Design and Prototypical Implementation of an Accounting System for an AAA Server

Semesterarbeit im Fach Informatik

Vorgelegt von

Adrian Bachmann
aus Emmen, LU
02-704-245

Angefertigt am
Institut für Informatik
der Universität Zürich
Prof. Dr. Burkhard Stiller

Betreuer: Cristian Morariu

Abgabe der Arbeit: 31. August 2005

Summary

Introduction

Modern communication requires strong security support in many different aspects including access control, accounting of service usage, secure communication, non-repudiation and auditing. Most of the Interent Service Providers today require a user to be authenticated and authorized for using their services (access control). The classic AAA (Authentication, Authorization and Accounting) term describes a collection of requirements (techniques) of delivering a secure approach to Service Provisioning.
Authentication is the process of “validating the authenticity of something or someone”. The authentication proccess is used to verify the user requesting a service from the service providers is really who he pretends to be.
Authorization is the process of checking an entity’s rights to perform a certain action. For example the process of finding out if a user has the rights to view a file, or if a certain process is allowed to be run on a specific processor.
Accounting describes the collection of data about resource consumption. This includes the control of data gathering, transport and storage of accounting data.
Currently the IETF and IRTF are working on the standardization of a new protocol for AAA systems - called Diameter - as well as a generic architecture for AAA services.
For every service a user is consuming, the provider has to have an information about which user consumed the service, what exactly was the service offered, when, and “how much” of that service the user consumed. For this, monitoring components are embedded into the service provisoning platform that permanently gathers data about service consumption. These components are seen as Application Specific Modules in the AAA Generic Architecture and use an AAA protocol (Diameter) for transferring recorded data to a central accounting database.

Aims and Goals

The key aim of this thesis work is to design and prototypically implement the Accounting module for an AAA server, based on the Generic AAA Architecture defined in RFC 2903 and Diameter protocol specifications. The resulting protocol and architecture shall provide a solution for offering accounting services to a Mobile Grid. It will also be used at a later stage together with various charging models for creating a charging mechanism for future mobile grids. A mobile grid environment, by its heterogenos nature brings new challenges to all the tree A-s in a traditional AAA environment. Regarding the accounting proccess, new types of resources have to be accounted for which require new parameters that have to be present in accounting records. Besides the traditional accounting of time, bytes, and packets, a grid service might need to account for CPU usage, memory consumption, or even accessed/containing information. The accounting module shall provide generic interfaces for possibly different monitoring entities that adapt to the type of resource being accounted for. The access to accounted for data shall be secure and reliable. Secure in this context means that accounted for records for a certain service can be created by certain entities that are aproved by that service provider. This requirement can be realized using X.509 certificates or other kind of credential tokens. Encryption of accounting messages shall be offered as a communication option between the AAA client and the AAA server. Reliability refers to the posibility of retreiving the accurate accounting information for a certain service/resource usage for charging consumptions.

Results

The field of AAA-Services today is widly used and well-known in professional circles. Nevertheless an introduction into the topic is essential. This thesis introduces in a good readable and understandable way into the topic also for non-professionals. This introduction is insofar important, that the reader needs a basic knowledge about the problems in the classical field of AAA-Services. The introduction shows in this case also the difference between the two AAA-protocs: RADIUS and Diameter and shows the problems in the old DIAMETER protocol which are solved in the mean time by Diameter.
In respect on the requirements of mobile grids, the resulting prototype has to be implemented with the new Diameter protocol. OpenDiameter, an opensource-project, implements the diameter base protocol in respect to the generic AAA-architecture. In fact of that, this work bases on the opendiameter-framework. But the installation and compilation of the framework wasn't really unproblematic, so the thesis delivers also some help in this case with an installation manual which explains the whole installation procedure including all the additionally needed libraries.
In the design and implementation part, the thesis checks the requirements in respect on mobile grids. In respect on the ressources which sould be measured and accounted in a mobile grid environment, new accounting parameters were defined. This parameters are afterwards mapped into Diameter-AVPs to use them in OpenDiameter. The resulting prototype itself offers a reliable communication between client and server in respect on the generic AAA-Architecture and the Diameter base protocol specification. The interfaces are designed and implemented as open as possible so that we have an easy scalable and expandable product. Many additionally implemented AVPs offer the required functionality for the usage of mobile grids. Later also additionaly AVPs could be added easily to expand the ressources which a mobule grid whish account for. The underlaying architecture of the prototype is well explained and also the source code is well documented. All the important design choices are explained and described.
In the last part the thesis checks all the requirements for SIP compatibility. In this case the thesis not only shows what's needed, but also concrete design proposals including the design of new AVPs for SIP are given.

Further Work

In this prototype the interface to the database isn't implemented, thus it's a goal of another work. Althrough we should think about the data which should be stored in the database and in which format.
Acctually the accounting-client-applications which are able to send records to the server are fix implemented in the server and client. In a further release there should be defined an interface to register new accounting-client-applications. This could be realized for example with an additional diameter-command.
The class AAA-AVPDatabase actually holds an own array with all the informations about the AVPs. This informations about the AVPs like ID, name and datatype are hardcoded in the sourcecode. That's not really a good solution in respect on flexibility and adaptability. This data should be collected though the class (while it's object-creation) from the dictionary.xml-file.
At least, there exists still many functions which sould be implemented properly with an accurate functionality. Many functions of the framework actually are only implemented for testing-purposes. As example we can take the function IsLastRecordInStorag() of the class AAA_AcctModuleClientAcctRecCollector.

Danksagung

An dieser Stelle möchte ich mich bei all denen herzlich bedanken, welche zur Verwirklichung dieser Semesterarbeit beigetragen haben.
Aufrichtiger Dank gilt an erster Stelle Herrn Prof. Dr. Burkhard Stiller, welcher mir die interessante Auseinandersetzung mit diesem Thema in Rahmen einer Semesterarbeit ermöglichte.
Spezieller Dank gilt auch Cristian Morariu, welcher mir stets tatkräftig mit hilfreichen Hinweisen sowie aufschlussreichen Meetings zur Seite stand und mich motivierend während meiner Arbeit begleitete.
Zu guter Letzt möchte ich meiner Freundin Maria danken, welche mein grosses Engagement für diese Arbeit stets unterstütze und mir den nötigen privaten Halt gab.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Einführung

Moderne Kommunikation und die damit verbundenen Dienstleistungen verursachen Kosten, welche im Allgemeinen von den Benutzern zu tragen sind. Um die Nutzung dieser Dienste nur registrierten Personen zu gestatten und die beanspruchten Leistungen exakt abzurechnen, ist es nötig, sowohl den Benutzer eindeutig zu identifizieren als auch Daten über den Umfang der Nutzung zu sammeln.
Mit Mobiltelefonen ist heutzutage fast weltweit ein Netzzugang und die Nutzung der entsprechenden Dienste möglich. Es muss jedoch sichergestellt werden, dass einmal nur bestimmte, sowie registrierte Benutzer den Netzzugang nutzen können und weiterhin abrechnungsrelevante Informationen, wie die Dauer der Nutzung oder das Transfervolumen, gesammelt werden. Dies gestaltet sich insofern schwierig, da die Zugangsknoten bzw. Netze in verschiedenen Verwaltungsdomänen liegen und von verschiedenen Dienstanbietern betrieben werden können. Trotzdem sollte eine für den Kunden transparente und einfache Abrechnung angestrebt werden.
Bereits 1985, als der Rechnerzugang per Modem möglich wurde, traten ähnliche Probleme auf. AAA-Protokolle (Authentifizierung, Autorisierung und Abrechnung), welche eine genaue Zugriffssteuerung auf die verschiedenen Dienste gestatten, waren die Lösung. Mit RADIUS, damals als Dial-In Protokoll konzipiert, konnte man alle Anforderungen erfüllen. AAA-Server wie auch RADIUS werden heute in vielen Bereichen erfolgreich eingesetzt. Ob das ursprüngliche RADIUS-Protokoll jedoch heutigen Bedürfnissen auch noch gewachsen ist, werden wir zu einem späteren Zeitpunkt sehen.
Relativ früh zeichneten sich die ersten Probleme mit RADIUS ab, wodurch bereits 1996 durch Pat Calhoun unter der Bezeichnung Diameter mit der Weiterentwicklung von RADIUS begonnen wurde. In dieser Semesterarbeit werden wir beide Protokollen kurz kennen lernen wie auch deren Vor- und Nachteile.

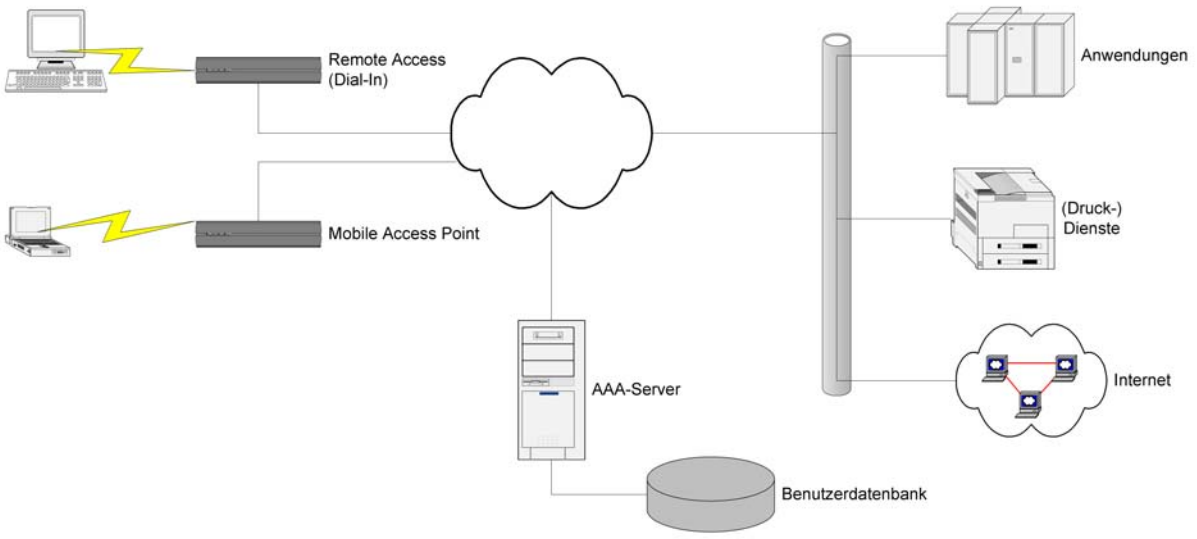

Abb. 1: AAA-Server steuern die Nutzung verschiedener Dienste

1.2 Aufgabenstellung und Ziel dieser Semesterarbeit

Ziel dieser Semesterarbeit ist das Design und die prototypische Implementierung eines Abrechnungs-Modul für einen AAA-Sever. Die Basis bildet die generische AAA Architektur,

welche in RFC 2903 definiert ist, sowie die Diameter Protokoll-Spezifikationen. Das resultierende Protokoll sowie die Architektur soll Abrechungs-Dienste für Mobile Grids zur Verfügung stellen. Das resultierende Abrechnungs-Modul soll auch für eine spätere Verwendung zusammen mit verschiedenen Verrechnungsmodellen für Mobile Grids genutzt werden können. Eine Mobile Grid Umgebung bringt dabei, durch seine heterogene Natur, neue Herausforderungen für alle drei A-s einer traditionellen AAA-Umgebung. In Bezug auf die Abrechnung müssen neue Arten von Ressourcen verrechnet werden können, weshalb weitere Parameter in den Abrechnungs-Aufzeichnungen berücksichtigt werden müssen. Nebst der traditionellen Verrechnung von Zeit, Bytes und Pakete sollte ein Grid Service auch CPU-Auslastung, Speicher-Konsum oder zugegriffene/inhaltsbezogene Informationen verrechnen können. Das Abrechnungs-Modul sollte dabei generische Schnittstellen für alle möglichen messbaren Grössen bieten. Der Zugriff zu den Abrechnungs-Daten sollte dabei sicher und zuverlässig sein. Sicher in diesem Kontext bedeutet, dass Abrechununngs-Dateneinträge nur von den vom Service Provider dazu ermächtigen Stellen vorgenommen werden dürfen. Diese Anforderung kann durch den Einsatz von X.509 Zertifikaten oder einer anderen Art von Berechtigungsnachweisen realisiert werden. Eine Verschlüsselung der Abrechnungs-Nachrichten sollte als Kommunikationsoption zwischen dem AAA-Client und AAA-Server offeriert werden.

1.3 Aufbau der vorliegenden Semesterarbeit

Nach einer kurzen Einleitung sowie Erläuterung der Aufgabenstellung werden kurz die AAA-Architektur inklusive der verfügbaren Protokolle RADIUS und Diameter vorgestellt. Dabei werden die Schwächen des RADIUS-Protokolls und damit die Notwendigkeit des neuen Protokolls Diameter aufgezeigt.
In einer nächsten Phase wird konkret auf das OpenDiameter-Framework eingegangen, welches die Basis der vorliegenden Arbeit bildet. Danach wird die gefundene Lösung eines Abrechnungs-Moduls vorgestellt und beschreiben wobei auch die getroffenen (Design-)Entscheide erläutert werden.
Die Anforderungen für eine SIP-Kompatibilität werden in einem seperaten Kapitel geprüft und entsprechende Lösungsvorschläge vorgestellt.
Der Abschluss dieses Berichts bildet ein Fazit, ungelöste Probleme sowie weiterführender Arbeiten bzw. Literatur.

2 AAA

2.1 Die drei As

Die Abkürzung AAA steht für Authentifizierung (Authentication), Autorisierung (Authorization) und Abrechnung (Accounting) [WIKI05]:

Authentifizierung ist die Grundlage für Autorisierung und Abrechnung. Bei der Authentifizierung wird die Identität eines Benutzers überprüft. Typischerweise werden dazu eine Benutzerkennung bzw. Benutzername und ein Passwort verwendet, welche der Benutzer über ein Authentifizierungsprotokoll (beispielsweise PAP oder CHAP) dem AAA-Server miteilt.

Autorisierung bestimmt, welche Dienste und Anwendungen eine authentifizierte Identität benutzen darf. Dabei kann nicht nur ein Internetzugang geregelt werden, sondern auch Speicherplatz, Druckdienste oder Benutzereinstellungen können mittels AAA-Protokollen verwaltet werden.
In einer Benutzerdatenbank lassen sich verschiedenste Regeln und Benutzerprofile zentral

ablegen und warten. Dabei sind die Regeln sehr flexibel konfigurierbar. Kriterien für den Zugang reichen von Gruppenregeln bis zu orts- oder zeitbasierten Regeln.

Abrechnung stellt sicher, dass Daten über die Nutzung der einzelnen Dienste aufgezeichnet werden. Die Daten umfassen die Benutzerkennung, die Nutzungsdauer (Sessiondauer), das Transfervolumen oder andere dienstspezifische Angaben wie Speicherkonsum, Prozessorkonsum, etc.

Diese Daten können ausgewertet werden. Kommerzielle Anbieter erstellen darauf basierend Rechnungen für Kunden. Andere Anwendungen, wie Auslastungs- und Nutzungsanalysen sind ebenfalls möglich.

2.2 AAA-Architektur

Die AAA-Architektur besteht aus AAA-Clients, AAA-Servern und Proxies [RFC2903, HP02]:

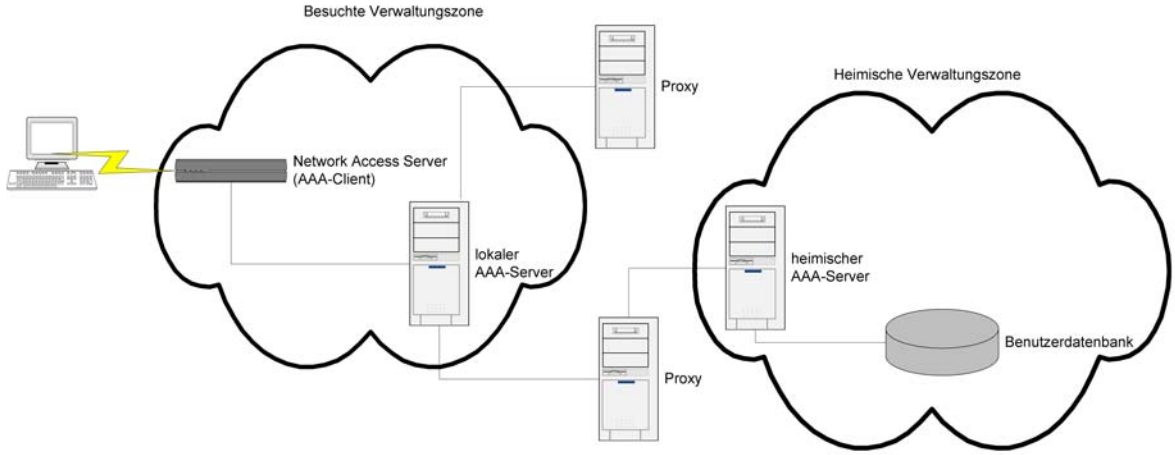

Abb. 2: AAA-Client, AAA-Server und Proxy

AAA-Clients nehmen die Zugangsanforderung vom Benutzer entgegen. Hierbei handelt es sich oft um Network Access Server (NAS), welche Benutzern beispielsweise die Modemeinwahl ermöglichen. Der AAA-Client gibt die Authentifizierungsanfrage an den lokalen AAA-Server weiter.

AAA-Server beantworten die Anfragen der Clients. Anhand einer Benutzerdatenbank entscheidet der Server, ob dem Benutzer die gewünschten Rechte gewährt werden. Kann der Server eine Anfrage nicht bearbeiten, da er die Identität des Benutzers nicht kennt, so leitet er die Anfrage an einen ihm bekannten AAA-Server weiter.

AAA-Proxies sind AAA-Server, welche AAA-Anfragen weiterleiten.

Will ein Benutzer einen Dienst nutzen, so stellt er eine Verbindung zum entsprechenden Network Access Server her. Er meldet sich mit seiner Benutzerkennung sowie in den meisten Fällen einem Passwort an.

Der NAS agiert gegenüber dem lokalen AAA-Server als AAA-Client und übermittelt die Authentifizierungsanfrage an den Server. Kennt der Server die Identität des Benutzers, so kann der Server nach Prüfung des Passwortes die Nutzung der Dienstleistungen gestatten.

Ist dem Server der Benutzer jedoch unbekannt, so leitet er die Anfrage des Clients an ihm bekannte AAA-Server weiter. Dies setzt jedoch voraus, dass der Server möglichst viele weitere Server kennt und auch stets eine sichere Kommunikation zwischen den Servern gewährleistet ist. Eine so genannte Security Association muss dabei zwischen den Servern bestehen.

Damit Benutzer über Verwaltungszonen hinweg und auch ausserhalb ihrer Heimatdomäne Dienstleistungen nutzen können, wurden Broker eingeführt.

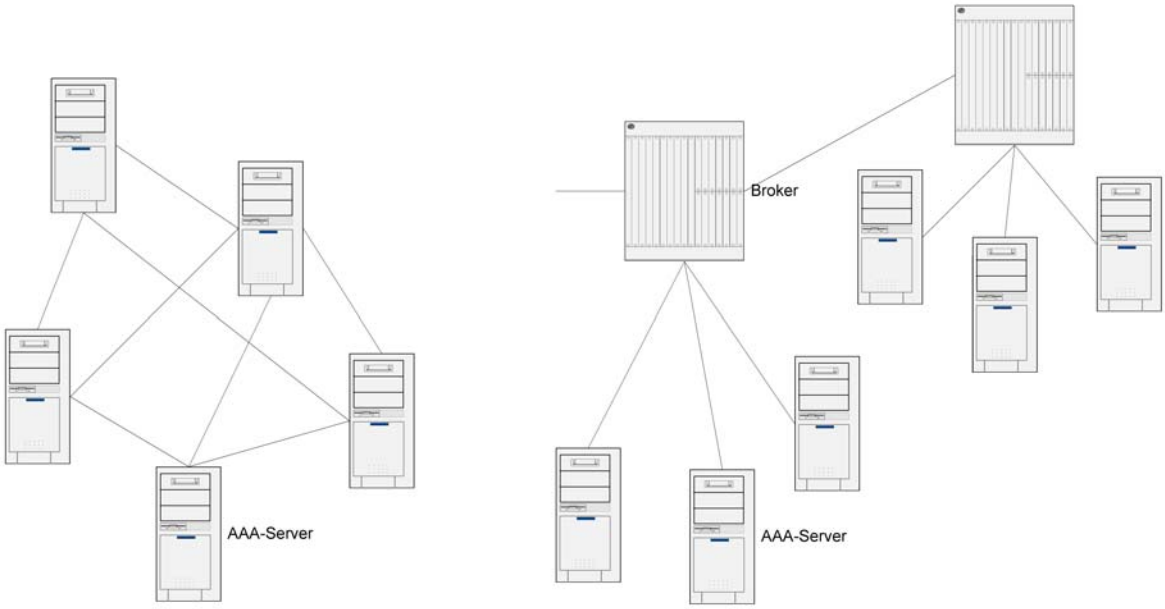

Abb. 3: AAA-Architektur ohne und mit Broker

AAA-Server besitzen dabei Security Associations mit Brokern, welche wiederum Security Associations mit sehr vielen weiteren Servern besitzen. Der Broker kann dann eingehende Anfragen an den entsprechenden Server weiterleiten. Dadurch müssen AAA-Server nicht mit jedem existierenden AAA-Server eigene Security Associations eingenen, sondern können über einen Broker, welcher als Vermittler auftritt, agieren.
Die AAA-Architektur entspricht einem klassischen Client-Server-Modell. Dabei fordert stets der Client die Dienste eines Servers an. Dies bedeutet insbesondere, dass der Server dem Client nicht willkürlich und von sich aus Nachrichten senden kann. Einer Nachricht an den Client geht immer eine Anfrage des Clients an den Server voraus.

2.3 AAA und Mobile Grids

Ursprünglich wurden AAA-Protokolle entwickelt, um Dial-In-Benutzern den Zugang zu gestatten. Im Vergleich zu Dial-In-Benutzern stellt die Mobilität bzw. Mobile Grids jedoch neue Anforderungen an das AAA-Modell:

- Die Benutzer müssen eindeutig identifiziert werden. IP-Adressen sind hierfür beispielsweise ungeeignet. Es sollten Network Access Identifier (NAI) [RFC2486] verwendet werden. Der NAI hat die Form Benutzer@Heimatdomäne, wobei sich die heimische Verwaltungszone des Benutzers direkt ablesen lässt.
- AAA-Server müssen skalierbar sein, da die Zahl der mobilen Geräte und damit auch der potentiellen AAA-Benutzer stark anwächst. Ferner fallen bei heutigen Diensten wesentlich mehr Registrierungsanfragen an, als bei ursprünglichen Dial-In Zugangsservern. Zum einen muss bei einem Standortwechsel eine Neuregistrierung erfolgen, andererseits fallen Registrierungen an, um Timeouts bestehender Sessions zu verlängern.
- Die Antwortzeit des AAA-Servers sollte kurz sein, damit mobile Geräte eine Antwort vor Ablauf eines Timeouts erhalten und nicht etwa einen unnötigen Wiederholungsversuch der Anfrage starten müssen.

