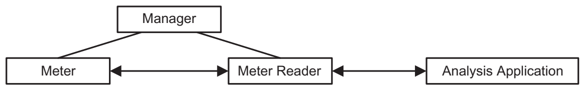


Figure 3. The IETF Real Time Traffic Flow Measurement Architecture.

Figure 3 depicts the overall metering architecture. The heart of this traffic flow measurement architecture is the meter, which is placed at measurement points determined by the network operator. Each meter selectively records network activity as directed by its configuration settings. It can also aggregate, transform and further process the recorded activity before the data is stored. The processed and stored results are called the usage data. In MobyDick the IETF RTFM metering framework is used. In order to fulfil the MobyDick requirements, the existing IETF RTFM reference architecture has been modified and integrated into the overall AAAC Framework. An AAAC/Meter interface was implemented in order to allow the DIAMETER Client to access the metered data and to configure the meter.

B. *Accounting*

Accounting is the process of associating meter data—i.e. data about resource consumption—with a user (user ID) and a session (session ID). The metering infrastructure will receive a trigger from the AAAC Client to start metering when a new session is started. The AAAC Client forwards the accounting data (via the AAAC.f, in the case of roaming) to the AAAC.h, which stores the accounting data in the Accounting DB. During session runtime, the AAAC Client is generating interim messages, containing accounting data. When a session is closed, the AAAC.h adds the last entry to the Accounting DB for that session.

If the result of a user registration is negative, no accounting takes place. In case of a successful registration the AAAC Client configures its accounting functionality according to the accounting policies contained in the registration response message. If no such policies are present it configures accounting according to a default configuration specified before starting the client.

C. *Charging*

Charging calculates the price for a given service consumption based on accounting information and the SLA. It maps technical values into monetary units by applying a tariff. The accounting data together with the user profiles covers all information needed by the charging component (CC).

In MobyDick an initial scenario of the postpaid business case has been investigated. Therefore, it is sufficient that the CC will access the Accounting DB periodically and extract new records. Charging is session-based, i.e. only closed sessions will be processed. After selecting new accounting data, the CC extracts the user (User ID) associated with the current session (Session ID). The User ID is the key index used by the CC to select the appropriate SLA from the user profile database. Now the CC has obtained all necessary information to apply the tariff function for a given customer and a given session. The result of the charge calculation—i.e. the charge in €—is written to and stored in the charging database located in the AAAC.h.

D. *Auditing*

MobyDick's auditing objective is the detection of SLA violation. MobyDick defines the following commitments: entity availability, success of user registration, and success of session setup. The availability guarantee defines the availability of MobyDick entities responsible for user registration, session setup, and service delivery, in unit of time or in percentage within each period of a predefined length. Those entities encompass AAAC Client, AAAC Server, QoS Manager, and QoS Broker.

The success of a user registration within <n> minutes is guaranteed, if at least <r> valid retries have been made with a valid user ID. To determine whether registration attempts are successful each AAAC Client must log all valid registration requests and responses.

The success of a session setup within <n> minutes is guaranteed, if at least <r> valid retries have been made with valid DSCP. To determine whether session setups are successful each QoS Manager must log all valid service requests sent to the QoS Broker and its corresponding response.

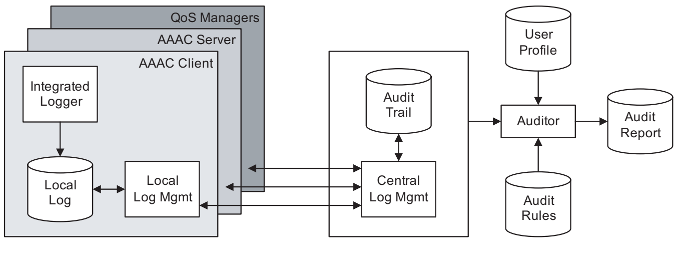


Figure 4. Logging and Auditing Architecture.

Figure 4 shows the logging and auditing architecture. Events and actions of AAAC Clients, AAAC Server, QoS Managers and QoS Brokers, which need to be logged will first be stored in the local log via an integrated logger. This local log will be managed by Local Log Management module, which has the responsibility to transfer this log to the Central Log Management module (CLM). The Central Log Management module will store this log to the Audit Trail from where it is later fetched by the auditor to be examined with respect to the SLA. In processing this audit trail, the auditor will make use of the user profile database and the audit rules, which define the violation conditions. The results of the auditing are stored in an audit report.

V. AAAC IN THE MOBYDICK TESTBED

The first subsection describes the MobyDick Testbeds and the following subsection contains the evaluation results.