- Das AAA-Protokoll sollte simpel sein, um die Konstruktion von kostengünstigen Clients zu ermöglichen. Durch den steigenden Bedarf an Mobilität werden viele Zugangspunkte und AAA-Clients im Zuge des Netzausbaus benötigt.

2.4 Das RADIUS-Protokoll

RADIUS (Remote Authentication Dial In User Service) [RFC2865] wurde 1992 von Steve Willens erstmals spezifiziert. Um die Entwicklung eines offenen Standards zu unterstützen, gründete die IETF (Internet Engineering Task Force) 1995 die RADIUS Working Group.
Bis heute hat sich RADIUS als offene Lösung für verschiedenste Dienstarten, von Dial-In bis zu Terminalzugängen, etabliert. Doch die Beschränkungen des Protokolls, auf welche wir später noch eingehen werden, lassen den Wunsch nach einem Nachfolger aufkommen.

2.4.1 Einführung in RADIUS

RADIUS ist nicht nur ein Einwahlprotokoll, sondern beschreibt eine AAA-Architektur, welche AAA-Dienste verwaltungszonenübergreifend ermöglicht (vgl. hierzu [HA04]).
Das RADIUS-Protokoll entspricht dem im vorangehenden Kapitel erwähnten Client-Server-Modell. Der Client, im Allgemeinen ein NAS, sendet dem Server eine Authentifizierungsanforderung. Entweder beantwortet der Server diese Anfrage sofort oder er leitet sie in der Rolle eines Proxies an einen anderen Server weiter. Dabei agiert er aus der Sicht eines anderen Servers als Client. Den weitergeleiteten Nachrichten kann der Proxy ein Attribut anhängen, welches die Nachrichten als Weiterleitungen identifiziert.
Dabei bestehen alle RADIUS-Nachrichten aus einem Typbezeichner (z.B. Access-Request, Access-Reject...), einer Kennung, welche die Zuordnung von Anfragen und Antworten ermöglicht, der Länge der Nachricht, einem 16 Byte Datum, welches die Echtheit der Nachricht gewährleistet und mindestens einem Attribut. Attribute enthalten Daten, welche zur Authentifizierung, Autorisierung oder Abrechnung benötigt werden. Als Beispiel sei hier eine Benutzerkennung in Form eines Network Access Identifiers genannt. Die Attribute oder auch Attribute Value Pairs (AVP) wurden eingeführt, um flexibel genug zu sein, unterschiedliche Daten in den Nachrichten aufzunehmen und zu übertragen.
Hier soll kurz ihr Aufbau beschrieben werden:

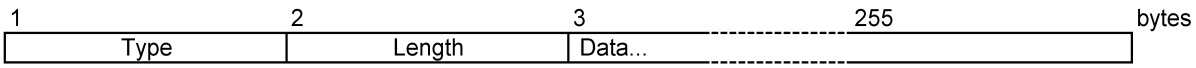

Abb. 4: RADIUS Attribute Value Pair (AVP)

Jedes AVP besteht aus dem Typbezeichner (Type), der Länge des AVP in Bytes (Length) und einem Datum (Data). Die Länge des AVP und des Typbezeichners sind je ein Byte gross. Dadurch können maximal 256 Typen unterschieden werden. Das Datum darf dabei maximal 253 Byte lang sein.
Die Nachrichten zwischen Client und Server werden per UDP übertragen. Dabei verzichtet man bewusst auf die Transportsicherheit von TCP. Der Verzicht wird damit begründet, dass die TCP-Mechanismen, welche den sicheren Transport gewährleisten, nicht den Anforderungen von RADIUS genügen. Ist beispielsweise ein RADIUS-Server nicht erreichbar, so soll der Client die Anfrage nach einigen wiederholt fehlgeschlagenen Versuchen sofort einem alternativen Server zustellen.

2.4.2 Einsatz von RADIUS

Mit der Registrierungsanfrage wird durch den RADIUS-Client eine Authentifizierungsanfrage an den Server gesendet. Sollte innerhalb einer bestimmten Frist keine Antwort vom Server vorliegen, kann der Client die Anfrage wiederholen oder einem anderen Server schicken.

Empfängt der Server eine gültige Anfrage, so sucht er in einer Benutzerdatenbank nach dem entsprechenden Datensatz. Ist der Benutzer unbekannt, so kann die Anfrage an einen weiteren Server weitergeleitet werden.
In der Datenbank findet der Server Bedingungen, welche der Benutzer erfüllen muss, um authentifiziert zu werden. Bedingungen können zum Beispiel ein korrektes Passwort sein oder auch eine Beschränkung auf ein bestimmtes Endgerät, über welche der Benutzer Zugang erhält.
Erfüllt der Benutzer alle Bedingungen, so bestätigt der RADIUS-Server die Authentizität des Benutzers mit einer Access-Accept-Meldung. Diese Nachricht enthält weitere Attribute, welche der Client benötigt, um dem Benutzer den Zugang zu den Diensten zu ermöglichen.
Kann der Benutzer die Bedingungen nicht erfüllen, so erhält der Client (oder NAS) eine Access-Reject-Meldung. Der Benutzer bekommt in diesem Fall keinen Zugang zu den gewünschten Dienstleistungen.

2.4.3 Schwächen des RADIUS-Protokolls

Nachdem RADIUS-Implementierungen seit über zehn Jahren im Einsatz sind, zeigen sich Schwächen des Protokolls, die eine Nutzung unter gesteigerten Anforderungen in punkto Transportsicherheit, Skalierbarkeit und Flexibilität in Frage stellen [CA02, EK00]. Einige dieser Problemstellungen sollen hier kurz erläutert werden:

- Der Datentransport per UDP stellt sich als Schwachstelle heraus. RADIUS beschreibt, dass auf ausbleibende Antworten eines Servers mit Wiederholungsversuchen nach einem abgelaufenen Timeout zu reagieren ist. Doch die Anzahl der Wiederholungen als auch die Zeitintervalle sind nicht im Protokoll festgelegt, so dass es zwischen verschiedenen Implementierungen zu Inkompatibilitäten kommen kann.
- Der Server sendet dem Client keine Status- oder Fehlermeldungen. Somit ist weder in der Transportschicht noch auf höherer Ebene eine Fehlererkennung implementiert. Der Client kann somit lediglich anhand der Timeouts erkennen, ob der Server erreichbar ist. Muss der Server zur Bearbeitung der Anfrage auf langsame Systeme zugreifen, so interpretiert der Client eine lange Antwortzeit eventuell als einen Fehler (der Datenübertragung oder des Serves) und reagiert unangemessen mit einer Wiederholung der Anfrage.
  Tritt ein Übertragungsfehler zwischen einem Proxy und einem Server auf, so reagiert nicht der Proxy mit einem neuen Versuch, sondern der Client (bzw. NAS). Dabei wird ein unnöig hoher Datenverkehr in Kauf genommen.
- Bei ungültigen Anfragen, sei es wegen einer unkorrekter Länge oder fehlerhaftem Typ der Nachricht, muss der Server die Nachricht (bzw. Anfrage) ignorieren. Insbesondere ist keine Fehlermeldung an den Client vorgesehen. Durch die implizite Annahme eines Übertragungsfehlers versucht der Client dadurch mehrfach die ungültigen Daten zu versenden.
- RADIUS ist nicht modular erweiterbar. Zwar lassen sich weitere Attribute definieren, doch sind insgesamt nur deren 256 verschiedene erlaubt. Implementieren verschiedene Hersteller neue Funktionen und Attribute, so kann es zu Überschneidungen in dem begrenzten Nummernraum und damit Inkompatibilitäten kommen.
  Ausserdem ist die Länge eines AVP, durch das lediglich ein Byte grosse Längenfeld, auf 256 Bytes beschränkt. Sollen grössere Nachrichten versendet werden, so müssen sie auf mehrere AVPs verteilt werden.
- Da RADIUS einem Client-Server-Modell entspricht, ist keine Möglichkeit vorgesehen, dass der Server eine Kommunikation mit dem Client beginnt. Besonders im Bereich Abrechnung ist dieses Verfahren aber üblich, um Informationen anzufordern oder den Client aufzufordern, die Session eines Benutzers zu beenden.

- Die Daten in den Nachrichten sind byteweise ausgerichtet. Da aktuelle Rechnerarchitekturen nur auf Daten, die an 32 Bit-Grenzen ausgerichtet sind, besonders schnell zugreifen, verschenkt man hier Leistung.

2.5 Das Diameter-Protokoll

Um den Problemen des RADIUS-Protokolls zu begegnen, wurde bereits 1996 von Pat Calhoun mit der Entwicklung von Diameter begonnen. Zusammen mit Firmen wie Bell, Cisco, Sun oder Microsoft wurde das Ziel angestrebt, einen Nachfolger für RADIUS zu finden. Dabei wurde besonders auf die Kompatibilität mit RADIUS Wert gelegt, um eine schrittweise Migration zu Diameter zu erleichtern.
Diameter ist in einen allgemeinen Teil, dem Base Protocol, und in anwendungsspezifische Erweiterungen, den so genannten Applications, unterteilt. Das Base Protocol beschreibt den Transport der Daten, Fehlermeldungen und grundlegende Session- und Abrechnungs-Funktionalität. Die Applications stellen dann die Unterstützung für bestimmte Technologien, wie zum Beispiel Mobile Grids oder Dial-In Zugänge, zur Verfügung. Durch das Konzept der Applications ist Diameter modular erweiterbar, gegenwärtig sind jedoch nur Applikationen für Mobile IPv4 und Dial-In Network Access Server spezifiziert.

2.5.1 Diameter Base Protocol

Das Diameter Base Protocol definiert den Aufbau und Transport der Nachrichten, die Fehlermeldungen und weitere Grundlagen, auf welche Diameter Applikationen aufbauen können [RFC3588].

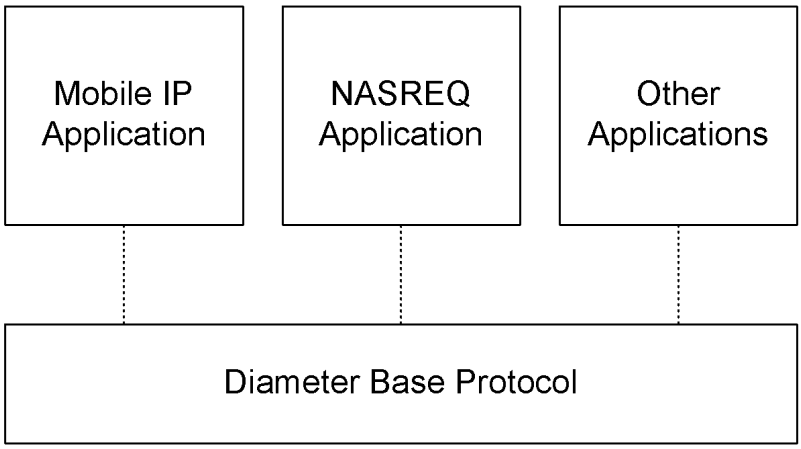

Abb. 5: Diameter Protokoll

Zusätzlich zu Client und Server definiert das Diameter Protokoll noch Proxy-, Relay-, Redirect-, und Translation-Agents:

Relay Agents leiten Nachrichten anhand von Routing AVPs in den Nachrichten sowie anhand von Routingtabellen weiter. Dabei können Relay-Agents der Nachricht lediglich Routing-Informationen hinzufügen und arbeiten sonst transparent.

Proxy-Agents routen ebenso Diameter-Nachrichten, können jedoch ungültige Anfragen mit Fehlermeldungen und Access-Reject-Meldungen abfangen. Proxies müssen also die Nachricht auswerten und die damit verwendeten Applikationen auch unterstützen.

Redirect-Agents senden dem Client auf eine Anfrage hin die Adresse des Zielservers, so dass eine direkte Kommunikation zwischen Client und gesuchtem Server möglich ist.

Translation-Agents stellen Gateways dar, welche die AAA-Protokollnachrichten übersetzen. Typischerweise geschieht dies zwischen RADIUS- und Diameter- Implementierungen.

Somit wird die Migration auf Diameter vereinfacht indem Server schnell ausgetauscht und die grosse Zahl an Clients langsamer und schrittweise umgerüstet werden können.

Im Gegensatz zu RADIUS wurde das Client-Server-Modell zugunsten eines Peer-to-Peer-Modells in Diameter aufgegeben. Der Server kann dem Client gleichberechtigt ohne vorherige Anforderung Nachrichten senden, womit einige Aspekte im Sessionmanagement (Terminierung von Sitzungen) und in der Abrechnung (der Server kann Accounting-Daten vom Client anfordern) erleichtert werden.

Wie in RADIUS werden in den Nachrichten alle Informationen als AVPs versendet. Die Beschränkung auf 256 verschiedene AVPs wurde in Diameter auf 2^{32} erhöht. Ebenso dürfen AVPs bis zu 2^{24} Bytes gross sein.

Die Nachrichten werden bei Diameter auch nicht mehr per UDP, sondern mit dem Stream Control Transmmission Protocol (SCTP) [RFC2960] oder TCP übertragen. SCTP ist ein relativ junges Transportprotokoll, dessen Charakteristika hier kurz erwähnt werden sollte, da sie stark zur Fehlervermeidung auf Transportebene beiträgt. Wie TCP sorgt auch SCTP für einen sicheren Transport ohne Datenverlust oder doppelt angekommene Pakete. SCTP unterstützt jedoch im Gegensatz zu TCP nicht nur einen, sondern mehrere Datenströme zwischen den Kommunikationspartnern. In jedem Datenstrom werden die Daten in der richtigen Reihenfolge und unabhängig von anderen Strömen übertragen, so dass Fehler in anderen Strömen keine Auswirkung, wie Verbindungsabbruch oder Verzögerung, auf den betrachteten Kommunikationskanal haben. Durch die Unterstützung mehrerer Datenströme ist es möglich, Anfragen an den Server parallel abzusenden.

Bei Diameter ist aber nicht nur auf der Transportschicht eine Fehlererkennung möglich, sondern auch auf Anwendungsebene. So muss, bis auf wenige Ausnahmen, jede Anfrage vom Server beantwortet werden. Dazu enthalten alle Antwortnachrichten ein Result-Code-AVP, in welchem ein Fehlerstatus oder ein Erfolgsstatus übergeben wird.

Durch die Fehlermeldungen kann jetzt der Client oder Proxy angemessen reagieren. Obliegt bei RADIUS die Reaktion auf Fehler allein beim Client, so kann bei Diameter auch der Proxy entsprechend handeln. Bei einem Serverausfall muss nicht darauf gewartet werden, dass das Timeout des Clients abläuft und ein Wiederholungsversuch beginnt. Vielmehr kann der Proxy selbst eine Wiederübertragung veranlassen oder die Nachricht gleich an einen alternativen Server senden.

Durch die Erfahrung mit RADIUS konnten viele Schwachstellen aufgedeckt werden, welche im Diameter Base Protocol berücksichtigt wurden, so dass nun ein Ressourcen schonendes sowie sicheres Protokoll zur Verfügung steht.

2.5.2 Diameter Nachrichten

Diameter Nachrichten bestehen, wie auch RADIUS Nachrichten, aus einem Header gefolgt von einer Menge von Attribure Value Pairs (AVPs). Das erste AVP einer Nachricht ist stehts das Command AVP, welches das Kommando der Nachricht beinhaltet. Anschliessend folgen weitere AVPs mit Kommando-spezifischen Feldern [RFC3588].

Der Nachrichten-Header ist bei Diameter wie folgt aufgebaut:

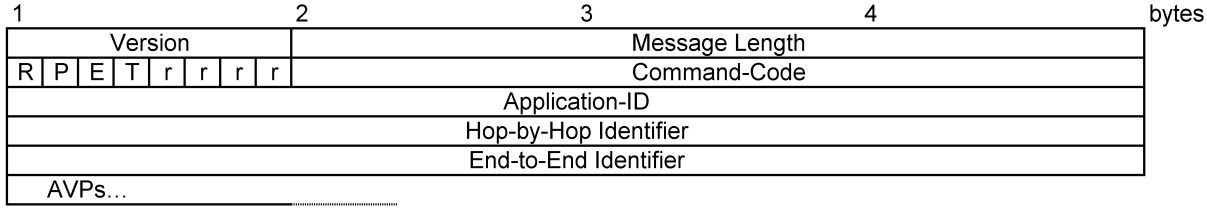

Abb. 6: Diameter Nachrichten-Header

Das Versionsfeld muss stehts auf 1 gesetzt sein, was in Bezug zu Version 1 von Diameter steht. Die Nachrichtenlänge beinhaltet die Länge der gesamten Nachricht inklusive dem Header in Bytes.
Das fünfte Octet beinhaltet die Command-Flags welche wie folgt definiert sind:

R: Request	Falls das Flag gesetzt ist, ist die Nachricht eine Anfrage sonst eine Antwort
P: Proxiable	Falls das Flag gesetzt ist, darf die Nachricht von einem Proxy, Relay oder Redirect verarbeitet werden
E: Error	Falls das Flag gesetzt ist, beinhaltet die Nachricht einen Protokollfehler
T: Re-Transmitted	Falls das Flag gesetzt ist, handelt es sich bei der Nachricht möglicherweise um eine erneute Übertragung
r: reserved	Reservierte Flags, welche auf null gesetzt werden sollten

Der Kommando-Code ist ein Feld von drei Bytes welches das mit der Nachricht verknüpfte Kommando definiert. Die Verwaltung dieses 3-Byte Adressraumes liegt bei der IANA [IANA05]. Kommando-Codes für Testzwecke wurden dabei im Base Protocol vorgesehen. Die Applikations-ID definiert die Applikation für welche die Nachricht bestimmt ist. Der Hop-by-Hop identifier wird verwendet, um die Zusammenhörigkeit von Anfragen und Antworten zu bestimmen. Zusätlich hilft der End-to-End identifier, mehrfach vorhandene Nachrichten heruszufiltern. Zum Schluss der Nachricht folgen die Daten in Form von AVPs.

2.5.3 Diameter Kommandos

Diameter Kommandos definieren den Inhalt einer Nachricht in Form einer AVP-Liste. Kommandos können dabei beispielsweise eine Abrechnungsanfrage sein. Das Diameter Base Protocol definiert die in der untenstehenden Tabelle aufgeführten Standardkommandos [RFC3588].

Tabelle 1: Kommandos des Diameter Base Protocol

Kommando-Name	**Abkürzung**	**Kommando-Code**
RADIUS compatibility codes	-	0-255
Abort-Session-Request	ASR	274
Abort-Session-Answer	ASA	274
Accounting-Request	ACR	271
Accounting-Answer	ACA	271
Capabilities-Exchange-Request	CER	257
Capabilities-Exchange-Answe	CEA	257
Device-Watchdog-Request	DWR	280
Device-Watchdog-Answer	DWA	280
Disconnect-Peer-Request	DPR	282
Disconnect-Peer-Answer	DPA	282
Re-Auth-Request	RAR	258
Re-Auth-Answer	RAA	258
Session-Termination-Request	STR	275
Session-Terminaton-Answer	STA	275

Zusätzliche Kommandos können definiert werden und sind meistens applikationsspezifisch. Auch die im Diameter Base Protokoll beschriebenen Kommandos müssen in den meisten Fällen applikationsspezifisch angepasst werden. Je nach Kommando ist Server- bzw. Clientseitig eine entsprechende Reaktion bzw. ein entsprechender Verarbeitungschitt definiert und implementiert. Die Verwaltung der Kommando-Codes obliegt dabei der IANA welche eine Liste sämtlicher Codes jeweils auf dem Internet publiziert [AAAP05].

2.5.4 Diameter Attribute Value Pairs

Diameter verwendet, wie bereits sein Vorgänger RADIUS, AVPs zur Informationsübermittlung. Die AVPs beinhalten dabei spezifische Informationen für die Authentifizierung, Autorisierung, Abrechnung aber auch das Routing und die Sicherheit. Ebenso werden auch Konfigurationsdetails für Anfragen und Antworten mit AVPs übermittelt. Jedes AVP beinhaltet nebst den eingentlichen Daten (Beispielsweise einen Hostnamen) auch einen Header, welcher wie folgt aufgebaut ist [RFC3588]:

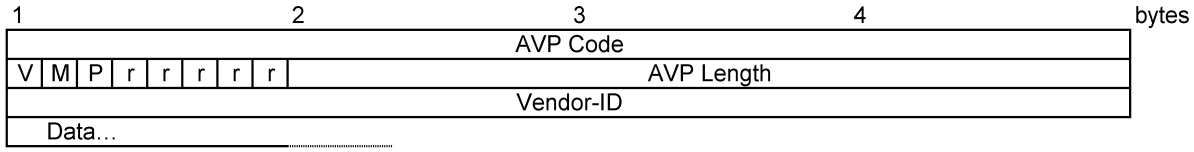

Abb. 7: Diameter Attribute Value Pair

Die ersten vier Bytes beinhalten den Code des AVP. Die Codes, zusammen mit der optionalen Vendor-ID, identifizieren ein AVP einmalig. AVP-Codes bis 255 sind aus Kompatibilitätsgründen für RADIUS reserviert. Die darüberliegenden Codes werden wie bereits die Kommando-Codes von der IANA verwaltet und können unter [AAAP05] jeweils abgerufen werden. Das fünfte Octet beinhaltet analog zum Nachrichtenheader wiederum Flags welche wie folgt definiert sind:

V: Vendor	Falls das Flag gesetzt ist, ist im Header eine Vendor-ID untergebracht
M: Mandatory	Falls das Flag gesetzt ist, muss der Empfänger die AVP unterstützen oder das AVP zurückweisen. Ist das Flag nicht gesetzt, gilt die Nachricht als informativ.
P: Encryption	Falls das Flag gesetzt ist, wird die Benötigung von Verschlüsselung für eine Ende-zu-Ende Sicherheit signalisiert.
r: reserved	Reservierte Flags, welche auf null gesetzt werden sollten

Die AVP Länge definiert auch in diesem Fall die Länge des gesamten AVP inklusive dem Header in Bytes. Die Vendor-ID ist im Header nur dann untergebracht, falls das V-Flag gesetzt wurde. Dieses Feld beinhaltet eine Vendor-ID welche wiederum auch von der IANA verwaltet werden. Abschliessend folgen die eigentlichen Daten welche in einem AVP-spezifischen Datenformat vorliegen. Das Diameter Base Protokoll kennt hierfür diverse Basisdatentypen wie auch abgeleitete Datentypen, welche zu einem späteren Zeitpunkt noch genauer vorgestellt werden.