A. *Testbed Description*

In order to show and to prove our new concepts of the AAAC Architecture, two MobyDick Testbeds were built up. The testbeds were located in Spain (Madrid) and Germany (Stuttgart) and they had the same infrastructure, besides the UMTS access-network which could only be set up in Stuttgart—there was no public license available in Spain. Both testbeds were completely running and interconnected with IPv6. Figure 5 shows all major components and the interconnection of the two MobyDick Testbeds.

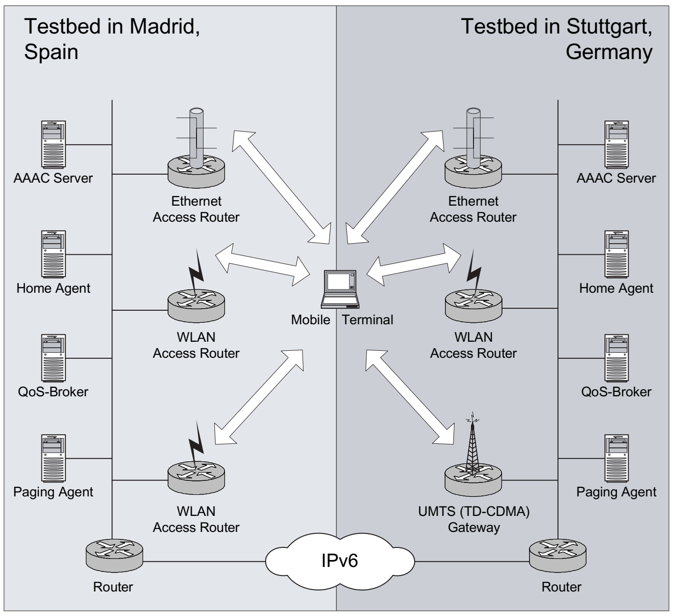


Figure 5. The MobyDick Testbeds in Madrid and Stuttgart.

On both testbeds, several access technologies were installed like UMTS (TD-CDMA), WLAN (802.11b), and fixed networks (Ethernet). The TD-CDMA support was achieved by the direct connection of the base station to the IP network. Several of the elements defined in the 3GPP UMTS architecture such as RNC, SGSN, GGSN could be eliminated and thus, simplifying the overall protocol stack. The base station was developed using a platform supporting W-CDMA-TDD mode and the network access is provided by a radio access router which controls one radio cell.

The Mobile Terminals (MT) were equipped with interfaces of all three access technologies mentioned before. Therefore, seamless intertechnology handovers were possible: From the fixed network (Ethernet) to any wireless technology, or from one wireless technology to another one. Different multimedia user applications were running on the MT to test the handover scenarios.

The two MobyDick Testbeds were connected together via a QoS aware IP backbone. The two administrative domains—Madrid and Stuttgart—were connected via at least one DiffServ egress/ingress router to the IP backbone. By having these two administrative domains, it was possible to define a roaming scenario like it is present in the current GSM networks: A user who has left its home domain (e.g. the domain in Stuttgart) and who is then accessing the internet in a foreign domain (e.g. the domain in Madrid).

The Paging Agent was responsible for generation and distribution of generic IP paging messages. The paging signalling messages addressed all registered paging attendants in the paging area of the access routers.

Whereas DIAMETER allows roaming and user mobility, it must be completed by device mobility. Device mobility is achieved by MIPv6 with the Home Agent being the key entity.

QoS is a basic service within MobyDick and part of the user profile. The entity in charge of its control is the QoS-Broker.

B. *Evaluation Results*

For a thorough test of registration, a roaming scenario was defined. The two testbeds and their AAAC platform allowed the verification of multi provider mobility scenarios by means of AAAC Interactions between Stuttgart and Madrid. The following tests have been performed: a Stuttgart-based roaming user registered in Madrid's domain, cf. Figure 6.