3 Das OpenDiameter Projekt

3.1 Einführung in das OpenDiameter Projekt

3.1.1 OpenDiameter Framework

OpenDiameter ist ein OpenSource-Projekt welches das Diameter Basisprotokoll implementiert [WU04]. Dabei stellt die Bibliothek eine Standard-Implementierung des Diameter Basisproto-

kolls sowie einiger Diameter Applikationen zur Verfügung. Die Architektur basiert dabei wesentlich auf den Design Patterns welche im Adaptive Communication Environment (ACE) [ACE05] verwendet werden. Nebst den ACE-basierten Patterns wird auch die Betriebssystemabstraktions-Schicht der ACE-Bibliothek verwendet.
Die aktuelle Architektur von OpenDiameter ist aufgeteilt in vier logische Module:

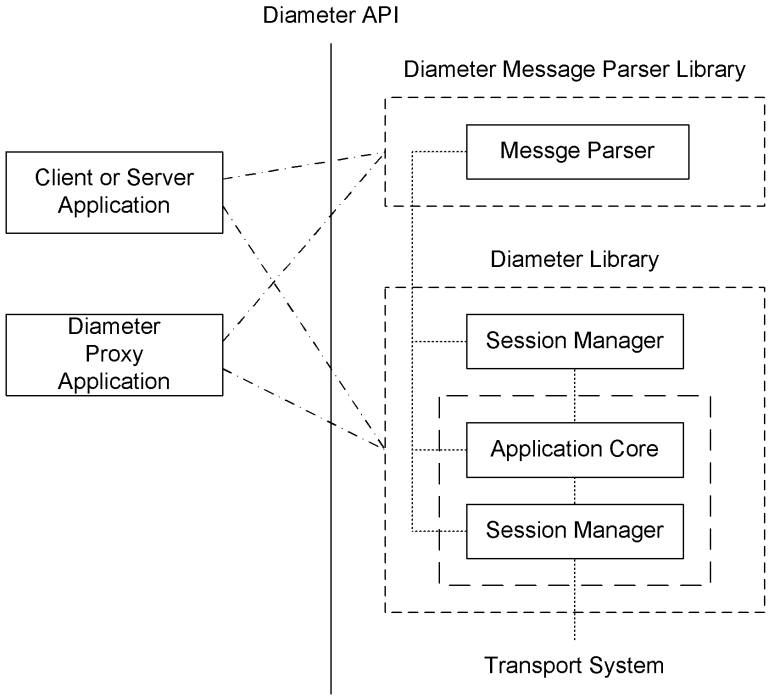

Abb. 8: OpenDiameter Bibliothek Module

Application Core
Der Application Core ist die zentrale Instanz für die globalen Daten und wird verwendet um die Dienste der gesamten Diameter-Bibliothek zu initialisieren und zu beenden. Die globalen Daten beinhalten dabei sämtliche Informationen, welche in den XML-Konfigurationsdateien abgelegt sind.

Session Manager
Der Session Manager ist verantwortlich für die Speicherung und Verwaltung von Diameter-Sessions der Client- bzw. Server-Authentifikationsapplikationen welche die Diameter-Bibliothek verwenden.

Transport Manager
Der Transport Manager ist ein Integrationspartner für die Application Core. Die Hauptverantwortung des Transport Managers ist dabei die Verwaltung der Verbindungen mit anderen Diameter-Knoten sowie das Routing und der Nachrichtenversand von Diameter-Nachrichten.

Message Parser
Der Message Parser wird verwendet um die Diameter-Nachrichten wieder aufzuteilen in Header und Nutzdaten wobei als Nutzdaten hier die Diameter AVPs gemeint sind.
Der libdiamparser ist als eine eigenständige von libdiameter getrennte Bibliothek implementiert. Beide Bibliotheken werden für die Client- wie auch Server-Authentifikation verwendet.

Die Diameter-Funktionalität wird dabei durch mehrere Threads bereitgestellt. Dabei verwendet das OpenDiameter-Framework auch die ACE-Funktionalitäten in Bezug auf die Threads.

3.1.2 Adaptive Communication Environment (ACE)

Das ACE ist ein in C++ geschriebenes und weit verbreitetes objekt-orientiertes Toolkit [ACE05]. Es stellt multiple Zugriffsmöglichkeiten sowie Kommunikationspattern für Kommunikationssoftware zur Verfügung. ACE beinhaltet dabei viele Komponenten, welche die Entwicklung von Kommunikationssoftware erleichtern und erhöht dabei die Flexibilität, Effizienz, Anpassbarkeit und Portabilität der Applikationen.

3.1.3 Xerces

Xerces-C++ ist ein Parser zur Validierung von XML [XERCES05]. Es steht eine gemeinsame Bibliothek für Parsing, Generierung, Manipulierung und Validierung von XML-Dokumenten zur Verfügung. Der Parser bietet dabei eine hohe Performance, ist modular und skalierbar.
Xerces wird dank seinen grossen Möglichkeiten im Bereich Generierung und Validierung in vielen Bereichen erfolgreich eingesetzt.
OpenDiameter verwendet Xerces um die Konfigurations-Dateien zu Parsen.

3.2 Installation von OpenDiameter

3.2.1 Allgemeine Hinweise

Die Installation von OpenDiameter hat sich im Zuge dieser Semesterarbeit als nicht ganz trivial herausgestellt. Daher wird auf den folgenden Seiten das Installationsprozedere des OpenDiameter Frameworks detailliert beschrieben.
Als Betriebssystem verwenden wird dabei Linux. Linux bietet zahlreiche Vorteile gegenüber anderen Betriebsystemen und wird wohl auch später im produktiven Einsatz zum Zuge kommen.
Die Installation von OpenDiameter auf ein Linux Betriebssystem sollte an sich keine grosse Probleme mit sich bringen. Trotzdem traten einige Probleme auf, wodurch es auch nach mehrerern Stunden Arbeit nicht möglich war, das OpenDiameter Framework auf Debian GNU Linux, weder auf stable (woody) noch unstable (sarge), zu installieren bzw. kompilieren. In der Folge fiel daher der Entscheid auf SUSE LINUX Professional 9.3 von Novell.
An dieser Stelle sei auch auf die kurze Installationsanleitung auf der Webseite des OpenDiameter Projektes [OD05] hingewiesen.

3.2.2 Installation von Linux (SUSE LINUX Professional 9.3)

Für die Installation von SUSE LINUX Professional 9.3 (anschliessend als SUSE bezeichnet) werden die original Installationsmedien des Herstellers benötigt. OpenDiameter stellt dabei keine besonderen Anforderungen an das System, wonach die Standardinstallation von SUSE durchführt werden kann. Allfällig zusätzlich benötigte Komponenten können nach der erfolgreichen Grundinstallation nachinstalliert werden. Die Grundinstallation an sich verläuft sehr einfach.

Allfällige Supporthinweise für die Installation von SUSE können direkt der Homepage [SUSE05] entnommen werden.

Abb. 9: Installation von SUSE

Nach der erfolgreichen Installation und erfolgtem Login auf der grafischen Oberfläche muss das Konfigurationstool YaST gestartet werden. Unter Software - Software installieren, werden nachträglich noch die Pakete autoconf, automake, openssl-devel, gcc und gcc-c++ mit den entsprechenden Abhängigkeiten auf dem System installiert. Anschliessend muss nach dem erfolgreihen Login (als root) auf der Konsole im Verzeichnis /root ein Unterverzeichnis für OpenDiameter erstellt werden.

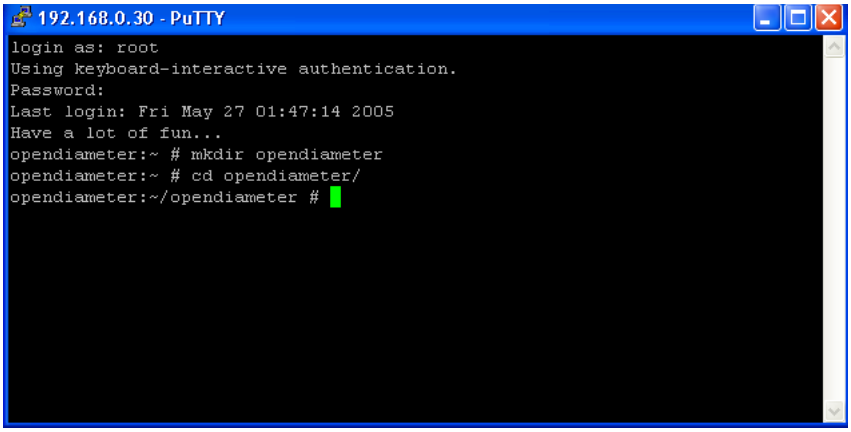

Abb. 10: Login nach erfolgreicher Installation

3.2.3 Installation von XERCES C++ XML Parser (XERCES-C++ 2.6.0)

Die Quellen in der Version 2.6.0 können sehr einfach über die Apache Webseite [XERCES05] heruntergeladen werden. Die Installation läuft relativ einfach ab:

1) Download der Quellen mit wget
```
wget http://mirror.switch.ch/mirror/apache/dist/xml/xerces-c/xerces-c-src_2_6_0.tar.gz
```
2) Entpacken der tar.gz-Datei
```
tar -xzvf xerces-c-src_2_6_0.tar.gz
```
3) Speicherung des root-Verzeichnis von XERCES als Variable
```
export XERCESCROOT=/root/opendiameter/xerces-c-src_2_6_0
```
4) Wechsel in das Source-Verzeichnis
```
cd $XERCESROOT/src/xercesc
```
4) Ausführen von autoconf und configure
```
autoconf
./configure
```

5) Nach erfolgreicher Konfiguration müssen die Quellen mit gmake kompiliert und installiert werden

```
gmake
gmake install
```

3.2.4 Installation der ACE library (ACE 5.4)

Anschliessend müssen die Quellen der ACE library heruntergeladen, kompiliert und installiert werden:

1) Download der Quellen mit wget
```
wget http://deuce.doc.wustl.edu/ACE-5.4.tar.gz
```
2) Entpacken der tar.gz-Datei
```
tar -xzvf ACE-5.4.tar.gz
```
3) Wechsel in das Source-Verzeichnis
```
cd ACE_wrappers/
```
4) Erstellen eines Verzeichnis “objdir”
```
mkdir objdir
```
5) Wechsel in das neu erstellte Verzeichnis
```
cd objdir
```
6) Ausführen von configure
```
../configure
```
7) Nach erfolgreicher Konfiguration müssen die Quellen mit make kompiliert und installiert werden
```
make
make install
```

3.2.5 Installation der Boost library (Boost 1.32)

Die Installation der Boost library läuft ebenso einfach von der Hand:

1) Download der Quellen mit wget
```
wget http://switch.dl.sourceforge.net/sourceforge/boost/boost_1_32_0.tar.gz
```
2) Entpacken der tar.gz-Datei
```
tar -xzvf boost_1_32_0.tar.gz
```
3) Wechsel in das jam_src-Verzeichnis
```
cd boost_1_32_0/tools/build/jam_src/
```
4) Tools kompilieren
```
./build.sh
```
5) Wechsel in das bin.linuxx86-Verzeichnis
```
cd bin.linuxx86/
```
6) Soeben erstellte Tools nach /usr/bin kopieren
```
cp * /usr/bin
```
7) Wechsel in das root-Verzeichnis der Boost library
```
cd ../../../../
```
8) Anschliessend müssen die Binaries mittels bjam erstellt und installiert werden
```
bjam * “-sTools-gcc” install
```

3.2.6 Installation von OpenDiameter (OpenDiameter 1.0.7-f)

Zum Abschluss folgt die Installation des OpenDiameter Framework:

1) Download der Quellen mit wget
```
wget http://switch.dl.sourceforge.net/sourceforge/diameter/opendiameter-1.0.7-f.tar.gz
```
2) Entpacken der tar.gz-Datei
```
tar -xzvf opendiameter-1.0.7-f.tar.gz
```
3) Wechsel in das Source-Verzeichnis
```
cd opendiameter-1.0.7-f/
```
4) Registrierung der Root-Verzeichnisse der libraries als Variablen
```
export XERCESCROOT=/root/opendiameter/xerces-c-src_2_6_0/
export ACE_ROOT=/root/opendiameter/ACE_wrappers/
export BOOST_ROOT=/root/opendiameter/boost_1_32_0/
```

5) Ausführen von configure
```
./configure
```
6) Nach erfolgreicher Konfiguration müssen die Quellen mit make kompiliert und installiert werden
```
make
make install
```

4 Abrechnungs-Modul für einen AAA-Server

4.1 Anforderungen

Das neue Einsatzgebiet der AAA-Architektur für Mobile Grids bringt auch neue Anforderungen mit sich. So muss beispielsweise von der klassischen Client-Server-Architektur wie sie in RADIUS verwendet wird zu einer Peer-to-Peer-Architektur, welche Diameter verwendet, übergegangen werden. Zudem fordert der Einsatzbereich von Mobile Grids nach neuen Abrechnungs-Parameter wie beispielsweise Speicherkonsum, Prozessorkonsum oder Anzahl verwendeter Knoten.
Um diesen neuen Anforderungen gerecht zu werden, wird in der vorliegenden Semesterarbeit das neue AAA-Protokoll Diameter verwendet. Wie bereits in Kapitel 2.5 beschrieben, besteht die Diameter-Architektur aus einem Basisprotokoll und anwendungsspezifischen Applikationen. Nebst der Definition von neuen Abrechnungsparameter (und der daraus resultierenden neuen AVPs) sind auch die Diameter-Kommandos anzupassen.

4.2 Übersicht

In einem ersten Teil werden die Mobile-Grid spezifischen Abrechnugnsparameter definiert und anschliessend als zusätzliche AVPs modelliert. In einem zweiten Teil wird die erforderliche Erweiterung der Diameter-Kommandos aufgezeigt und beschrieben. Dannach wird die Softwarearchitektur des Abrechnungs-Moduls beschrieben und die Designentscheide erläutert. Der Abschluss dieses Kapitels bildet eine kurze Installationsanleitung sowie eine Auflistung noch offener Probleme.

4.3 Mobile-Grids spezifische Abrechnungs-Parameter

4.3.1 Festlegung der neuen Abrechnungs-Parameter

Der erweiterte Einsatzbereich für den Bereich Mobile Grids erfordert auch zusätzliche Abrechnungs-Parameter wie beispielweise CPU-, Speicher- oder Hauptspeicher-Konsum. Das Global Grid Forum hat bezüglich dieser Grid-spezifischen Parameter ein Memo [MA04] herausgegeben, welches diese Parameter sehr gut festzulegen vermag.
Basierend auf diesem Memo ergeben sich die in Tabelle 2 dargestellten Grid-spezifischen Abrechnungs-Parameter welche in einem Abrechnungs-Modul für Mobile Grids unterstützt werden sollten. Die Messgrössen beziehen sich dabei jeweils auf die Dauer einer Session.

Tabelle 2: Mobile-Grids spezifische Abrechnungs-Parameter

Parameter	**Datentyp**	**Beschrieb**
CPUUsage	integer	Verwendete CPU-Zeit summiert über alle Prozesse in Sekunden
DiskUsage	integer	Verwendeter Diskspeicher in Kilobyte
EndTime	times	Zeitpunkt des Ende einer Session
HostName	string	Systemhostname auf welchem der Job läuft

Tabelle 2: Mobile-Grids spezifische Abrechnungs-Parameter

Parameter	**Datentyp**	**Beschrieb**
JobName	string	Name des verabeitenden Job
MachineName	string	Deskriptiver Name der Maschine auf welcher der Job läuft
MemoryUsage	integer	Verwendeter physikalischer Speicher in Kilobyte
NetworkUsage	integer	Anzahl gesendete Kilobytes über das Netzwerkinterface
NodeCount	integer	Anzahl verwendete Nodes
ProcessId	integer	Prozessnummer des Job
ProcessorCount	integer	Anzahl der verwendeten Prozessoren (meist identisch zur Anzahl der physikalischen CPUs)
QueueName	string	Name der Warteschlange von welcher der Job ausgeführt oder übermittelt wurde
ScratchUsage	integer	Verwendeter Scratchspeicher in Kilobyte
ServiceLevelQuality	string	QOS welche während des Ressourcenkonsums eingesetzt wurde
StartTime	times	Zeipunkt des Starts einer Session
Status	integer	Status nach Beendigung eines Jobs: 1: aborted - Job wurde durch eine Regel oder Benutzer abgebrochen 2: completed - Job wurde erfolgreich beendet 3: failed - Job brach ab ohne externen Einfluss 4: held - Job wurde angehalten 5: queued - Job ist in der Warteschlange 6: started - Job wurde gestartet 7: suspended - Job wurde abgebrochen (Abbruchvorgang am laufen)
SubmitHost	string	Systemhostname von welchem der Job übermittelt wurde
SwapUsage	integer	Verwendeter Swapspeicher
TempUsage	integer	Verwendeter temporärer Diskspeicher in Kilobyte

4.3.2 Mobile-Grids spezifische AVPs

Basierend auf der in Tabelle 2 aufgeführten Parameter, müssen für den Einsatz in OpenDiameter neue AVPs definiert werden. Sinvollerweise würde man für die zusätzlichen AVPs, wie auch die Applikation selbst, eine Vendor-ID einführen, um eine von der IANA unabhängige AVP-Nummerierung verwenden zu können (vgl. auch Seite 177 in [RFC3588]). Leider erlaubt die aktuelle Version des OpenDiameter Frameworks jedoch den Einsatz von Vendor-IDs noch nicht. Aufgrund der Geschlossenheit unseres Systemes, kann aber momentan noch beruhigt auf den Einsatz und die Beantragung einer Vendor-ID bei der IANA verzichten werden. Es wird deshalb keine Vendor-ID (bzw. die Vendor-ID 0) im vorliegenden Prototyp verwendet. Um jedoch Überschneidungen mit bestehenden AVPs zu vermeinden, werden für die neuen Abrechnungs-AVPs AVP-Nummern von 10000 bis 20000 verwendet.
Die bereits in Tabelle 2 aufgeführten Datentypen müssen dabei ebenso in den AVPs umgesetzt werden. Diameter kennt dabei bereits diverse AVP Datentypen welche in den folgenden zwei Abschnitten kurz erläutert werden.

AVP Basisdatentypen

In Diameter sind mehrere Basisdatentypen (Tabelle 3) definiert, in welchen die Daten eines AVP vorliegen können.

Tabelle 3: AVP Basisdatentypen

Datentyp	**Beschrieb**
OctetString	Daten variabler Länge
Integer32	32bit-Ganzzahl mit Vorzeichen
Integer64	64bit-Ganzzahl mit Vorzeichen
Unsigned32	32bit-Ganzzahl ohne Vorzeichen
Unsigned64	64bit-Ganzzahl ohne Vorzeichen
Float32	32bit-Fliesskommawert
Float64	64bit-Fliesskommawert
Grouped	Die Daten sind eine Sequenz von AVPs

AVP abgeleitete Datentypen

Weiterhin können aus den AVP Basisdatentypen auch so genannte abgeleitete Datentypen erzeugt werden. In Tabelle 4 sind einige dieser abgeleiteten Datentypen aufgelistet.

Tabelle 4: AVP abgeleitete Datentypen

Datentyp	**Basisdatentyp**	**Beschrieb**
Address	OctetString	32bit IPv4 bzw. 128bit IPv6 Adresse wobei die ersten zwei Octets den Adresstyp definieren
Time	OctetString	Vier Octet lange Zeitangabe als NTP Timestamp, wobei die Anzahl Sekunden seit 0Uhr am 1. Januar 1900 in UTC (Coordinated Universal Time) angegeben werden
UTF8String	OctetString	UFT-8 codierter lesbarer Text in beliebiger Länge (Octets, nicht Zeichen)

In der Folge werden nun die benötigten AVPs auf Basis von Tabelle 2 definiert.

Accounting-CPUUsage AVP

Das Accounting-CPUUsage AVP (AVP-Nummer 10000) ist vom Typ Unsigned32 und beinhaltet die verwendete CPU-Zeit in Sekunden summiert über alle Prozesse.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-DiskUsage AVP

Das Accounting-DiskUsage AVP (AVP-Nummer 10001) ist vom Typ Unsigned64 und beinhaltet den verwendeten Diskspeicher in Kilobyte.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 16 Byte.

Accounting-EndTime AVP
Das Accounting-EndTime AVP (AVP-Nummer 10002) ist vom abgeleiteten Typ Time und beinhaltet den Zeitpunkt des Endes einer Session.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-HostName AVP
Das Accounting-HostName AVP (AVP-Nummer 10003) ist vom abgeleiteten Typ UTF8String und beinhaltet den Systemhostname auf welchem der Job läuft (die maximale Länge des Systemhostnamens beträgt 20-ASCII-Zeichen).
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 28 Byte.

Accounting-JobName AVP
Das Accounting-JobName AVP (AVP-Nummer 10004) ist vom abgeleiteten Typ UTF8String und beinhaltet den Namen des verarbeitenden Job (die maximale Länge des Jobnamens beträgt 20-ASCII-Zeichen).
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 28 Byte.

Accounting-MachineName AVP
Das Accounting-MachineName AVP (AVP-Nummer 10005) ist vom abgeleiteten Typ UTF8String und beinhaltet den deskriptiven Namen der Maschine auf welcher der Job läuft (die maximale Länge des Namens beträgt 20-ASCII-Zeichen).
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 28 Byte.

Accounting-MemoryUsage AVP
Das Accounting-MemoryUsage AVP (AVP-Nummer 10006) ist vom Typ Unsigned32 und beinhaltet den verwendeten physikalischen Speicher in Kilobyte.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-NetworkUsage AVP
Das Accounting-NetworkUsage AVP (AVP-Nummer 10007) ist vom Typ Unsigned32 und beinhaltet die Anzahl gesendeter Daten über das Netzwerkinterface in Kilobyte.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-NodeCount AVP
Das Accounting-NodeCount AVP (AVP-Nummer 10008) ist vom Typ Unsigned32 und beinhaltet die Anzahl werwendeten Nodes.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-ProcessId AVP
Das Accounting-ProcessId AVP (AVP-Nummer 10009) ist vom Typ Unsigned32 und beinhaltet die Prozessnummer des Job.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-ProcessorCount AVP
Das Accounting-ProcessorCount AVP (AVP-Nummer 10010) ist vom Typ Unsigned32 und beinhaltet die Anzahl der verwendeten Prozessoren.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-QueueName AVP
Das Accounting-QueueName AVP (AVP-Nummer 10011) ist vom abgeleiteten Typ UTF8String und beinhaltet den Namen der Warteschlange von welcher der Job ausgeführt oder übermittelt wurde (die maximale Länge des Warteschlangennamens beträgt 20-ASCII-Zeichen).
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 28 Byte.

Accounting-ScratchUsage AVP
Das Accounting-ScratchUsage AVP (AVP-Nummer 10012) ist vom Typ Unsigned32 und beinhaltet den verwendeten Scratchspeicher in Kilobyte.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-ServiceLevelQuality AVP
Das Accounting-ServiceLevelQuality AVP (AVP-Nummer 10013) ist vom abgeleiteten Typ UTF8String und beinhaltet die QoS welche während des Ressourcenkonsums eingesetzt wurde (die maximale Länge der QoS-Bezeichnung beträgt 20-ASCII-Zeichen).
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 28 Byte.

Accounting-StartTime AVP
Das Accounting-StartTime AVP (AVP-Nummer 10014) ist vom abgeleiteten Typ Time und beinhaltet den Zeitpunkt des Startes einer Session.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-Status AVP
Das Accounting-Status AVP (AVP-Nummer 10015) ist vom Typ Unsigned32 und beinhaltet den Status nach Beendigung des Jobs:
1: aborted - Job wurde durch eine Regel oder Benutzer abgebrochen
2: completed - Job wurde erfolgreich beendet
3: failed - Job brach ab ohne externen Einfluss
4: held - Job wurde angehalten
5: queued - Job ist in der Warteschlange
6: started - Job wurde gestartet
7: suspended - Job wurde abgebrochen (Abbruchvorgang am laufen)
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-SubmitHost AVP
Das Accounting-SubmitHost AVP (AVP-Nummer 10016) ist vom abgeleiteten Typ UTF8String und beinhaltet den Systemhostname von welchem der Job übermittelt wurde (die maximale Länge des Systemhostnamens beträgt 20-ASCII-Zeichen).
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 28 Byte.

Accounting-SwapUsage AVP
Das Accounting-SwapUsage AVP (AVP-Nummer 10017) ist vom Typ Unsigned32 und beinhaltet den verwendeten Swapspeicher in Kilobyte.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-TempUsage AVP
Das Accounting-TempUsage AVP (AVP-Nummer 10018) ist vom Typ Unsigned32 und beinhaltet den verwendeten temporären Diskspeicher in Kilobyte.
Das Vendor-Bit darf nicht gesetzt sein. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Accounting-Application-Id AVP
Das Accounting-Application-Id AVP (AVP-Nummer 20000) ist vom Typ Unsigned32 und beinhaltet die Client-Applikations-Id. Die Applikations-Id dient dem Server als Information, welche Daten-AVPs von einem bestimmten Client gesendet werden können.
Das Vendor-Bit darf nicht gesetzt seind. Das Mandatory-Bit muss gesetzt sein und die Länge beträgt 12 Byte.

Die entsprechenden AVPs müssen dabei auch in der Datei dictionary.xml hinterlegt und definiert werden (vgl. Anhang C.3).

4.4 Diameter Kommandos

4.4.1 Übersicht

Im Diameter Basis-Protokoll [RFC3588] ist bereits ein Abrechnugs-Kommando (Nr. 271) vorgesehen. Dieses muss jedoch für eine spezifische Applikation angepasst und erweitert werden. Im Zuge des Designentscheides (siehe Kapitel 4.6) wurde entschieden, nicht für jede mögliche Mobile Grid Abrechnungsapplikation ein neues Accounting-Kommando und somit einen eigenes Acocunting-Modul zu definieren, sondern eine möglichst offene und erweiterbare Architektur zu wählen. Daher beinhaltet das erweiterte Abrechnungs-Kommando eine Vielzahl von optionalen AVPs, wodurch es allen möglichen Kombinationen von Abrechnungs-Parameter gerecht werden kann.

4.4.2 Accounting-Request (ACR)

Der Accounting-Request wurde insofern erweitert, dass sämtliche neu erstellten AVPs optional in einem ACR vorhanden sein können. Somit ist nicht für jedes Abrechnungs-Appliktion eine eigene Definition eines ACR notwendig.

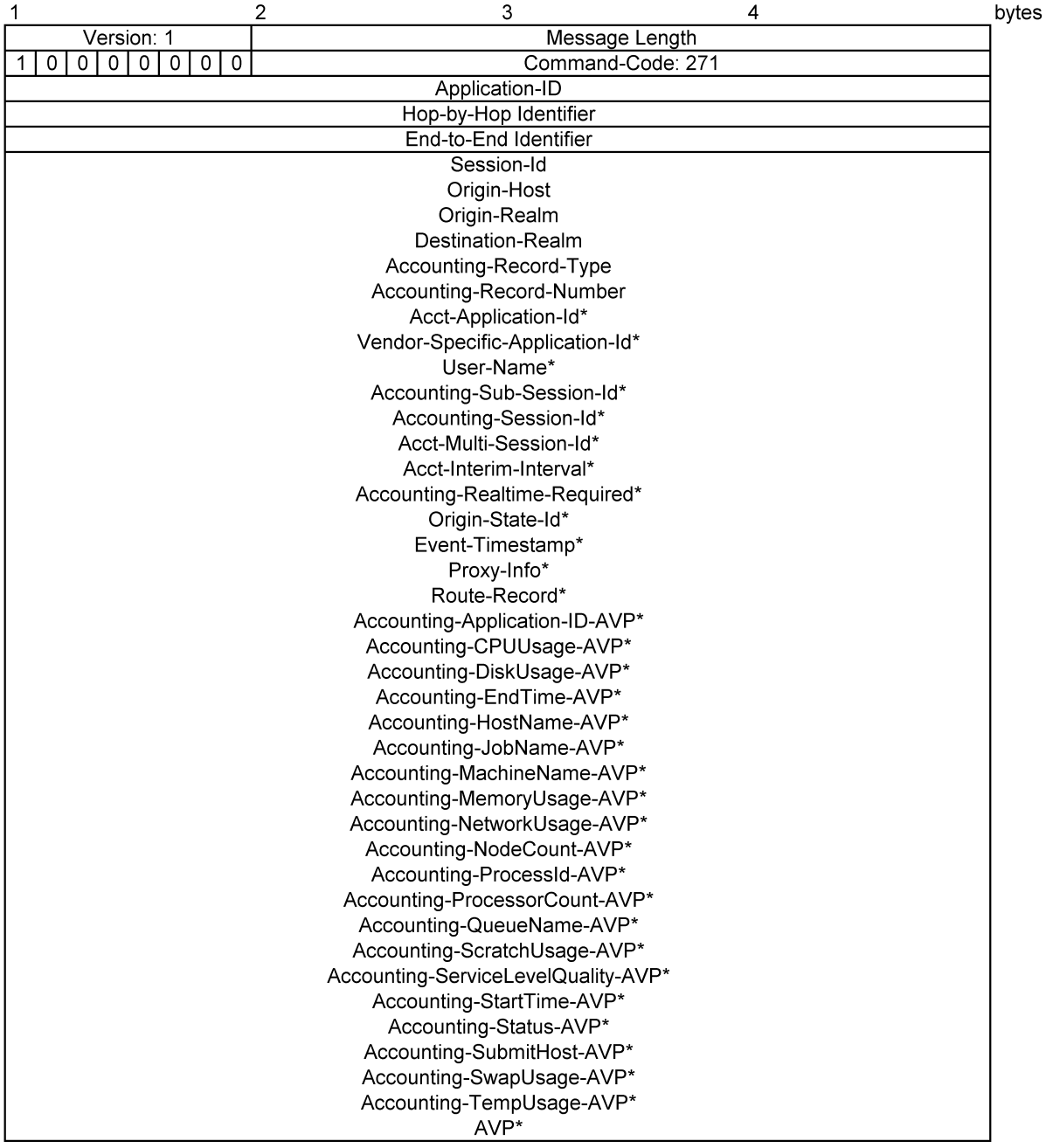

Abb. 11: Accounting-Request (ACR) des Abrechnungs-Moduls

4.4.3 Accounting-Answer (ACA)

Die Accounting-Answer wurde nur sehr geringfügig angepasst, da in der Antwort keine grundlegende Zusatzfunktionalität von Seiten Mobile Grids erforderlich ist.

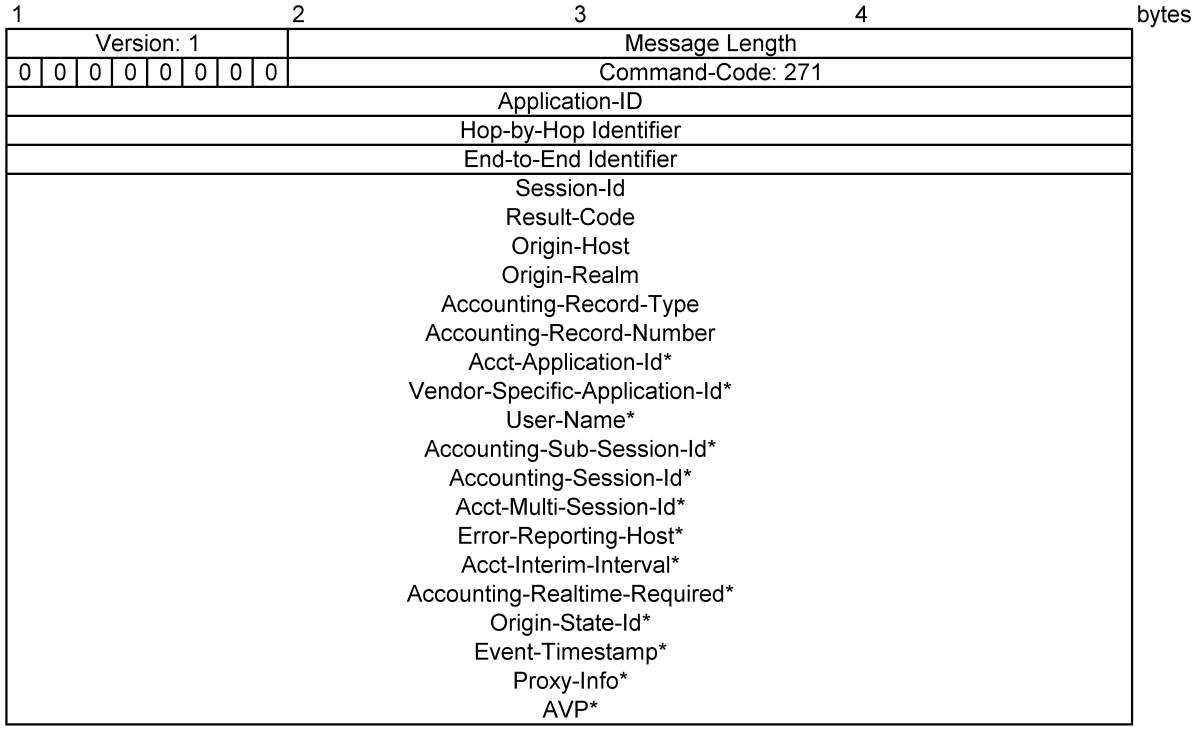

Abb. 12: Accounting-Answer (ACA) des Abrechnungs-Moduls

4.5 Softwarearchitektur

Die Architektur des vorliegenden Abrechnungs-Moduls ist zwangsläufig sehr stark an die Architektur des OpenDiameter-Frameworks angelegt. Es wurden daher bestehende Klassen und Methoden des Frameworks abgeleitet und überladen. Zudem wurden zusätzliche Klassen eingeführt um die gewünschte Funktionalität bereitstellen zu können.
Die Architektur sollte dabei so gestaltet sein, dass das Abrechnungsmodul sehr einfach in Applikationen integriert werden kann. Dies ist mit der vorliegenden Variante insofern möglich, dass eine externe Applikation lediglich ein Objekt vom Typ AccountingModule (Client- wie auch Serverseitig) erzeugen muss und anschliessend die zur verfügung gestellten Methoden als Schnittstellen verwenden kann. Die ganze Komplexität des OpenDiameter-Frameworks bleibt dabei für die externe Applikation transparent im Sinne der Informatik.
Die Schnittstellen sollten dabei so einfach wie möglich gehalten werden um Anfälligkeiten zu vermeiden.

4.5.1 AccountingServerModule

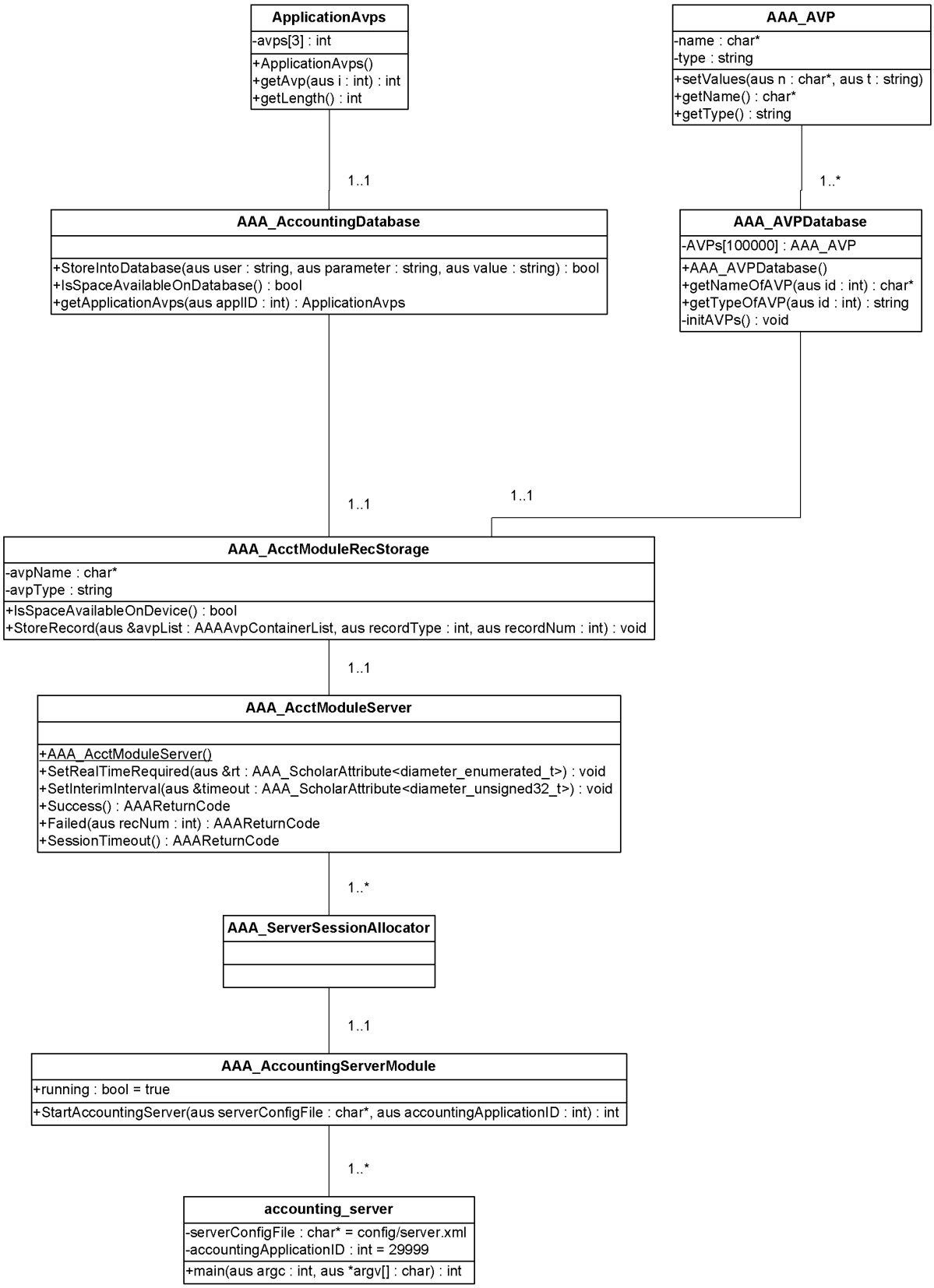

Abb. 13: AccountingServerModule-Architektur

Die oben dargestellte Server-Architektur besteht aus neun Klassen, welche nachfolgend kurz vorgestellt werden.

accounting_server
Diese Klasse zeigt auf, wie eine externe Applikation das Server-Modul einbinden müsste. Sie hat nebst Demonstrations- und Testzwecken keine weitere Funktion.

AAA_AccountingServerModule
Diese Klasse ist das Herzstück des Abrechnungsservers. Eine externe Applikation muss von dieser Klasse ein Objekt erzeugen um die Server-Abrechnungsfunktionalität einzubinden. Diese Klasse erstellt dann wiederum ein Objekt der Klasse AAA_ServerSessionAllocator.

AAA_ServerSessionAllocator
Die Allocator-Klasse ist dafür verantwortlich genügend Sessions für Clientverbindungen zur Verfügung zu stellen. Pro Clientsession erzeugt der Allocator somit ein Objekt der Klasse AAA_AcctModuleServer.

AAA_AcctModuleServer
Die AccountingModuleServer-Klasse verwaltet die gesamte Funktionalität des Transportes und überwacht den Erfolg bzw. Misserfolg einer Nachrichtenübermittlung. Bei einem Übertragungsfehler sendet diese Klasse dem Client eine Fehlermeldung und bei einer erfolgreichen Übermittlung wird eine positive Anwort ebenfalls von dieser Klasse versendet. Die Klasse erstellt zudem ein Objekt der Klasse AAA_AcctModuleRecStorage.

AAA_AcctModuleRecStorage
Diese Klasse ist verantwortlich für das Aufsplitten einer erfolgreich empfangenen Nachricht sowie der Speicherung dessen Inhaltes. Der Inhalt einer Nachricht wird anhand der in der Nachricht vorhandenen Informationen extrahiert und entsprechend an ein Objekt der Klasse AAA_AccountingDatabase weitergereicht. In jeder Nachrichtenübertragung wird durch den Client mitgeteilt, welche Clientapplikation die entsprechende Nachricht versendet hat. Anhand dessen kann die RecordStorage-Klasse eine Anfrage an das Datenbankobjekt stellen um zu erfahren, welche AVPs von der entsprechenden Clientapplikation zu erwarten sind. Anschliessend bereitet sich die Klasse auf die möglicherweise zu empfangenden AVPs (bzw. Informationen) vor, verarbeitet diese und speichert den Inhalt in die Datenbank.

AAA_AccountingDatabase
Die Database-Klasse ist als Schnittstelle zur Datenbank dafür verantwortlich, die empfangenen Informationen abzuspeichern. Danneben muss die Klasse auch anhand einer Clientapplikations-ID Auskunft darüber erteilen können, welche APVs (AVP-IDs) von der entsprechenden Clientapplikation zu erwarten sind. Im vorliegenden Prototyp wurde diese Funktionalität mit der zusätzlichen Klasse ApplicationAvps realisiert.

ApplicationAvps
Diese Klasse wurde nur für Testzwecke in diesem Prototyp implementiert und sollte später durch die korrekte Funktionalität (Datenbankschnittstelle) ersetzt werden.

AAA_AVPDatabase
Die AVPDatabase-Klasse speichert sämtliche Informationen über die AVPs. Anhand der AVP-ID kann von dieser Klasse die Bezeichnung sowie den Datentyp einer AVP angefordert werden. Diese Informationen sind für die Verarbeitung in der Klasse AAA_AcctModuleRecStorage unbedingt notwendig. Die Informationen werden dabei in Form eines Objektarrays von Objekten der Klasse AAA_AVP gehalten.

AAA_AVP
Objekte dieser Klasse representieren einzelne AVPs, welche die Informationsbasis der Klasse AAA_AVPDatabase bilden.

4.5.2 Einbindung des AccountingServerModule

Die Einbindung des AccountingServerModule ist für eine externe Applikation äusserst einfach. Die Applikation muss lediglich, analog zur Demonstrationsklasse accounting_server, ein Objekt vom Typ AAA_AccountingServerModule erzeugen und anschliessend die Methode StartAccountingServer mit den entsprechenden Parametern aufrufen. Die benötigten Parameter sind in diesem Falle der Pfad zur Konfigurationsdatei sowie die ID der Abrechnungsapplikation, welche in unserem Fall jeweils „29999“ ist (vgl. hierzu die dictionary.xml-Datei im Anhang).
Folgende Codezeilen zeigen die Einbindung des AccountingServerModule auf:

```
AAA_AccountingServerModule aaaAcctSrvModule;
aaaAcctSrvModule.StartAccountingServer("config/server.xml", 29999);
```

4.5.3 AccountingClientModule

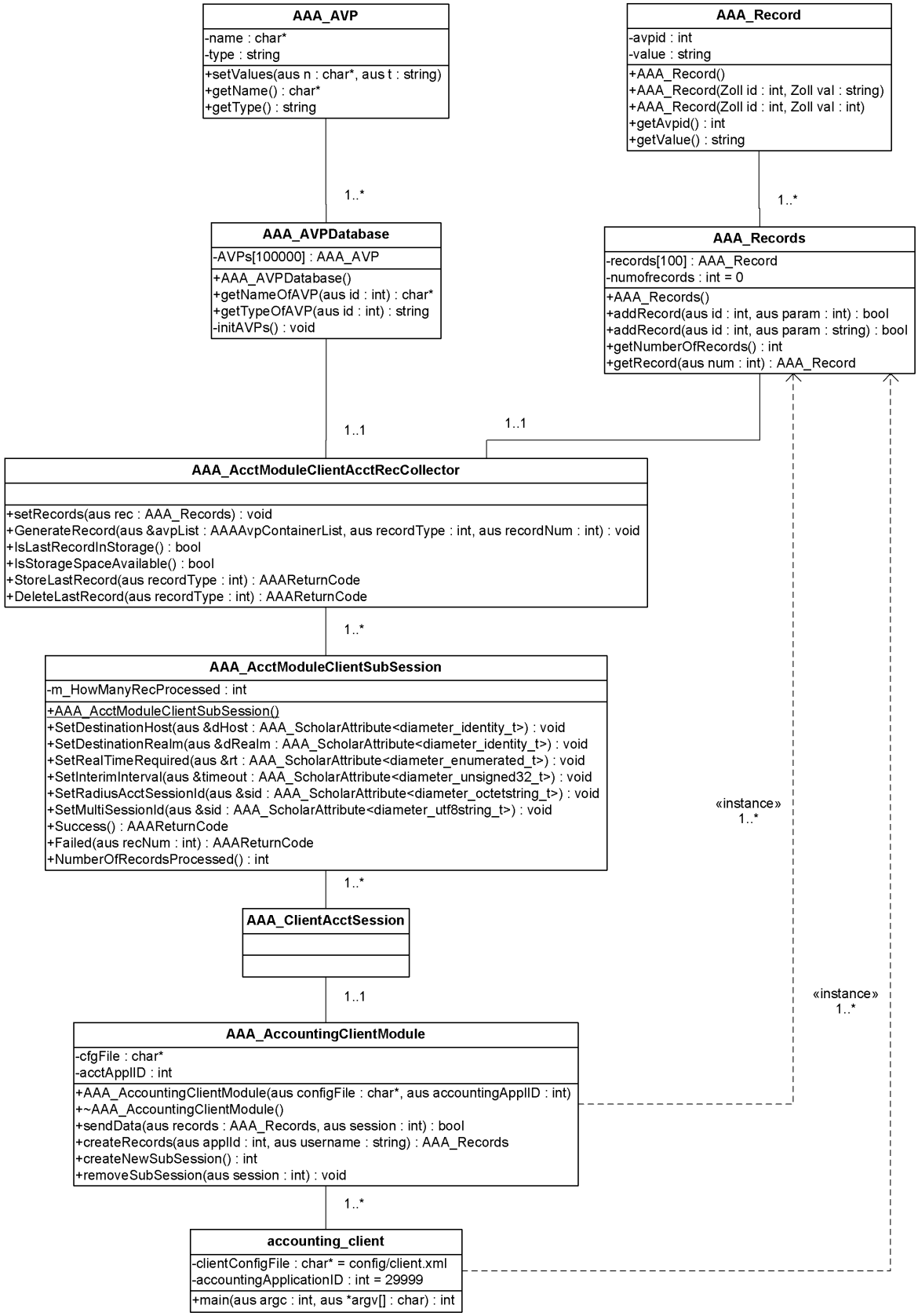

Abb. 14: AccountingClientModule-Architektur

Die oben dargestellte Client-Architektur besteht ebenso aus neun Klassen, welche nachfolgend kurz vorgestellt werden.

accounting_client
Diese Klasse zeigt auf, wie eine externe Applikation das Client-Modul einbinden müsste. Sie hat nebst Demonstrations- und Testzwecken keine weitere Funktion.