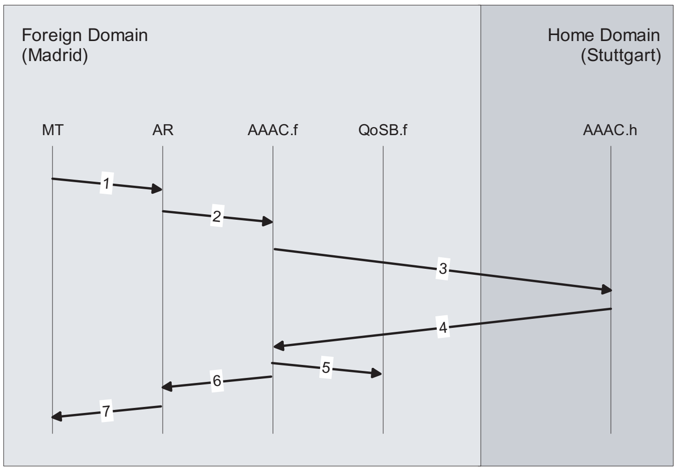


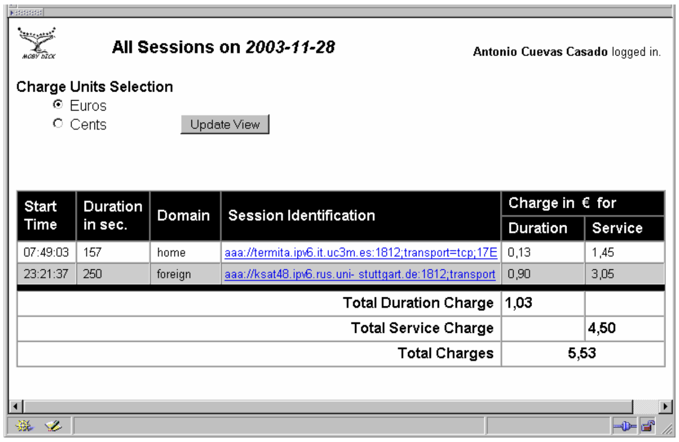
Figure 6. Roaming scenario (simplified): A user with home domain in Stuttgart is registering in foreign domain of Madrid.

In a first step—cf. (1) in Figure 6—the MT contacts the AR, i.e. it sends out a registration request. The AR is forwarding this request to the AAAC.f, (2). The DIAMETER register request is forwarded by Madrid's AAAC Server (acting as AAAC.f) to Stuttgart's AAAC Server (acting as AAAC.h), (3). The user profile is then transferred from the AAAC.h to the AAAC.f, (4). The AAAC.f dumps the NVUP to the QoSB.f in Madrid, (5). As a final step, the AR and the MT is informed about the successful registration, (6) and (7).

The average time it took to register a user in its home domain was about 280ms and 350ms for a roaming user. Major factor in this time is the Diffie Hellmann key calculation in the AAA client. In the case of roaming, routing from the AAAC.f to the AAAC.h and the processing in AAAC.h account for less than 10% to this time. The Round trip time (RTT) between the AAAC server in Madrid and in Stuttgart (in case of roaming) is about 70ms. Thus, the stateless AAA implementation in MobyDick—the RFC allows it to be stateful or stateless—causes very few processing time.

Charging is a resource consuming process, since it has to deal with a huge amount of accounting records. To increase the charging performance, the AAAC.h entity was distributed onto two machines. From a logical point of view, the AAAC.h still acts as one entity. On the physical level, AAA is done on one and charging on the another machine. The MySQL database holding the accounting and charging data, can be located on either machine and is accessed by both machines.

After the charge calculation is completed, the users are able to login and to view their charges, cf. Figure 7. The scenario in Figure 7 is just the opposite of the one mentioned earlier: A user from his home domain in Madrid (first session with AR termita) was roaming in Stuttgart (second session with AR ksat48). The tariff (SLA) for this user was duration- and QoS-based. Charges are presented in a web browser and the logic behind was implemented using a webserver, both completely running over IPv6.



All Sessions on *2003-11-28*

Antonio Cuevas Casado logged in.

Charge Units Selection

[✓ Euros]
Cents

Update View

Start Time	Duration in sec.	Domain	Session Identification	Charge in € for	
				Duration	Service
07:49:03	157	home	aaa://termita.ipv6.it.uc3m.es:1812;transport=tcp;17E	0,13	1,45
23:21:37	250	foreign	aaa://ksat48.ipv6.rus.uni- stuttgart.de:1812;transport	0,90	3,05
			Total Duration Charge	1,03	
			Total Service Charge		4,50
			Total Charges	5,53	

Figure 7. Consolidated Charge Presentation to the User.

Two parameters regarding logging and auditing are evaluated: the Log Transfer Delay and the Auditing Speed.

The Log Transfer Delay (LTD) is defined as the average time required to transfer a single log stored in a Local Log into a central Audit Trail. This delay depends surely on the network delay, database access time, etc., however there are other more interesting parameters, which may also have an impact. These parameters are $n_{\text{local logger}}$ (number of Local Log Management Modules (LLM)) and n_{log} (#Logs = number of logs in the Local Log). Therefore, LTD = f($n_{\text{local logger}}$, n_{log}). Figure 8 compares the LTDs obtained from the experiments with 1 LLM (1-LLM-System) and the experiments with 2 LLMs (2-LLMs-System) for the same number of logs. The figure shows that the implemented LLM and Central Log Management Module (CLM) can transfer the same amount of logs a bit faster in a 2-LLMs-System than in a 1-LLM-System. This is reasonable because both transfers in a 2-LLMs-System (from LLM-1 and LLM-2) were running in parallel and the CLM was able to receive both data streams simultaneously. The figure also shows that there is only a small deviation of LTD for different number of logs ranging from 1'000 to 50'000 logs.