AAA_AccountingClientModule
Die ClientModule-Klasse ist das Herzstück des Abrechnungsclients. Eine externe Applikation muss von dieser Klasse ein Objekt erzeugen um die Client-Abrechnungsfunktionalität einzubinden. Diese Klasse erstellt dann wiederum ein Objekt der Klasse AAA_ClientAcctSession.

AAA_ClientAcctSession
Diese Klasse representiert die Muttersession aller Subsessions einer Clientapplikation. Basierend auf der Muttersession können dann wiederum Objekte (Subsessions) der Klasse AAA_AcctModuleClientSubSession erzeugt werden.

AAA_AcctModuleClientSubSession
Die SubSession-Klasse beinhaltet die eigentliche Funktionalität für das zuverlässige versenden von Diameter-Nachrichten. Dabei werden Rückmeldungen vom Server empfangen und je nach Status entsprechend reagiert. Die Klasse beinhaltet zudem eine gewisse Konfigurationsfunktionalität wobei beispielsweise der Zielserver fix einprogrammiert werden kann. Pro Clientsession sollte eine Subsession erzeugt werden. Die SubSession-Klasse erzeugt wiederum Objekte der Klasse AAA_AcctModuleClientAcctRecCollector.

AAA_AcctModuleClientAcctRecCollector
Die Collector-Klasse ist für das Aufbereiten der Informationen und das anschliessende Senden an den Server verantwortlich. Dabei werden auch Nachrichten, welche momentan gerade nicht an den Server versendet werden können, zwischengespeichert. Die Klasse wird dabei mit einem AAA_Records-Objekt bedient, welches sämtliche Informationen beinhaltet, welche an den Server in Form von AVPs verschickt werden sollen. Dabei werden aus den Informationen zuerst AVPs erzeugt. Hierfür werden wiederum Informationen wie Datentyp und Bezeichnung einer AVP aus der AAA_AVPDatabase abgefragt.

AAA_AVPDatabase
Die AVPDatabase-Klasse speichert sämtliche Informationen über die AVPs. Anhand der AVP-ID kann von dieser Klasse die Bezeichnung sowie der Datentyp einer AVP angefordert werden. Diese Informationen sind für die Verarbeitung in der Klasse AAA_AcctModuleClientAcctRecCollector unbedingt notwendig. Die Informationen werden dabei in Form eines Objektarrays von Objekten der Klasse AAA_AVP gehalten.

AAA_AVP
Objekte dieser Klasse representieren einzelne AVPs, welche die Informationsbasis der Klasse AAA_AVPDatabase bilden.

AAA_Records
Die Records-Klasse hält alle AAA_Record-Objekte in Form eines AAA_Record-Arrays welche vom Accountingclient an den Server verschicht werden sollen. Ein Objekt dieser Klasse wird ursprünglich von der Klasse AAA_AccountingClientModule erzeugt und anschliessend an den AccountingClient zur Informationseingabe weitergegeben. Sobald sämtliche Informationen im Objekt abgelegt sind, wird dieses wiederum über die Klasse

AAA_AccountingClientModule an den AAA_AcctModuleClientAcctRecCollector zum Versand weitergereicht.

AAA_Record
Objekte dieser Klasse representieren einzelne Informationseinträge, welche als eine gesamte Information von der Klasse AAA_Records gehalten werden.

4.5.4 Einbindung des AccountingClientModule

Die Einbindung des AccountingClientModule ist für eine externe Applikation wiederum relativ einfach. Die Applikation muss zu Beginn, analog zur Demonstrationsklasse accounting_client, ein Objekt vom Typ AAA_AccountingClientModule mit den entsprechenden Parameter erzeugen. Die benötigten Parameter sind in diesem Falle der Pfad zur Konfigurationsdatei sowie die ID der Abrechnungsapplikation, welche in unserem Fall jeweils „29999“ ist (vgl. hierzu die dictionary.xml-Datei im Anhang).
Für jeden User sollte anschliessend eine eigene SubSession erzeugt werden. Dabei ist zu beachten, dass jede Subsession ihre eigene SubSession-Id während der Dauer der Subsession behält.
Anschliessend muss ein AAA_Records-Objekt basierend auf der Rückgabe des AAA_AccountingClientModule erstellt werden. Als Parameter sind dabei die eigene Accountingclient-ID sowie der entsprechende Benutzername anzugeben. Die eigene Accountingclient-ID muss dabei zuvor auf dem Server registriert worden sein und bildet für den Server die Informationsbasis über die verwendeten AVPs.
Anschliessend können über die Methode addRecord Informationen abgelegt werden.
Sobald alle Informationen hinterlegt sind, kann das erzeugte records-Objekt mit Angabe der SubSession über das AAA_AccountingClientModule an den Server versendet werden.
Wird die SubSession anschliessend nicht mehr benötigt, sollte diese geschlossen werden.

Folgende Codezeilen zeigen die Einbindung des AccountingClientModule auf:

```
AAA_AccountingClientModule aaaAcctClientModule("config/client.xml", 29999);
int testsession = aaaAcctClientModule.createNewSubSession();
AAA_Records records = aaaAcctClientModule.createRecords(100, "abachmann@access.unizh.ch");
records.addRecord(10001, 123456);
records.addRecord(10003, "ARVO.IFI.UNIZH.CH");
records.addRecord(10008, 23);

if(aaaAcctClientModule.sendData(records, testsession)) {
   std::cout << "Data successfully sent" << std::endl;
}
else {
   std::cout << "A failure occured while etablishing connection / sending data..." << std::endl;
}
aaaAcctClientModule.removeSubSession(testsession);
```

4.6 Designentscheide

In der Wahl des Designs musste grosse Rücksicht auf das OpenDiameter-Framework genommen werden. Nichts deszu Trotz mussten einige ausschlaggebende Designentscheide getroffen werden.
Es wurde dabei besonders Wert auf möglichst generische Schnittstellen gelegt. Weiterhin wurde der Einfachheit spezielle Aufmerksamkeit geschenkt um die Komplexität des Systemes nicht unnötig zu erhöhen.
Ein wichtiger Designentscheid ist sicherlich, dass nur ein einziges Accounting-Kommando vorgesehen ist. Somit muss nicht für jede AVP-Konstellation ein eigenes bzw. neues Kommando implementiert werden. Das implementierte Kommanto behält für alle möglichen

Accountingclient-Varianten seine Gültigkeit indem alle neu definierten Abrechnungs-AVPs im vorliegenden Accounting-Kommando optional vorhanden sein können.
Dies machte es jedoch erforderlich, eine zusätzliche Accountingclient-ID einzuführen, welche jeder Client besitzen muss. Diese ID muss zusammen mit den AVPs, welche der Client versenden können möchte, in der Serverapplikation vorgängig registriert werden. Damit kann verhindert werden, dass der Server bei jeder empfangenen Nachricht nach allen möglichen (im Kommando als optional definierten) AVPs suchen muss. Dank der Zusatzinformation in Form der AccountingClient-ID, hat der Server die benötigte Infromation, sich auf die richtigen potentiellen AVPs vorzubereiten.
Zu Beginn wurde die Implementierung so vorgenommen, dass für jede zu übertragende Nachricht eine eigene SubSession (mit entsprechender eigener Subsession-Id) erzeugt und nach der Übertragung wieder beendet wurde. Später wurde jedoch entschieden, dass der Abrechnungsclient selber entscheiden sollte, was zu welcher Subsession gehört und somit dessen Verwaltung selber übernehmen sollte. So kann die SubSession beispielsweise während einer Kundensession aufrecht erhalten bleiben wobei die SubSessionId während der Dauer der Kundensession ebenfalls gleich bleibt. Hierfür wurden zwei zusätzliche Methoden implementiert, welche für die Erzeugung und Beendigung von SubSessions zuständig sind. Die Übertragungsprotokolle (siehe Anhang E.1 sowie E.2) zeigen dabei deutlich den Unterschied zwischen der früheren und der aktuellen Version der SubSession-Verwaltung auf.

4.7 Installation des Abrechnungs-Moduls

Die Installation bzw. die Kompilierung des Abrechnungsmoduls stellt sich dabei als ein Leichtes heraus. Hierfür sind lediglich die Quellen auf der beiliegenden CD-Rom (siehe Anhang) in das libdiameter-Verzeichnis zu kopieren. Einem allfälligen Überschreiben ist zuzustimmen. Anschliessend kann mittels „make“-Kommando eine Kompilierung der Quellen vorgenommen werden:

```
mount /cdrom
cp /cdrom/Abrechnungs-Modul/libdiameter /root/opendiameter/opendiameter-1.0.7-f/libdiameter
cd /root/opendiameter/opendiameter-1.0.7-f/libdiameter
./make
```

4.8 Offene Probleme

Im vorliegenden Prototyp sind noch einige Probleme vorhanden, welche bisher nicht gelöst werden konnten oder bewusst offen gelassen wurden:

- Die Schnittstelle zur Datenbank ist nur testhalber für die funktionalen Tests implementiert worden. Eine endgültige Programmierung dieser Schnittstellen müsste deshalb in einem zukünftigen Release noch vorgenommen werden.
- Der Klasse AAA_AVPDatabase sind zur Zeit alle benötigten Informationen über die AVPs im Quellcode fest einprogrammiert. Darauf sollte jedoch aus Gründen der Anpassbarkeit und Flexibilität verzichtet werden. Die entsprechend benötigten Daten sollten bei der Objekterstellung direkt aus der Konfigurationsdatei dictionary.xml gelesen werden. Ein entsprechende Schnittstelle müsste auch hier noch implementiert werden.
- Die Implementierung von Vendor-spezifischen AVPs war im Zuge dieses Prototyps leider nicht möglich. Bei weiteren Releases sollte unbedingt darauf geachtet werden, die im Rahmen dieser Semesterarbeit definierten neuen AVPs, als Vendor-spezifische AVPs zu definieren und zu verarbeiten.
- Wird das ClientModule beendet, erscheint jeweils zum Schluss ein SegmentationFault (bzw. Speicherzugriffsfehler). Dieser Fehler hat jedoch nicht den geringsten Einfluss auf die Funktionalität des Modules. Der Fehler konnte jedoch nicht eliminiert werden, da er

bereits in den Beispielsprogrammen des Frameworks vorhanden ist und die Ursache somit im Framework selber zu suchen ist.

5 Das Session Initiation Protocol (SIP) im Zusammenspiel mit AAA-Server

5.1 Ausgangslage

Das Signalisierungsprotokoll SIP setzt sich in diversen multimedialen Bereichen wie beispielsweise Voice-over-IP (VoIP) länger je mehr durch. Analog ist in vielen Einsatzbereichen vom SIP auch die Funktionalität von AAA-Servern erwünscht. RADIUS hat dabei den entscheidenden Nachteil, dass sinnvolle AAA-Funktionen in Verbindung mit SIP nur über den Umweg von herstellerspezifischen Erweiterungen realisiert werden können. Diese Erweiterungen haben jedoch den entscheidenden Nachteil, dass diese wiederum inkompatibel zueinander sind. Diameter ist daher auch im Einsatzbereich vom SIP eine denkbare Lösung zur Bewältigung der vorhandenen Probleme.
Martin Horner beschreibt in seiner Diplomarbeit [HO04] dabei ausführlich die Anforderungen an eine Diameter-SIP-Applikation basierend auf [SIP05].

5.2 Einführung in das Session Initiation Protocol (SIP)

Das Session Initiation Protocol ist in [RFC3261] spezifiziert und ist ein Signalisierungsprotokoll zum Erstellen, Verändern und Beenden von Multimedia-Dialogen zu einer Vielzahl von Endpunkten in paketorientierten Netzen. Es wird für verschiedene Multimedia-Anwendungen wie Beispielsweise Instant-Messaging, Internet-Telefonie oder Online-Spiele benutzt. SIP ist dabei zeichenbasiert, ähnelt im Aufbau sehr dem geläufigen Hypertext Transfer Protocol (HTTP) und ist einfach und erweiterbar, weshalb es sich mittlerweile als eines der geläufigsten Signalisierungsprotokolle etabliert hat.
Im TCP-IP-Modell ist SIP in der Anwendungsschicht angesiedelt und daher unabhängig vom verwendeten Transportprotokoll, wodurch es mit TCP, SCTP wie auch UDP zusammen benutzt werden kann.

5.3 Diameter-SIP-Applikation

5.3.1 Überblick

Die Diameter-SIP-Applikation soll die AAA-Funktionen im SIP-Umfeld definieren. Leider existiert für diese Applikation jedoch nur ein Draft [SIP05] welcher jedoch primär nur die Authentisierung und Autorisierung rudimentär beschreibt. Bezüglich der Abrechnungs-Erweiterung existierte früher einen Draft der IETF, welcher jedoch im Jahr 2003 abgelaufen ist und leider nicht mehr gepflegt wird. Bedingt durch diese recht spärliche Dokumentation sind die folgenden Unterkapitel als Entwurf zu sehen, wobei keine Gewährleistung gegeben werden kann, dass diese Applikation mit einer zukünftigen standardisierten Applikation kompatibel sein wird (vgl. hierzu [HO04]).

5.3.2 SIP-spezifische AVPs

Um die benötigten Elemente aus den SIP-Nachrichten sinnvoll in Diameter-Nachrichten transportieren zu können, sind einige neue AVPs notwendig welche wie folgt definiert werden:

SIP-AOR-AVP
Dieses AVP ist aus [SIP05] entnommen und ist vom Typ UTF8String. Es beinhaltet die Request-URI aus der SIP-Nachricht. Diese kann bei SIP-Anfragen aus der Start-Line und bei SIP-Antworten aus der From-Zeile entnommen werden.

SIP-Method-AVP
Dieses AVP ist aus [SIP05] entnommen und ist vom Typ UTF8String. Es beinhaltet die SIP-Methode der Nachricht, die diese Diameter-Nachricht ausgelöst hat. Es wird bei Anfragen aus der Start-Line und bei Antworten aus der CSeq Zeile gewonnen.

SIP-Server-URI-AVP
Dieses AVP ist aus [SIP05] entnommen und ist vom Typ UTF8String. Es beinhaltet eine SIP-URI, die einen SIP Server identifiziert.

SIP-Response-Code-AVP
Dieses AVP ist vom Typ Unsigned32, enthält den Response-Code von SIP-Antworten.

SIP-From-AVP
Dieses AVP ist vom Typ UTF8String und beinhaltet den Inhalt der From-Zeile einer SIP-Nachricht.

SIP-To-AVP
Dieses AVP ist vom Typ UTF8String und beinhaltet den Inhalt der To-Zeile einer SIP-Nachricht.

SIP-Remote-IP-Address-AVP
Dieses AVP ist vom Typ Unsigned32 und beinhaltet die IP-Adresse des Senders der SIP-Nachricht, die diese Diameter-Nachricht ausgelöst hat.

SIP-Remote-Port-AVP
Dieses AVP ist vom Typ Unsigned32 und beinhaltet den Port des Senders der SIP-Nachricht, die diese Diameter-Nachricht ausgelöst hat.

Für eine detaillierte Beschreibung der für SIP erforderlichen AVPs sei der interessierte Leser auf [HO04] verwiesen. Weitere SIP-spezifische AVPs sind zudem in [SIP05] definiert und beschrieben.

5.3.3 Autorisierung und Authentifizierung: MAR / MAA

Das Diameter-Basisprotokoll stellt keine speziellen Kommandos für Autorisierung und Authentifizierung zur Verfügung, sondern übergibt diese Aufgabe an die jeweilige spezifische Applikation. Der Grund hierfür liegt in den sehr verschiedenen Anforderungen, welche bei den einzelnen Applikationen bei der Authentifizierung und Autorisierung benötigt werden. In [SIP05] wird dafür der Multimedia-Authentication-Request (MAR) und die Multimedia-

Authentication-Response (MAA) als zusätzliches SIP-spezifisches Diameter-Kommando definiert.

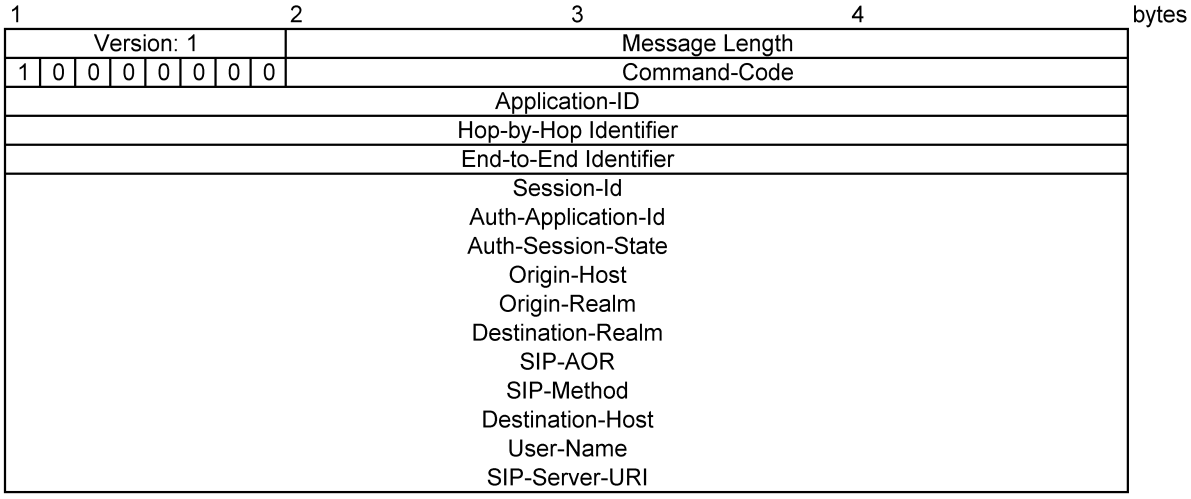

Abb. 15: Multimedia-Authentication-Request (MAR) der Diameter-SIP-Applikation

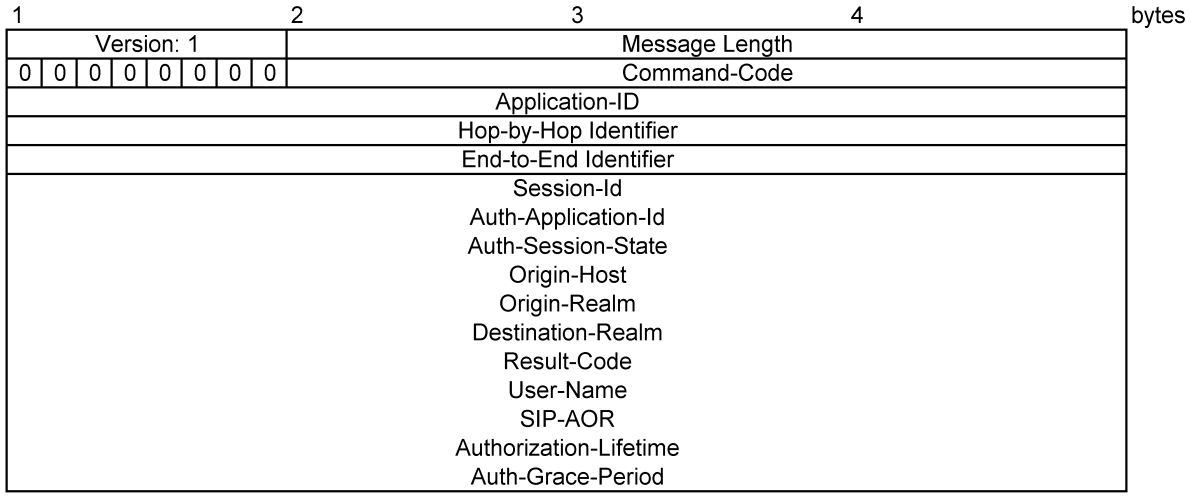

Abb. 16: Multimedia-Authentication-Answer (MAA) der Diameter-SIP-Applikation

In der MAR ist das User-Name-AVP und das SIP-Method-AVP massgeblich. Das User-Name-AVP beinhaltet die SIP-URI des Benutzers, welcher den MAR auslöst, wobei das SIP-Method-AVP die Methode der entsprechenden SIP-Nachricht enthält (vgl. hierzu [HO04]).

5.3.4 Abrechnung: ACR / ACA

Das Diameter-Basisprotokoll gibt für die Abrechnungsfunktionalität ein Abrechnungs-Kommando, in Form von Accounting-Request (ACR) und Accounting-Answer (ACA), vor. Dieses sollte jedoch von den jeweiligen Applikationen angepasst und erweitert werden. In [SIP05] sind jedoch keinerlei Erweiterungen für das Abrechnungs-Kommando vorgesehen, wodurch nur die Vorgaben des Diameter-Basisprotokolls ausschlaggebend sind. Da jedoch einige SIP-Daten (beispielsweise die SIP-Methode) bei der Abrechnung eine wichtige Rolle spielen, muss dafür das Abrechnungs-Kommando zwingend SIP-spezifisch erweitert werden. Daher ist eine eigene Erweiterung erforderlich, welche auf die SIP-spezifischen AVPs zurückgreift. Diese

AVPs waren ursprünglich durch [SIP05] für die Authentisierungs- und Autorisierungsnachrichten vorgesehen, können jedoch auch für Abrechnungszwecke verwendet werden.

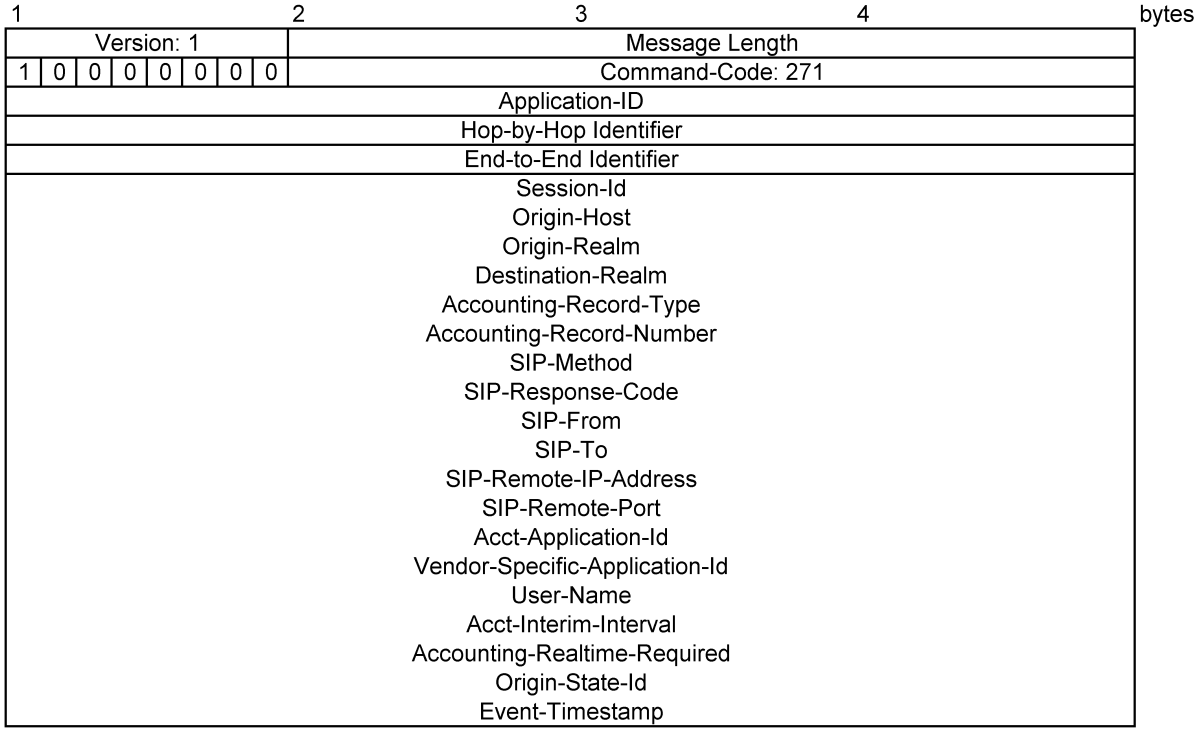

Abb. 17: Accounting-Request (ACR) der Diameter-SIP-Applikation

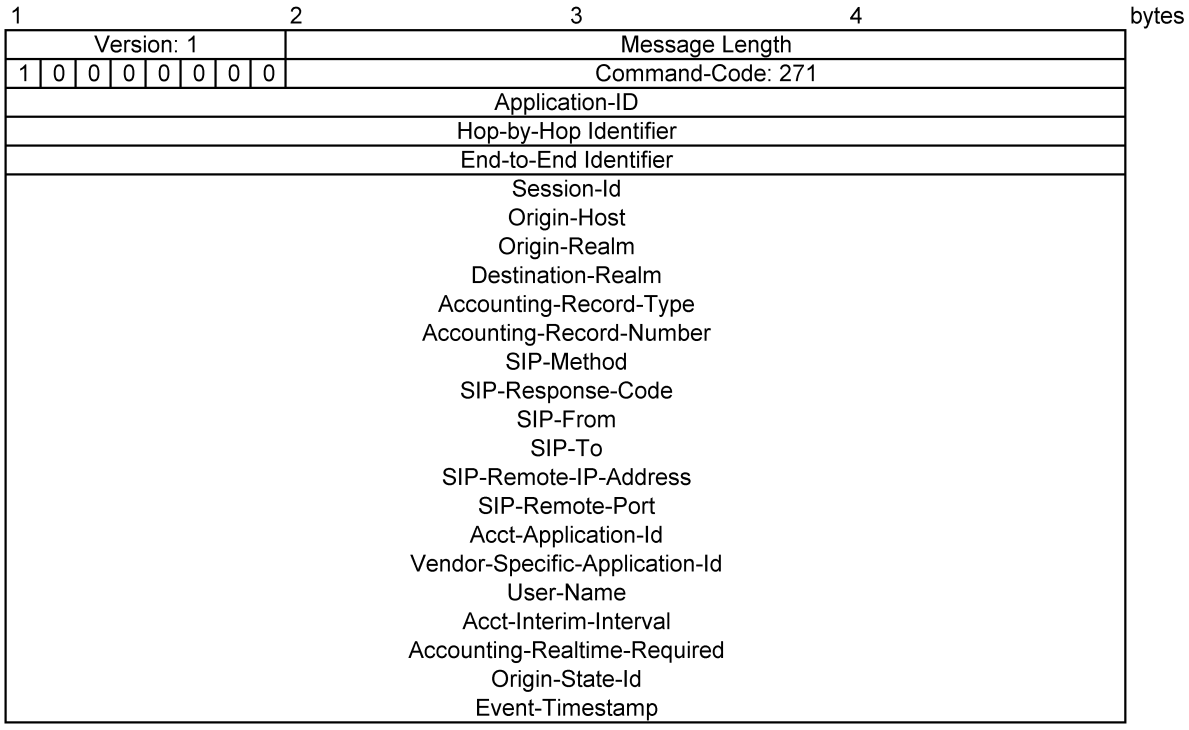

Abb. 18: Accounting-Answer (ACA) der Diameter-SIP-Applikation

Der ACR transportiert dabei alle abrechnungsrelevanten Daten zum Diameter-Server (vgl. hierzu [HO04]).

6 Besondere Herausforderungen

Die vorliegende Semesterarbeit stellte für mich als Autor eine grosse Herausforderung dar. Um dem Leser einige Referenzpunkte meiner persönlichen Herausforderungen aufzeigen zu können, ist dieses Kapitel in einge Unterkapitel logisch unterteilt.

6.1 Einführung in die Thematik

Als Wirtschafsinformatikstudent waren für mich die Bereiche MobileGrids sowie AAA-Architekturen (inkl. RADIUS und Diameter) komplettes Neuland. Es verlangte daher viel Aufwand für die Einarbeitung und insbesondere die Lektüre entsprechender Literatur. Dabei wurde viel der zur Verfügung stehenden Zeit beantsprucht, alleine um sich in die Thematik einzuarbeiten ohne dabei konkrete Resultate erzielt zu haben.

6.2 Installation von OpenDiameter

Das OpenDiameter-Framework steht der Internet-Community als OpenSource-Paket frei zur Verfügung. Leider tauchten bereits bei den ersten Installationsversuchen grössere Probleme auf. OpenDiameter stellte dabei grosse Anforderungen an die benötigten Zusatzbibliotheken, welche in einer exakt definierten Version vorliegen müssen. Auch war es selbst nach mehreren Versuchen, mit unterschiedlichsten Konfigurationen des Betriebssystemes, nicht möglich, OpenDiameter auf einer Debian-Distribution in einen lauffähigen Zustand zu versetzten. Erst ein Wechsel auf Suse Professional Linux brachte den gewünschten Erfolg.

6.3 OpenDiameter-Architektur, API und Dokumentation

Leider begrenzten sich die Probleme mit OpenDiameter nicht nur auf die schwierige Installation sondern schlugen sich auch in bei den ersten Testversuchen nieder. OpenDiemeter ist ein grosses und äusserst komplexes Framework, welches zudem relativ schlecht dokumentiert ist. Es liegt nur eine spärliche API vor, welche nur einige Beispiele nennt. Die einzige Zusatzhilfe liefern die 5 Codebeispiele welche im Paket mitgeliefert werden. Jedoch sind selbst diese schlecht dokumentiert. So gestaltete sich die Implementierung aufgrund des fehlenden Verständnisses, zu Beginn als ein regelrechter Blindflug.

6.4 Undokumentierte Funktionalität(en)

Wie bereits im vorhergenden Unterkapitel angesprochen, verursachte die fehlende bzw. schlechte Dokumentation des Frameworks einiges Kopfzerbrechen. So waren/sind auch einige benötigte Funktionen des Frameworks nirgends dokumentiert oder in einem Beispiel demonstriert. Es musste daher in gewissen Bereichen Reverse-Engineering betrieben werden, um an die benötigen Informationen zur Klassenhierarchie, Schnittstellen und vorhandener Methoden zu kommen und ein Verständins dafür aufbauen zu können.

6.5 Erarbeitung des Designs

Als letzte Herausforderung standen die Designentscheide auf dem Programm. In diversen Meetings wurde zusammen mit dem betreuenden Assistenten über die Designmöglichkeiten und dessen Vor- und Nachteile debatiert. Die Entscheidungsfindung stellte dabei eine weitere Herausforderung dar.

7 Zusammenfassung und Schlusswort

7.1 Praktischer Nutzen im Zusammenhang mit Mobile Grids

Mobile Grids bringen eine Vielzahl von neuen Anforderugen an AAA-Systeme welche diese Arbeit abzudecken versuchte. Dafür wurden weitere Abrechnugnsparameter definiert und in AVPs umgesetzt. Weiterhin wurde diese Arbeit auf Basis des Diameter-Protokolls erarbeitet, womit eine Vielzahl der zusätzlichen Anforderungen (u.a. durch die Peer-to-Peer-Architektur) abgedeckt werden konnten.
Ich bin davon überzeugt, dass auf Basis dieser Arbeit, dank den offenen, skalierbaren und erweiterbaren Schnittstellen, ein praktischer Nutzen für den Einsatz von Mobile Grids vorhanden ist. Selbstversändlich ist dies erst der Anfang auf einem langen Entwicklungsprozess bis hin zu einem marktreifen Produkt, jedoch wurde mit dieser Arbeit der erste Grundstein gelegt.

7.2 Schlusswort

AAA-Systeme gewinnen länger je mehr an Bedeutung. Dabei ist gerade im Umfeld der Authentifizierung und Autorisierung grosses Interesse durch die steigene Anzahl authentifizierungsrelevanter Dienste vorhanden. Zukünfigtige Verrechnungsmodelle möchten nicht mehr nur die klassischen Verrechnungsparameter berücksichtigen, sondern auch nach konsumiertem Inhalt oder anderen Parameter eine Abrechnung erstellen. Diameter eröffnet dabei neue Möglichkeiten in all diesen Bereichen, welche das alte Protokoll RADIUS nicht erfüllen konnte. Trotzdem fehlt es an marktreifen Produkten und es sind nur wenige, meist prototypische, Implementierungen vorhanden.
Die vorliegende Arbeit hat sich die neuen Möglichkeiten von Diameter zu Nutze gemacht und fördert eine grössere Nutzung dieses Protokolls in Zukunft. Eine Weiterführung dieses ersten Grundsteines wäre daher wünschenswert, um auch Impulse in anderen Bereichen für die Verwendung von Diameter zu setzen. Dabei ist der vorliegende Prototyp ganz klar als solcher zu betrachten und bedarf noch einiger Anpassungen sowie Weiterentwicklungsarbeiten um diesen Weg zu beschreiten.

7.3 Weiterführende Arbeiten

Im vorliegenden Prototyp wurden die Schnittstellen zu einer Datenbank nicht implementiert, da dies Gegenstand einer anderen Arbeit ist. In diesem Bereich muss daher sicherlich nochmals überprüft werden, welche Daten sinnvollerweise in der Datenbank und in welcher Form abgelegt werden sollten.
Die in jeder Nachricht an den Server übermittelte Accountingclient-ID bildet die Informationsbasis für die potentiell in der Nachricht vorhandenen AVPs. Der Server kann diese Informationen von der Datenbankschnittstelle aufgrund der empfangenen Accountingclient-ID anfordern. Damit sich jedoch auch neue Abrechnungsclients, mit ihren entsprechenden AVPs, Nachrichten an einen Server senden können, ist eine Registrierungsmöglichkeit für neue Applikationen von Nöten. Ein entsprechendes Interface müsste hierfür zusätzlich implementiert werden, beispielsweise mit Hilfe eines neuen Diameter-Kommandos.
Die Klasse AAA_AVPDatabase unterhält zur Zeit eine eigene Datenbank der definierten AVPs mit der entsprechenden ID, Name sowie Datentyp. Die Klasse sollte daher insofern erweitert werden, dass die benötigten Daten über die AVPs nicht zusätzlich hartkodiert im Sourcecode vorliegen müssen, sondern direkt aus der Datei dictionary.xml bei der Objekterzeugung gelesen werden. Damit wird der Flexibilität und Anpassbarkeit des Produktes Rechnung getragen.

Abschliessend sollten noch diverse, momentan nur oberflächlich implementeierte, Methoden des OpenDiameter-Frameworks tiefergreifend implementiert werden. Als Beispiels sei hier die Methode IsLastRecordInStorage() der Klasse AAA_AcctModuleClientAcctRecCollector genannt.

7.4 Verwandte Literatur

Zur ganzen AAA-Thematik existiert eine grosse Zahl an Literatur. An dieser Stelle sei deshalb nur auf weitere bzw. verwandte Literatur verwiesen, welche im direkten Zusammenhang mit dieser Arbeit steht. Sämtlich hier genannte Literatur befindet sich auch auf der CD-Rom (siehe Anhang).

- [RFC2903] bietet sicherlich einen sehr guten Einstieg in die gesamte Thematik in dem die generische AAA-Architektur spezifiziert wird.
- [RFC3588] bildet die Basis der vorliegenden Arbeit, indem das Diameter-Basisprotokoll inklusive den Basis-AVPs sowie Basis-Befehlen definiert wird.
- [CA05] bildet die API von Diameter und ist daher sicherlich auch empfehlenswert.
- [AR99] beschäftigt sich mit der Abrechnungserweriterung für Diameter. Durch den Fokus auf die Abrechnung in dieser Arbeit bildet das inzwischen leider etwas in die Jahre gekommene Dokument sicherlich auch eine gute Basis.
- [RFC2924] definiert die Abrechungsattribute sowie die Datenformate für die Dateneinträge.
- [WU04] ist eine Diplomarbeit über das Design und die Implementierung von WIRE Diameter. Durch den unmittelbaren Bezug zur vorliegenden Arbeit bildet auch diese Arbeit eine interessante Basis.
- [RFC3702] beschreibt die Anforderungen an AAA-Dienste im Zusammenhang mit dem Session Initiation Protocol (SIP).
- [SIP05] geht konkret auf die Implementierung einer Diameter SIP-Applikation ein. Leider vernachlässigt dieser Entwurf etwas die Abrechnungsfunktionalität und spezifiziert primär nur die Funktionen Autorisierung und Authentifizierung.
- [HO04] ist eine Diplomarbeit zum Thema Diameter-Accounting beim Session Initiation Protocol (SIP). Diese Arbeit geht konkret auf [SIP05] ein und zeigt Schwachstellen und deren Lösung auf.

Literaturverzeichnis

[AAAP05] IANA, Webseite, AAA Parameters: http://www.iana.org/assignments/aaa-parameters, August 2005.

[ACE05] Adaptive Communication Environment (ACE), Webseite: http://www.cs.wustl.edu/~schmidt/ACE.html, August 2005.

[AR99] J. Arkko et. al.: DIAMETER Accounting Extension (Internet Draft). The Internet Society, Dezember 1999.

[CA02] P. R. Calhoun et. al.: AAA Problem Statements (Internet Draft). The Interent Society, Januar 2002.

[CA05] P. R. Calhoun et. al.: The Diameter API (Internet Draft). The Internet Society, März 2005.

[EK00] R. Ekstein et. al.: AAA Protocols: Coparison between RADIUS, DIAMETER and COPS (Internet Draft). The Interent Society, April 2000.

[HA02] Jonathan Hassell: RADIUS - Securing Public Access to Private Resources. O'REILLY, 1st Edition, Oktober 2002.

[HO04] M. Horner: Diameter-Accounting beim Session Initiation Protocol (SIP). Fachhochschule Worms, Fachbereich Informatik / TK, 2004.

[HP02] Hewlett-Packard: HP-UX AAA Server A.05.01, Administrator's Guide. Hewlett-Packard Company, Cupertino, 2002. Abrufbar unter http://docs.hp.com/en/T1428-90044/T1428-90044.pdf

[IANA05] IANA, Webseite: http://www.iana.org, Juli 2005.

[MA04] R. Mach et. al.: Accounting Interchange Natural Language Description (Requirements). Global Grid Forum, Februar 2004.

[OD05] OpenDiameter Webseite, Documentation Page: http://diameter.sourceforge.net, Mai 2005.

[RFC2486] B. Aboba, M. Beadles: The Network Access Identifier (RFC 2486). The Interent Society, Januar 1999.

[RFC2865] C. Rigney et. al.: Remote Authentication Dial In User Service (RADIUS) (RFC 2865). The Interent Society, Juni 2000.

[RFC2903] C. de Laat et. al.: Generic AAA Architecture (RFC 2903). The Internet Society, August 2000.

[RFC2924] N. Brownlee et. al.: Accounting Attributes and Record Formats (RFC 2924). The Internet Society, September 2000.

[RFC2960] R. Stewart et. al.: Stream Control Transmission Protocol (RFC 2960). The Internet Society, Oktober 2000.

[RFC3261] J. Rosenberg et. al.: SIP: Session Initiation Protocol (RFC 3261). The Internet Society, Juni 2002.

[RFC3588] P. Calhoun et. al.: Diameter Base Protocol (RFC 3588). The Internet Society, September 2003.

[RFC3702] J. Loughney et. al.: Authentication, Authorization and Accounting Requirements for the Session Initiation Protocol (SIP) (RFC 3702). The Internet Society, Februar 2004.

[SIP05] M. Garcia-Martin et. al.: Diameter Session Initiation Protocol (SIP) Application (Internet Draft). The Internet Society, März 2005.

[SUSE05] SUSE LINUX, Webseite: http://www.suse.de, Mai 2005.

[WIKI05] Wikipedia, Webseite, Triple-A-System: http://de.wikipedia.org/wiki/Authentication%2C_Authorization%2C_Accounting, Juni 2005

[WU04] Wen-Ting Wu: Design and Implementation of WIRE Diameter, Seite 22. National Tsing Hua University, Juli 2004.

[XERCES05] Apache XML Project, Xerces C++ Parser: http://xml.apache.org/xerces-c/, Mai 2005.

Anhang

Anhangverzeichnis

Anhang A: Inhalt der CD-Rom

Verzeichnis/Datei(en)	Beschreibung
/Bericht/FM	Die vorliegende Semesterarbeit im FrameMaker-Format (Book)
/Bericht/PDF	Die vorliegende Semesterarbeit im PDF-Format
/Bericht/PS	Die vorliegende Semesterarbeit im Postscript-Format
/Grafiken	Sämtliche in der Semesterarbeit verwendeten Grafiken und Abbildungen im TIFF-Format
/Grafikquellen	Microsoft Visio- und Excel-Dateien welche zur Erzeugung gewisser Grafiken verwendet wurden
/Quellen	Sämtliche für die Programmierung verwendeten Quellen im tar.gz-Format
/Abrechnungs-Modul	Quellen des Prototyps
/Konfigurationsdateien	Sämtliche Konfigurationsdateien in der endgültigen Version
/Praesentation	PowerPoint-Slides der Abschlusspräsentation als PPT- und PDF-Dateien
/Literatur	Verwandte sowie weiterführende Literatur im PDF-Format

Anhang B: Verzeichnisstruktur Abrechnungs-Modul

Das im Rahmen dieser Semeserarbeit geschriebene Abrechnungs-Modul befindet sich auf der CD-Rom im Verzeichnis “Abrechnungs-Modul”. Die Verzeichnisstruktur ist dabei wie folgt ausgelegt:

libdiameter	Beinhaltet alle benötigten Quellen für das Abrechungs-Modul, die kompilierten Binaries sowie die Makefiles
libdiameter/.libs	Beinhaltet die libdiameter-Bibliotheken
libdiameter/accounting-module	Beinhaltet den Quellcode des Abrechnungs-Moduls
libdiameter/config	Beinhaltet sämtliche für das Abrechnungs-Modul benötigten Konfigurationsdateien
libdiameter/config/old	Beinhaltet sämtliche Konfigurationsdateien in der ursprünglichen Version
libdiameter/include	Beinhaltet sämtliche Header-Dateien des OpenDiameter-Frameworks
libdiameter/src	Beinhaltet den Quellcode des OpenDiameter-Frameworks
libdiameter/test	Beinhaltet den Quellcode der OpenDiameter-Testprogramme

Anhang C: Konfigurationsdateien

C.1 client.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation='configuration.xsd'>
  <general>
      <product>Open Diameter</product>
      <version>1</version>
      <vendor_id>0</vendor_id>
      <supported_vendor_id>0</supported_vendor_id>
      <supported_vendor_id>1</supported_vendor_id>
      <supported_vendor_id>9999</supported_vendor_id>
      <auth_application_id>1</auth_application_id>
      <auth_application_id>2</auth_application_id>
      <auth_application_id>19999</auth_application_id>
```

```
      <acct_application_id>3</acct_application_id>
      <acct_application_id>4</acct_application_id>
      <acct_application_id>29999</acct_application_id>
      <vendor_specific_application_id>
          <vendor_id>9999</vendor_id>
          <auth_application_id>19999</auth_application_id>
          <acct_application_id>29999</acct_application_id>
      </vendor_specific_application_id>
   </general>
   <parser>
      <dictionary>config/dictionary.xml</dictionary>
   </parser>
   <transport_mngt>
      <identity>client.aaaclient.net</identity>
      <realm>aaaclient.net</realm>
      <tcp_port>1811</tcp_port>
      <tls_port>0</tls_port>
      <watchdog_timeout>3</watchdog_timeout>
      <peer_table>
          <expiration_time>1</expiration_time>
          <peer>
              <hostname>server.aaaserver.net</hostname>
              <port>1812</port>
              <tls_enabled>0</tls_enabled>
          </peer>
      </peer_table>
      <route_table>
          <expire_time>0</expire_time>
          <route>
             <realm>aaaserver.net</realm>
             <role>1</role>
             <application>
                <application_id>1</application_id>
                <vendor_id>0</vendor_id>
                <peer_entry>
                    <server>server.aaaserver.net</server>
                    <metric>2</metric>
                </peer_entry>
             </application>
             <application>
                <application_id>0</application_id>
                <vendor_id>0</vendor_id>
                <peer_entry>
                    <server>server.aaaserver.net</server>
                    <metric>2</metric>
                </peer_entry>
             </application>
          </route>
          <default_route>
             <realm>aaaserver.net</realm>
             <role>0</role>
             <application>
                <application_id>1</application_id>
                <vendor_id>0</vendor_id>
                <peer_entry>
                    <server>server.aaaserver.net</server>
                    <metric>4</metric>
                </peer_entry>
                <peer_entry>
                    <server>server.aaaserver.net</server>
                    <metric>5</metric>
                </peer_entry>
             </application>
          </default_route>
      </route_table>
   </transport_mngt>
   <session_mngt>
      <max_sessions>10000</max_sessions>
      <auth_sessions>
         <stateful>1</stateful>
```