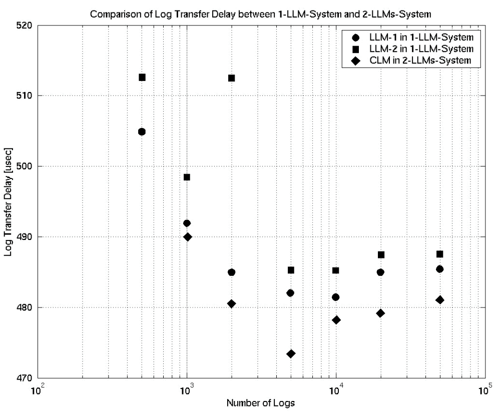


Figure 8. Comparison of LTDs between 1-LLM- and 2-LLMs-System.

These results show that the implemented LLM and CLM are scalable with respect to number of logs and number of LLMs (number of Access Routers).

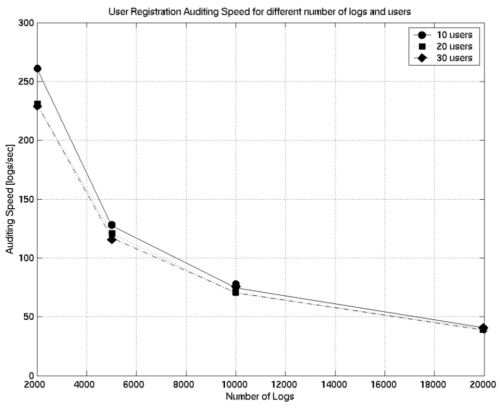


Figure 9. User Registration Auditing Speed for different number of logs and users.

Auditing Speed S is defined as the number of logs that is processed within a unit of time. The Auditing Speed is dependent on the following factors: n_N (number of users or entities in the audit trail), n_{log} (number of logs in the audit trail), $type_{log}$ (type of event logs). Therefore, $S = f(n_N, n_{log}, type_{log})$.

The Audit Time encompasses the time to retrieve the users or entities identity, the time to retrieve the logs from the Audit Trail, the time to store the processed logs in the Archive, and the time to delete the processed logs from the Audit Trail. The Audit Time has been evaluated with different number of users or entities, and different number of logs, cf. Figure 9. Auditing Speed is number of logs divided by the Audit Time.

The larger the amount of logs to be processed the smaller is the Auditing Speed. This is an undesirable behavior, although an asymptotic limit of this deceleration seems to exist. In this regard, the implementation of the auditor must be improved. Database queries may need to be made more optimal.

VI. CONCLUSIONS

The MobyDick AAAC Architecture as developed, implemented, and currently evaluated in the distributed field trial between Stuttgart and Madrid addresses a number of highly relevant questions. The traditional AAA Architecture is extended with charging, mobility, and security support in order to push the commercialization process of a wireless beyond 3G network. Major results have been obtained include the verification of the architecture. It should be mentioned that the overall AAAC System distributed between Stuttgart and Madrid is integrated into the public IPv6 network which provides the flexibility to extend the system to a world-wide scale.

The architecture considers all required interfaces between the AAAC System and a QoS Broker, between the traditional tasks of AAA and Charging, and between AAA and SLA Auditing. Additionally, the architecture has been integrated with IP-based mobility management.

ACKNOWLEDGMENT

The work presented was funded by the EU project IST-2000-25394 "MobyDick" (Mobility and Differentiated Services in a Future IP network). The authors would like to thank M. Lötscher and B. Schern for supporting the development of the accounting and charging component.

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", Internet Engineering Task Force, RFC 2475, December 1998.

[2] N. Brownlee, C. Mills, G. Ruth, "Traffic Flow Measurement: Architecture", IETF RFC 2722, October 1999.

[3] P. Calhoun et al, "Diameter Base Protocol", draft-ietf-aaa-diameter-07.txt, IETF work in progress, July 2001.

[4] A. Cuevas et al. "Mechanisms for AAA and QoS Interaction" 3rd Workshop on Application and Services in Wireless Networks, Bern, Switzerland, July 2003, ISBN: 3-9522719-0-X.

[5] D. Durham, Ed., "The COPS (Common Open Policy Service) Protocol" RFC 2748, January 2000

[6] Hasan, Davinder Singh, Sebastian Zander, Moritz Kulbach, Jürgen Jähnert, Burkhard Stiller, "The Design of an Extended AAAC Architecture", IST Mobile & Wireless Telecommunications Summit, Thessaloniki, 2002.

[7] Victor Marques et al., AN IP-BASED QOS ARCHITECTURE FOR 4G OPERATOR SCENARIOS. IEEE Wireless Communications. June 2003

[8] MobyDick Website, www.ist-mobydick.org