```
         <session_timeout>30</session_timeout>
         <lifetime_timeout>360</lifetime_timeout>
         <grace_period_timeout>30</grace_period_timeout>
         <abort_retry_timeout>20</abort_retry_timeout>
      </auth_sessions>
      <acct_sessions>
         <session_timeout>30</session_timeout>
         <interim_interval>5</interim_interval>
         <realtime>1</realtime>
      </acct_sessions>
   </session_mngt>
    <log>
      <flags>
         <debug>enabled</debug>
         <trace>enabled</trace>
         <info>enabled</info>
      </flags>
      <target>
         <console>enabled</console>
         <syslog>disabled</syslog>
      </target>
   </log>
<!--   <log>
      <flags>
         <debug>disabled</debug>
         <trace>disabled</trace>
         <info>disabled</info>
      </flags>
      <target>
         <console>disabled</console>
         <syslog>disabled</syslog>
      </target>
   </log>-->
</configuration>
```

C.2 server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation='configuration.xsd'>
   <general>
      <product>Open Diameter</product>
      <version>1</version>
      <vendor_id>0</vendor_id>
      <supported_vendor_id>0</supported_vendor_id>
      <supported_vendor_id>1</supported_vendor_id>
      <supported_vendor_id>9999</supported_vendor_id>
      <auth_application_id>1</auth_application_id>
      <auth_application_id>2</auth_application_id>
      <auth_application_id>19999</auth_application_id>
      <acct_application_id>3</acct_application_id>
      <acct_application_id>4</acct_application_id>
      <acct_application_id>29999</acct_application_id>
      <vendor_specific_application_id>
          <vendor_id>9999</vendor_id>
          <auth_application_id>19999</auth_application_id>
          <acct_application_id>29999</acct_application_id>
      </vendor_specific_application_id>
   </general>
   <parser>
      <dictionary>config/dictionary.xml</dictionary>
   </parser>
   <transport_mngt>
      <identity>server.aaaserver.net</identity>
      <realm>aaaserver.net</realm>
      <tcp_port>1812</tcp_port>
      <tls_port>0</tls_port>
      <watchdog_timeout>4</watchdog_timeout>
      <peer_table>
```

```
          <expiration_time>1</expiration_time>
          <peer>
              <hostname>client.aaaclient.net</hostname>
              <port>1811</port>
              <tls_enabled>0</tls_enabled>
          </peer>
      </peer_table>
      <route_table>
          <expire_time>0</expire_time>
          <route>
             <realm>aaaclient.net</realm>
             <role>0</role>
             <application>
                <application_id>0</application_id>
                <vendor_id>0</vendor_id>
                <peer_entry>
                    <server>client.aaaclient.net</server>
                    <metric>2</metric>
                </peer_entry>
             </application>
             <application>
                <application_id>1</application_id>
                <vendor_id>0</vendor_id>
                <peer_entry>
                    <server>client.aaaclient.net</server>
                    <metric>2</metric>
                </peer_entry>
             </application>
          </route>
          <default_route>
             <realm>aaaclient.net</realm>
             <role>0</role>
             <application>
                <application_id>0</application_id>
                <vendor_id>0</vendor_id>
                <peer_entry>
                    <server>client.aaaclient.net</server>
                    <metric>4</metric>
                </peer_entry>
             </application>
          </default_route>
      </route_table>
   </transport_mngt>
   <session_mngt>
      <max_sessions>10000</max_sessions>
      <auth_sessions>
         <stateful>1</stateful>
         <session_timeout>30</session_timeout>
         <lifetime_timeout>360</lifetime_timeout>
         <grace_period_timeout>30</grace_period_timeout>
         <abort_retry_timeout>20</abort_retry_timeout>
      </auth_sessions>
      <acct_sessions>
         <session_timeout>30</session_timeout>
         <interim_interval>5</interim_interval>
         <realtime>1</realtime>
      </acct_sessions>
   </session_mngt>
<!--<log>
      <flags>
         <debug>enabled</debug>
         <trace>enabled</trace>
         <info>enabled</info>
      </flags>
      <target>
         <console>enabled</console>
         <syslog>enabled</syslog>
      </target>
   </log>-->
<log>
```

```
      <flags>
         <debug>disabled</debug>
         <trace>disabled</trace>
         <info>disabled</info>
      </flags>
      <target>
         <console>disabled</console>
         <syslog>disabled</syslog>
      </target>
   </log>
</configuration>
```

C.3 dictionary.xml

Anmerkung: Die Datei dictionary.xml ist mehrere Seiten lang. Es wird daher darauf verzichtet, die komplette Datei hier abzudrucken. Der folgende Quellcode beinhaltet nur die zusätzlichen Informationen welche der Datei gegenüber dem Originalzustand hinzugefügt wurden.

```
  <!-- ************************************************************* -->
  <!-- ***************** AUTHORIZATION-APPLICATION ******************** -->
  <!-- ************************************************************* -->
    <application id="19999" name="Authorization" uri="http://www.ifi.unizh.ch/csg">
    <command name="Authorization" code="300">
      <requestrules>
        <fixed>
          <avprule name="Session-Id" maximum="1" minimum="1"/>
        </fixed>
        <required>
          <avprule name="Origin-Host" maximum="1" minimum="1"/>
          <avprule name="Origin-Realm" maximum="1" minimum="1"/>
          <avprule name="Destination-Realm" maximum="1" minimum="1"/>
          <avprule name="Auth-Application-Id" maximum="1" minimum="1"/>
          <avprule name="Re-Auth-Request-Type" maximum="1" minimum="1"/>
        </required>
        <optional>
          <avprule name="Auth-Session-State" maximum="1"/>
          <avprule name="Authorization-Lifetime" maximum="1"/>
          <avprule name="Auth-Grace-Period" maximum="1"/>
          <avprule name="Destination-Host" maximum="1"/>
          <avprule name="User-Name" maximum="1"/>
          <avprule name="Origin-State-Id" maximum="1"/>
          <avprule name="AVP"/>
          <avprule name="Proxy-Info"/>
          <avprule name="Route-Record"/>
        </optional>
      </requestrules>
      <answerrules>
        <fixed>
          <avprule name="Session-Id" maximum="1" minimum="1"/>
        </fixed>
        <required>
          <avprule name="Result-Code" maximum="1" minimum="1"/>
          <avprule name="Origin-Host" maximum="1" minimum="1"/>
          <avprule name="Origin-Realm" maximum="1" minimum="1"/>
        </required>
        <optional>
          <avprule name="Auth-Session-State" maximum="1"/>
          <avprule name="Authorization-Lifetime" maximum="1"/>
          <avprule name="Auth-Grace-Period" maximum="1"/>
          <avprule name="Session-Timeout" maximum="1"/>
          <avprule name="Error-Message" maximum="1"/>
          <avprule name="Error-Reporting-Host" maximum="1"/>
          <avprule name="Failed-AVP"/>
          <avprule name="Redirect-Host"/>
          <avprule name="Redirect-Host-Usage" maximum="1"/>
          <avprule name="Redirect-Max-Cache-Time" maximum="1"/>
          <avprule name="Proxy-Info"/>
          <avprule name="AVP"/>
        </optional>
```

```
    </answerrules>
  </command>
</application>

<!-- ************************************************************ -->
<!-- ***************** ACCOUNTING-APPLICATION ******************** -->
<!-- ************************************************************ -->
<application id="29999" name="Accounting" uri="http://www.ifi.unizh.ch/csg">
  <command name="Accounting" code="271">
    <requestrules>
      <fixed>
        <avprule name="Session-Id" maximum="1" minimum="1"/>
      </fixed>
      <required>
        <avprule name="Origin-Host" maximum="1" minimum="1"/>
        <avprule name="Origin-Realm" maximum="1" minimum="1"/>
        <avprule name="Destination-Realm" maximum="1" minimum="1"/>
        <avprule name="Accounting-Record-Type" maximum="1" minimum="1"/>
        <avprule name="Accounting-Record-Number" maximum="1" minimum="1"/>
      </required>
      <optional>
        <avprule name="Acct-Application-Id" maximum="1"/>
        <avprule name="Vendor-Specific-Application-Id" maximum="1"/>
        <avprule name="User-Name" maximum="1"/>
        <avprule name="Accounting-Sub-Session-Id" maximum="1"/>
        <avprule name="Accounting-Session-Id" maximum="1"/>
        <avprule name="Acct-Multi-Session-Id" maximum="1"/>
        <avprule name="Acct-Interim-Interval" maximum="1"/>
        <avprule name="Accounting-Realtime-Required" maximum="1"/>
        <avprule name="Origin-State-Id" maximum="1"/>
        <avprule name="Event-Timestamp" maximum="1"/>
        <avprule name="Proxy-Info"/>
        <avprule name="Route-Record"/>
        <avprule name="Test-AVP"/>
        <!-- ALL OUR NEW DEFINED AVPs ARE ALSO OPTIONAL POSSBILE -->
        <avprule name="Accounting-Application-Id-AVP" maximum="1"/>
        <avprule name="Accounting-CPUUsage-AVP"/>
        <avprule name="Accounting-DiskUsage-AVP"/>
        <avprule name="Accounting-EndTime-AVP"/>
        <avprule name="Accounting-HostName-AVP"/>
        <avprule name="Accounting-JobName-AVP"/>
        <avprule name="Accounting-MachineName-AVP"/>
        <avprule name="Accounting-MemoryUsage-AVP"/>
        <avprule name="Accounting-NetworkUsage-AVP"/>
        <avprule name="Accounting-NodeCount-AVP"/>
        <avprule name="Accounting-ProcessId-AVP"/>
        <avprule name="Accounting-ProcessorCount-AVP"/>
        <avprule name="Accounting-QueueName-AVP"/>
        <avprule name="Accounting-ScratchUsage-AVP"/>
        <avprule name="Accounting-ServiceLevelQuality-AVP"/>
        <avprule name="Accounting-StartTime-AVP"/>
        <avprule name="Accounting-Status-AVP"/>
        <avprule name="Accounting-SubmitHost-AVP"/>
        <avprule name="Accounting-SwapUsage-AVP"/>
        <avprule name="Accounting-TempUsage-AVP"/>
        <!-- END OF ADDITIONAL OPTIONAL AVPs -->
        <avprule name="AVP"/> <!-- THIS AVP MUST BE THE LAST DEFINED AVP! OTHERWISE THE AVPs
                                  BELOW "AVP" ARE UNKNOWN..... -->
      </optional>
    </requestrules>
    <answerrules>
      <fixed>
        <avprule name="Session-Id" maximum="1" minimum="1"/>
      </fixed>
      <required>
        <avprule name="Result-Code" maximum="1" minimum="1"/>
        <avprule name="Origin-Host" maximum="1" minimum="1"/>
        <avprule name="Origin-Realm" maximum="1" minimum="1"/>
        <avprule name="Accounting-Record-Type" maximum="1" minimum="1"/>
        <avprule name="Accounting-Record-Number" maximum="1" minimum="1"/>
```

```
        </required>
        <optional>
          <avprule name="Acct-Application-Id" maximum="1"/>
          <avprule name="Vendor-Specific-Application-Id" maximum="1"/>
          <avprule name="User-Name" maximum="1"/>
          <avprule name="Accounting-Sub-Session-Id" maximum="1"/>
          <avprule name="Accounting-Session-Id" maximum="1"/>
          <avprule name="Acct-Multi-Session-Id" maximum="1"/>
          <avprule name="Error-Reporting-Host" maximum="1"/>
          <avprule name="Acct-Interim-Interval" maximum="1"/>
          <avprule name="Accounting-Realtime-Required" maximum="1"/>
          <avprule name="Origin-State-Id" maximum="1"/>
          <avprule name="Event-Timestamp" maximum="1"/>
          <avprule name="Proxy-Info"/>
          <avprule name="AVP"/>
        </optional>
      </answerrules>
    </command>
    <!-- ***************** TEST APPLICATION AVP ************** -->
    <avp name="Test-AVP" code="99999" mandatory="mustnot" may-encrypt="no" protected="mustnot">
      <type type-name="UTF8String"/>
    </avp>

    <!-- ***************** NEW DEFINED ACCOUNTING AVPs ****************** -->
    <!-- ***************** UNIZH APPLICATION SPECIFIC AVPS ****************** -->
    <!-- ****** THE USAGE OF THE VENDOR-ID-BIT ISN'T SUPPORTED IN THIS VERSION !!!! ******** -->
    <avp name="Accounting-CPUUsage-AVP" code="10000" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <avp name="Accounting-DiskUsage-AVP" code="10001" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned64"/>
    </avp>

    <avp name="Accounting-EndTime-AVP" code="10002" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <!--<type type-name="Time"/> // UNSUPPORTED IN THIS VERSION-->
        <type type-name="UTF8String"/>
    </avp>

    <avp name="Accounting-HostName-AVP" code="10003" mandatory="mustnot" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="UTF8String"/>
    </avp>

    <avp name="Accounting-JobName-AVP" code="10004" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="UTF8String"/>
    </avp>

    <avp name="Accounting-MachineName-AVP" code="10005" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="UTF8String"/>
    </avp>

    <avp name="Accounting-MemoryUsage-AVP" code="10006" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <avp name="Accounting-NetworkUsage-AVP" code="10007" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <avp name="Accounting-NodeCount-AVP" code="10008" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
```

```
    </avp>

    <avp name="Accounting-ServiceLevelQuality-AVP" code="10013" mandatory="must" may-
encrypt="no" protected="mustnot">
        <type type-name="UTF8String"/>
    </avp>

    <avp name="Accounting-StartTime-AVP" code="10014" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <!--<type type-name="Time"/> // UNSUPPORTED IN THIS VERSION-->
        <type type-name="UTF8String"/>
    </avp>

    <avp name="Accounting-Status-AVP" code="10015" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <avp name="Accounting-SubmitHost-AVP" code="10016" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="UTF8String"/>
    </avp>

    <avp name="Accounting-SwapUsage-AVP" code="10017" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <avp name="Accounting-TempUsage-AVP" code="10018" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <avp name="Accounting-ProcessId-AVP" code="10009" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <avp name="Accounting-ProcessorCount-AVP" code="10010" mandatory="must" may-encrypt="no"
protected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <avp name="Accounting-QueueName-AVP" code="10011" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

   <avp name="Accounting-ScratchUsage-AVP" code="10012" mandatory="must" may-encrypt="no" pro-
tected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <avp name="Accounting-Application-Id-AVP" code="20000" mandatory="must" may-encrypt="no"
protected="mustnot">
        <type type-name="Unsigned32"/>
    </avp>

    <!-- ************* END OF ADDITIONALY DEFINED AVPs *************** -->
  </application>
```

C.4 dictionary.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
     <!--
        $Log: dictionary.dtd,v $
        Revision 1.2  2004/04/08 14:28:23  vfajardo
        Added changes for protocol id in dictionary
```

```
Revision 1.1  2004/02/19 16:47:43  vfajardo
migrated from top level directory

Revision 1.3  2003/09/30 23:39:05  yohba
Added Mobile IPv4 Application.

Revision 1.2  2003/09/30 23:37:42  yohba
Added Mobile IPv4 Application dictionary.

Revision 1.1.1.1  2002/11/06 21:19:49  frascone
Imported sources

Revision 1.1.1.1  2002/09/19 00:57:42  ohba
no message

Revision 1.1.1.1  2002/09/18 13:51:53  ohba
no message

Revision 1.4  2002/09/17 14:03:13  ohba
no message

Revision 1.3  2002/05/24 15:25:50  ohba
no message

Revision 1.2  2002/05/03 14:45:49  ohba
no message

Revision 1.1  2002/05/02 16:57:18  ohba
*** empty log message ***

Revision 1.2  2002/05/01 20:24:20  ohba
ebit and pbit added

Revision 1.1  2002/05/01 19:21:31  ohba
*** empty log message ***

Revision 1.1  2002/05/01 19:14:31  ohba
no message

Revision 1.7  2002/02/06 15:39:22  dave
Vendor name changed from #IMPLIED to #REQUIRED

Revision 1.6  2001/12/13 23:07:26  dave
Updated DTD and dictionary files with new changes.  Please review
and send me e-mail with any comments.

Revision 1.5  2001/09/27 16:50:44  dave
Testing CVS to mailing list.  Nothing changed in file.

Revision 1.4  2001/09/27 16:44:19  dave
Test for cvs

Revision 1.3  2001/09/19 21:38:57  mjones
Removed #PCDATA from command element.

Revision 1.2  2001/09/19 19:46:38  mjones
Moved the vendor element to be the same level as base and application.
Modified vendor-id to be SMI Private Enterprise Code instead of a label.
Removed vendor-id="None" since vendor-id was IMPLIED.
Added type attribute to command (request or answer).
Removed duplicate AVPs from nasreq.xml (Acct-Session-Id, Acct-Multi-Session-Id)
Corrected typos in enum codes for Auth-Session-State and Disconnect-Cause.

Revision 1.1  2001/08/24 18:04:44  chaos
Added per Mark's request

Revision 1.3  2001/07/31 17:43:36  chaos
Oops, forgot to turn on validity checking.
Fixed some errors found with validity checking turned on
```

```
   Revision 1.2  2001/07/31 16:56:15  chaos
   Lots of changes to support flags like in the draft, and to support commands

-->
<!ELEMENT dictionary (vendor*, base, application*)>
<!ATTLIST dictionary
     protocol CDATA #REQUIRED
>

<!ELEMENT vendor EMPTY>
<!ATTLIST vendor
     id CDATA #REQUIRED
     name CDATA #REQUIRED
>

<!ELEMENT base (command*, typedefn+, avp+)>
<!ATTLIST base
     uri CDATA #IMPLIED
>

<!--<!ELEMENT application EMPTY>
<!ATTLIST application
     name CDATA #REQUIRED
     filename CDATA #REQUIRED
>-->

<!ELEMENT application (command*, typedefn*, avp*)>
<!ATTLIST application
     id CDATA #REQUIRED
     name CDATA #IMPLIED
     uri CDATA #IMPLIED
>
<!-- Modified by Ohba
<!ELEMENT command (requestrules*, answerrules*)>
<!ATTLIST command
     name CDATA #REQUIRED
     code CDATA #REQUIRED
     vendor-id CDATA #IMPLIED
>
-->
<!ELEMENT command (requestrules*, answerrules*)>
<!ATTLIST command
     name CDATA #REQUIRED
     code CDATA #REQUIRED
     application-id CDATA #IMPLIED
     pbit CDATA #IMPLIED
>

<!ELEMENT typedefn EMPTY>
<!ATTLIST typedefn
     type-name ID #REQUIRED
     type-parent IDREF #IMPLIED
     description CDATA #IMPLIED
>
<!ELEMENT avp ((type | grouped), (enum*))>
<!ATTLIST avp
     name ID #REQUIRED
     description CDATA #IMPLIED
     code CDATA #REQUIRED
     may-encrypt (yes | no) "yes"
     mandatory (must | may | mustnot | shouldnot) "may"
     protected (must | may | mustnot | shouldnot) "may"
     vendor-id CDATA #IMPLIED
>
<!ELEMENT type EMPTY>
<!ATTLIST type
     type-name IDREF #REQUIRED
>
<!-- Modified by Ohba
<!ELEMENT grouped (gavp+)>
```

```
     -->
     <!ELEMENT grouped (fixed*, required*, optional*, fixed*)>
     <!-- Deleted by Ohba
     <!ELEMENT gavp EMPTY>
     <!ATTLIST gavp
          name IDREF #REQUIRED
          vendor-id CDATA #IMPLIED
     >
     -->
     <!ELEMENT enum EMPTY>
     <!ATTLIST enum
          name CDATA #REQUIRED
          code CDATA #REQUIRED
     >

     <!-- <!ELEMENT requestrules (avprule+)> Modified by Ohba -->
     <!ELEMENT requestrules (fixed*, required*, optional*, fixed*)>

     <!-- <!ELEMENT answerrules (avprule+)> Modified by Ohba -->
     <!ELEMENT answerrules (fixed*, required*, optional*, fixed*)>

     <!-- Next 3 lines were added by ohba -->
     <!ELEMENT fixed (avprule+)>
     <!ELEMENT required (avprule+)>
     <!ELEMENT optional (avprule+)>

     <!-- Modified by Ohba
     <!ELEMENT avprule EMPTY>
     <!ATTLIST avprule
          name IDREF #REQUIRED
             position (first | last | unspecified)  "unspecified"
             maximum CDATA "none"
             minimum CDATA "0"
     >
     -->
     <!ELEMENT avprule EMPTY>
     <!ATTLIST avprule
          name IDREF #REQUIRED
             maximum CDATA "none"
             minimum CDATA "0"
     >
```

C.5 configuration.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>

<xs:simpleType name="roleType">
   <xs:restriction base="xs:integer">
      <xs:enumeration value="0"/>
      <xs:enumeration value="1"/>
      <xs:enumeration value="2"/>
      <xs:enumeration value="3"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="enableType">
   <xs:restriction base="xs:string">
      <xs:pattern value="enabled|disabled"/>
   </xs:restriction>
</xs:simpleType>

<xs:complexType name="vendorApplicationIdType">
   <xs:sequence>
      <xs:element name="vendor_id" type="xs:integer" minOccurs="1" maxOccurs="100"/>
      <xs:element name="auth_application_id" type="xs:integer" minOccurs="0" maxOccurs="1"/>
      <xs:element name="acct_application_id" type="xs:integer" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="generalType">
   <xs:sequence>
      <xs:element name="product" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="version" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="vendor_id" type="xs:integer" minOccurs="1" maxOccurs="1"/>
      <xs:element name="supported_vendor_id" type="xs:integer" minOccurs="1" maxOccurs="100"/>
      <xs:element name="auth_application_id" type="xs:integer" minOccurs="1" maxOccurs="100"/>
      <xs:element name="acct_application_id" type="xs:integer" minOccurs="1" maxOccurs="100"/>
      <xs:element name="vendor_specific_application_id" type="vendorApplicationIdType" minOc-
curs="1" maxOccurs="100"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="parserType">
   <xs:all>
      <xs:element name="dictionary" type="xs:string"/>
   </xs:all>
</xs:complexType>

<xs:complexType name="peerType">
   <xs:all>
      <xs:element name="hostname" type="xs:string" maxOccurs="1" minOccurs="1"/>
      <xs:element name="port" type="xs:integer" default="1812" maxOccurs="1" minOccurs="1"/>
      <xs:element name="tls_enabled" type="xs:integer" default="0" maxOccurs="1" minOccurs="1"/
>
   </xs:all>
</xs:complexType>

<xs:complexType name="peerTableType">
   <xs:sequence>
      <xs:element name="expiration_time" type="xs:integer" minOccurs="1" maxOccurs="1"/>
      <xs:element name="peer" type="peerType" maxOccurs="100" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="routeApplicationPeerEntryType">
   <xs:all>
      <xs:element name="server" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="metric" type="xs:integer" minOccurs="1" maxOccurs="1"/>
   </xs:all>
</xs:complexType>

<xs:complexType name="routeApplicationType">
   <xs:sequence>
      <xs:element name="application_id" type="xs:integer" minOccurs="1" maxOccurs="1"/>
      <xs:element name="vendor_id" type="xs:integer" minOccurs="1" maxOccurs="1"/>
      <xs:element name="peer_entry" type="routeApplicationPeerEntryType" minOccurs="1" maxOc-
curs="100"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="routeType">
   <xs:sequence>
      <xs:element name="realm" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="role" type="roleType" minOccurs="1" maxOccurs="1"/>
      <xs:element name="application" type="routeApplicationType" minOccurs="1" maxOccurs="100"/
>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="routeTableType">
   <xs:sequence>
      <xs:element name="expire_time" type="xs:integer" minOccurs="1" maxOccurs="1"/>
      <xs:element name="route" type="routeType" maxOccurs="1000" minOccurs="0"/>
      <xs:element name="default_route" type="routeType" maxOccurs="1" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="transportType">
```

```
   <xs:all>
      <xs:element name="identity" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="realm" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="tcp_port" type="xs:integer" default="1812" maxOccurs="1" minOccurs="1"/
>
      <xs:element name="tls_port" type="xs:integer" default="1812" maxOccurs="1" minOccurs="1"/
>
      <xs:element name="watchdog_timeout" type="xs:integer" default="5" maxOccurs="1" minOc-
curs="1"/>
      <xs:element name="peer_table" type="peerTableType" minOccurs="1" maxOccurs="1"/>
      <xs:element name="route_table" type="routeTableType" minOccurs="1" maxOccurs="1"/>
   </xs:all>
</xs:complexType>

<xs:complexType name="authSessionType">
   <xs:all>
      <xs:element name="stateful" type="xs:integer" default="1" maxOccurs="1" minOccurs="0"/>
      <xs:element name="session_timeout" type="xs:integer" default="360"/>
      <xs:element name="lifetime_timeout" type="xs:integer" default="480"/>
      <xs:element name="grace_period_timeout" type="xs:integer" default="30"/>
      <xs:element name="abort_retry_timeout" type="xs:integer" default="30"/>
   </xs:all>
</xs:complexType>

<xs:complexType name="acctSessionType">
   <xs:all>
      <xs:element name="session_timeout" type="xs:integer" default="360"/>
      <xs:element name="interim_interval" type="xs:integer" default="5"/>
      <xs:element name="realtime" type="xs:integer" default="1"/>
   </xs:all>
</xs:complexType>

<xs:complexType name="sessionType">
   <xs:sequence>
      <xs:element name="max_sessions" type="xs:integer" default="65000"/>
      <xs:element name="auth_sessions" type="authSessionType"/>
      <xs:element name="acct_sessions" type="acctSessionType"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="logFlagsType">
   <xs:all>
      <xs:element name="debug" type="enableType" default="disabled"/>
      <xs:element name="trace" type="enableType" default="disabled"/>
      <xs:element name="info" type="enableType" default="disabled"/>
   </xs:all>
</xs:complexType>

<xs:complexType name="logTargetType">
   <xs:all>
      <xs:element name="console" type="enableType" default="enabled"/>
      <xs:element name="syslog" type="enableType" default="disabled"/>
   </xs:all>
</xs:complexType>

<xs:complexType name="logType">
   <xs:all>
      <xs:element name="flags" type="logFlagsType"/>
      <xs:element name="target" type="logTargetType"/>
   </xs:all>
</xs:complexType>

<xs:element name="configuration">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="general" type="generalType" minOccurs="1" maxOccurs="1"/>
      <xs:element name="parser" type="parserType" minOccurs="1" maxOccurs="1"/>
      <xs:element name="transport_mngt" type="transportType" minOccurs="1" maxOccurs="1"/>
      <xs:element name="session_mngt" type="sessionType" minOccurs="1" maxOccurs="1"/>
      <xs:element name="log" type="logType" minOccurs="1" maxOccurs="1"/>
```

```
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Anhang D: Source-Code

D.1 accounting-server.cxx

```
/* BEGIN_COPYRIGHT Open Diameter                                          */
/*                                                                        */
/* Open Diameter: Open-source software for the Diameter and               */
/*                Diameter related protocols                              */
/*                                                                        */
/* Copyright (C) 2002-2004 Open Diameter Project                          */
/*                                                                        */
/* This library is free software; you can redistribute it and/or modify   */
/* it under the terms of the GNU Lesser General Public License as         */
/* published by the Free Software Foundation; either version 2.1 of the   */
/* License, or (at your option) any later version.                        */
/*                                                                        */
/* This library is distributed in the hope that it will be useful,        */
/* but WITHOUT ANY WARRANTY; without even the implied warranty of         */
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU      */
/* Lesser General Public License for more details.                        */
/*                                                                        */
/* You should have received a copy of the GNU Lesser General Public       */
/* License along with this library; if not, write to the Free Software    */
/* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307    */
/* USA.                                                                   */
/*                                                                        */
/* In addition, when you copy and redistribute some or the entire part of */
/* the source code of this software with or without modification, you     */
/* MUST include this copyright notice in each copy.                       */
/*                                                                        */
/* If you make any changes that are appeared to be useful, please send    */
/* sources that include the changed part to                               */
/* diameter-developers@lists.sourceforge.net so that we can reflect your  */
/* changes to one unified version of this software.                       */
/*                                                                        */
/* END_COPYRIGHT Open Diameter                                            */
/*                                                                        */
/* BEGIN_COPYRIGHT Accounting Module                                      */
/*                                                                        */
/* Copyright (c) 2005 by Adrian Bachmann, adrian@bachmann.lu              */
/* This Accounting Module was programmed as a semester thesis             */
/* DESIGN AND PROTOTYPICAL IMPLEMENTATION OF AN ACCOUNTING SYSTEM FOR     */
/* AN AAA SERVER                                                          */
/* at the Institute of Informatics, University of Zurich                  */
/*                                                                        */
/* Author: Adrian Bachmann, adrian@bachmann.lu                            */
/* Supervisor: Cristian Morariu, morariu@ifi.unizh.ch                     */
/* Director: Prof. Dr. Burkhard Stiller, stiller@ifi.unizh.ch             */
/*                                                                        */
/* END_COPYRIGHT Accounting Module                                        */

// ACCOUNTING SERVER

// INCLUDE THE DIAMETER-API-HEADER-FILE
#include "diameter_api.h"
// INCLUDE THE STRING LIBRARY
#include <string>
// INCLUDE THE STRINGSTREAM LIBRARY FOR CASTING THE DATA TO STRING
#include <sstream>

// initialization of strings
using namespace std;
```

```
// This Class is only used for test purposes fpr simulating the database-interface
class ApplicationAvps {
 private:
   // Array with all the avps of an given Application-ID in it
   int avps[3];

 public:
   // Constructor which initializes the avps-array
   ApplicationAvps() {
     // set value #1
     avps[0] = 10003;
     // set value #2
     avps[1] = 10001;
     // set value #3
     avps[2] = 10008;
   }

   // this function returns the value for a given index
   int getAvp(int i) {
     // return array-value with the given index
     return avps[i];
   }

   // this function returns the lenght of the array... the function is used
   // for getting the number of parameters for a given Application-ID
   int getLength() {
     // Return a hard-coded value "3" which corresponds to the lenght of the array
     return 3;
   }
};

// The class AAA_AccountingDatabase is responsible for storing all
// the accounting data (received from the client through AVPs) into the
// database
class AAA_AccountingDatabase {
 public:
   // This function is used for storing the parameter and the corresponding
   // value received through an AVP into the AccountingDatabase
   // As Result-Code an unis-style code will sent back
   int StoreIntoDatabase(string user, string parameter, string value) {
     // send some informations to console
     std::cout << "Values stored in database:" << std::endl;
     // send some informations to console
     std::cout << "User: " << user << " Parameter: " << parameter << " Value: " << value <<
std::endl;
     /*
     THIS INTERFACE HAS TO BE IMPLEMENTED FOR STORING DATA INTO DATABASE
     */
     // send back the result code (unix-style)
     return (0);
   }

   // This function checks if there is enough space available on the
   // database server (or database itself)
   bool IsSpaceAvailableOnDatabase() {
     // THIS FUNCTION HAS TO BE IMPLEMENTED TO CHECK IF THERE IS ENOUGH
     // SPACE AVAILABLE ON DATABASE
     return (true);
   }

   // This function returns all the AVPs as an ApplicationAvps-object
   // which are metered by an AccountingApplication
   ApplicationAvps getApplicationAvps(int applID) {
     // Create ApplicationAvps object which contains some testdata
     ApplicationAvps a;
     // return the created object
     return a;
   }
};
```

```
// This class represents AVPs as objects
class AAA_AVP {
 public:
   // This function sets the values of an AVP
   void setValues(char* n, string t) {
     // Set the Name of AVP
     name = n;
     // Set the Type of AVP
     type = t;
   }

   // This function returns the name of the object (AVP)
   char* getName() {
     return name;
   }

   // This function returns the type of the object (AVP)
   string getType() {
     return type;
   }
 private:
   // Name of AVP e.g. "Destination-Host"
   char* name;
   // Type of AVP e.g. "DiameterIdentity"
   string type;
};

// The class AAA_AVPDatabase is used to store all the AVP-Names and AVP-datatypes
// which corresponds to an AVP-ID
class AAA_AVPDatabase{
 public:
   // Constructor which initializes automaticaly all the AVPs through the
   // funcion initAVPs()
   AAA_AVPDatabase() {
     // Initialize all the AVPs (manually)...
     initAVPs();
   }
 private:
   // Lets create an array with all the mapped AVPs (as objects) in it
   // The index corresponds to the AVP-ID
   AAA_AVP AVPs[100000];

   // Fills all the AVPs into the AVPs-Array
   // In this prototype we fill them all manually... later that should
   // be done based on the dictionary.xml-file automaticaly and directly
   void initAVPs() {
     /*
     AVPs[0].setValues("AVP","Any"); // DIAMETER BASE PROTOCOL
     AVPs[482].setValues("Acct-Interim-Interval","Unsigned32"); // DIAMETER BASE PROTOCOL
     AVPs[483].setValues("Accounting-Realtime-Required","Unsigned32"); // DIAMETER BASE PROTO-
COL
     AVPs[50].setValues("Acct-Multi-Session-Id","UTF8String"); // DIAMETER BASE PROTOCOL
     AVPs[485].setValues("Accounting-Record-Number","Unsigned32"); // DIAMETER BASE PROTOCOL
     AVPs[480].setValues("Accounting-Record-Type","Unsigned32"); // DIAMETER BASE PROTOCOL
     AVPs[44].setValues("Accounting-Session-Id","OctetString"); // DIAMETER BASE PROTOCOL
     AVPs[287].setValues("Accounting-Sub-Session-Id","Unsigned64"); // DIAMETER BASE PROTOCOL
     AVPs[259].setValues("Acct-Application-Id","Unsigned32"); // DIAMETER BASE PROTOCOL
     AVPs[275].setValues("Alternate-Peer","DiameterIdentity"); // DIAMETER BASE PROTOCOL
     AVPs[258].setValues("Auth-Application-Id","Unsigned32"); // DIAMETER BASE PROTOCOL
     AVPs[274].setValues("Auth-Type","Unsigned32"); // DIAMETER BASE PROTOCOL
     AVPs[291].setValues("Authorization-Lifetime","Unsigned32"); // DIAMETER BASE PROTOCOL
     AVPs[276].setValues("Auth-Grace-Period","Unsigned32"); // DIAMETER BASE PROTOCOL
     AVPs[277].setValues("Auth-Session-State","Enumerated"); // DIAMETER BASE PROTOCOL
     AVPs[25].setValues("Class","OctetString"); // DIAMETER BASE PROTOCOL
     AVPs[293].setValues("Destination-Host","DiameterIdentity"); // DIAMETER BASE PROTOCOL
     AVPs[283].setValues("Destination-Realm","DiameterIdentity"); // DIAMETER BASE PROTOCOL
     AVPs[273].setValues("Disconnect-Cause","Unsigned32"); // DIAMETER BASE PROTOCOL
     AVPs[281].setValues("Error-Message","UTF8String"); // DIAMETER BASE PROTOCOL
     AVPs[294].setValues("Error-Reporting-Host","DiameterIdentity"); // DIAMETER BASE PROTOCOL
```

```
    AVPs[55].setValues("Event-Timestamp","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[279].setValues("Failed-AVP","OctetString"); // DIAMETER BASE PROTOCOL
    AVPs[267].setValues("Firmware-Revision","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[257].setValues("Host-IP-Address","Address"); // DIAMETER BASE PROTOCOL
    AVPs[272].setValues("Multi-Round-Time-Out","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[264].setValues("Origin-Host","DiameterIdentity"); // DIAMETER BASE PROTOCOL
    AVPs[296].setValues("Origin-Realm","DiameterIdentity"); // DIAMETER BASE PROTOCOL
    AVPs[278].setValues("Origin-State-Id","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[269].setValues("Product-Name","UTF8String"); // DIAMETER BASE PROTOCOL
    AVPs[280].setValues("Proxy-Host","DiameterIdentity"); // DIAMETER BASE PROTOCOL
    AVPs[33].setValues("Proxy-State","OctetString"); // DIAMETER BASE PROTOCOL
    AVPs[284].setValues("Proxy-Info","G"); // DIAMETER BASE PROTOCOL
    AVPs[292].setValues("Redirect-Host","DiameterIdentity"); // DIAMETER BASE PROTOCOL
    AVPs[261].setValues("Redirect-Host-Usage","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[262].setValues("Redirect-Max-Cache-Time","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[268].setValues("Result-Code","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[282].setValues("Route-Record","DiameterIdentity"); // DIAMETER BASE PROTOCOL
    AVPs[263].setValues("Session-Id","UTF8String"); // DIAMETER BASE PROTOCOL
    AVPs[27].setValues("Session-Timeout","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[270].setValues("Session-Binding","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[271].setValues("Session-Server-Failover","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[286].setValues("Source-Route","DiameterIdentity"); // DIAMETER BASE PROTOCOL
    AVPs[265].setValues("Supported-Vendor-Id","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[295].setValues("Termination-Cause","Unsigned32"); // DIAMETER BASE PROTOCOL
    */
    AVPs[1].setValues("User-Name","UTF8String"); // DIAMETER BASE PROTOCOL
    /*
    AVPs[266].setValues("Vendor-Id","Unsigned32"); // DIAMETER BASE PROTOCOL
    AVPs[260].setValues("Vendor-Specific-Application-Id","G"); // DIAMETER BASE PROTOCOL
    AVPs[274].setValues("Auth-Request-Type","G"); // DIAMETER BASE PROTOCOL
    AVPs[285].setValues("Re-Auth-Request-Type","Enumerated"); // DIAMETER BASE PROTOCOL
    AVPs[297].setValues("Experimental-Result","G"); // DIAMETER BASE PROTOCOL
    AVPs[298].setValues("Experimental-Result-Code","Unsigned32"); // DIAMETER BASE PROTOCOL
    */
    AVPs[99999].setValues("Test-AVP","UTF8String"); // ACCOUNTING MODULE TEST-AVP
    AVPs[10000].setValues("Accounting-CPUUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10001].setValues("Accounting-DiskUsage-AVP","Unsigned64"); // ACCOUNTING MODULE
    AVPs[10002].setValues("Accounting-EndTime-AVP","UTF8String"); // ACCOUNTING MODULE
    AVPs[10003].setValues("Accounting-HostName-AVP","UTF8String"); // ACCOUNTING MODULE
    AVPs[10004].setValues("Accounting-JobName-AVP","UTF8String"); // ACCOUNTING MODULE
    AVPs[10005].setValues("Accounting-MachineName-AVP","UTF8String"); // ACCOUNTING MODULE
    AVPs[10006].setValues("Accounting-MemoryUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10007].setValues("Accounting-NetworkUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10008].setValues("Accounting-NodeCount-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10013].setValues("Accounting-ServiceLevelQuality-AVP","UTF8String"); // ACCOUNTING
MODULE
    AVPs[10014].setValues("Accounting-StartTime-AVP","UTF8String"); // ACCOUNTING MODULE
    AVPs[10015].setValues("Accounting-Status-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10016].setValues("Accounting-SubmitHost-AVP","UTF8String"); // ACCOUNTING MODULE
    AVPs[10017].setValues("Accounting-SwapUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10018].setValues("Accounting-TempUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10009].setValues("Accounting-ProcessId-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10010].setValues("Accounting-ProcessorCount-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10011].setValues("Accounting-QueueName-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[10012].setValues("Accounting-ScratchUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
    AVPs[20000].setValues("Accounting-Application-Id-AVP","Unsigned32"); // ACCOUNTING MODUL
  }

 public:
  // This function delivers back the name of an AVP in respect on the AVP-ID
  // e.g. getNameOfAVP(20000) ==> Accounting-Parameter-Id-AVP
  char* getNameOfAVP(int id) {
    // Return the Name of the AVP with the index id
    return AVPs[id].getName();
  }

  // This function delivers back the parametertype of an AVP in respect on the AVP-ID
  // e.g. getTypeOfAVP(20000) ==> Unsigned32
  string getTypeOfAVP(int id) {
    // Return the Type of the AVP with the index parameter
```

```
    return AVPs[id].getType();
   }
};

// The class AAA_AcctModuleRecStorage is derived from AAA_ServerAcctRecStorage and
// is used to handle all the AVPs and check if there is space available on the device
class AAA_AcctModuleRecStorage : public AAA_ServerAcctRecStorage {
 private:

   // Create an AAA_AccountingDatabase object
   // This object will be used to store data into the database
   // and also to check if there is enough space available on the database
   // server (or database itself)
   AAA_AccountingDatabase database;

   // Create an AAA_AVPDatabase object to find out which name and type and
   // AVP has based on it's ID
   AAA_AVPDatabase avpDatabase;

   // Define a variable for the avpName which will be sent by the client
   char* avpName;

   // Define a variable for the avpType (parameter) which will be sent by the client
   string avpType;

 public:
   // This function checks the server application if there is
   // enough storage space to hold the next record. We check this
   // by asking the AccountingDatabase-Part.
   virtual bool IsSpaceAvailableOnDevice() {
     // Check what our database-interface tells us and return that unmodified...
     return database.IsSpaceAvailableOnDatabase();
   }

   // This function is called to ask the application to store the newly
   // arrived record (AVPs).
   // Hint: Be aware of application-specific AVPs!
   // List contains all the relevant avp's in the ACR message such as
   // Accounting-Sub-Session-Id, Acct-Session-Id etc. which the application
   // may wish to track.
   virtual void StoreRecord(AAAAvpContainerList &avpList, int recordType, int recordNum) {

     // Variable for checking if the username was received
     bool userOK= false;
     // Variable for checking if the accounting-parameter was received
     bool acctApplOK = false;
     // Variable for checking if the value was received
     bool valueOK = false;
     // Variable for the received value
     string value;

/**** THIS PART IS USED ONLY FOR TESTING ****/
     // Create UTF8AVP-Container for the Session-Id
     AAA_Utf8AvpContainerWidget SessionId(avpList);
     // Get the content of the AVP named "Session-Id"
     diameter_utf8string_t *SessionIdrec = SessionId.GetAvp("Session-Id");
     // If there is content (if the AVP was present...)
     if (SessionIdrec) {
       // Send some informations tn console...
       std::cout << "===> SessionId: " << *SessionIdrec << std::endl;
     }

     // Create Usigned64-Container for the Accounting-Sub-Session-Id
     AAA_UInt64AvpContainerWidget AccountingSubSessionId(avpList);
     // Get the content of the AVP named "Accounting-Sub-Session-Id"
     diameter_unsigned64_t *AccountingSubSessionIdrec = AccountingSubSessionId.GetAvp("Account-
ing-Sub-Session-Id");
     // If there is content (if the AVP was present...)
     if (AccountingSubSessionIdrec) {
       // Send some informations tn console...
```

```
      std::cout << "===> AccountingSubSessionId: " << *AccountingSubSessionIdrec << std::endl;
    }
/**** END OF TESTING PART ****/

    // Create UTF8AVP-Container for the userAvp (username)
    AAA_Utf8AvpContainerWidget userAvp(avpList);
    // Get the content of the AVP named "User-Name"
    diameter_utf8string_t *userIDrec = userAvp.GetAvp("User-Name");
    // If there is content (if the AVP was present...)
    if (userIDrec) {
      // Send some informations tn console...
      std::cout << "User-Name transmitted by client: " << *userIDrec << std::endl;
      // User-name successfully received
      userOK = true;
    }

    // Create Unsigned32AVP-Container for the parameter-ID (AVP-ID)
    AAA_UInt32AvpContainerWidget applAvp(avpList);
    // Get the content of the AVP named "Accounting-Application-Id-AVP"
    diameter_unsigned32_t *applIDrec = applAvp.GetAvp("Accounting-Application-Id-AVP");
    // If there is content (if the AVP was present...)
    if (applIDrec) {
      // Send some informations tn console...
      std::cout << "Application-ID transmitted by client: " << *applIDrec << std::endl;
      // Accounting-Application-ID successfully received
      acctApplOK = true;

      // Fills all the AVPs which could be present in case of this application
      // into an array...
      ApplicationAvps avps = database.getApplicationAvps(*applIDrec);
      // Set the number of AVPs (which could be present) to the variable arraySize
      int arraySize = avps.getLength();

      // For all AVPs
      for(int i=0; i<arraySize;i++) {
        // Get the type of AVP (datatype) based on the AVP-ID
        avpType = avpDatabase.getTypeOfAVP(avps.getAvp(i));
        // Send some informations tn console...
        std::cout << "Datatyp of AVP: " << avpType << std::endl;
        // Get the name of AVP based on the AVP-ID
        avpName = avpDatabase.getNameOfAVP(avps.getAvp(i));
        // Send some informations tn console...
        std::cout << "Name of AVP: " << avpName << std::endl;

        // If te AVP-Datatype is UTF8String
        if(avpType=="UTF8String") {
          //std::cout << "UTF8String" << std::endl;
          // Create UTF8AVP-Container for the data
          AAA_Utf8AvpContainerWidget dataAvp(avpList);
          // Get the content of the AVP
          diameter_utf8string_t *datarec = dataAvp.GetAvp(avpName);
          // If there is content (if the AVP was present...)
          if (datarec) {
            // Send some informations tn console...
            std::cout << "AVP "<< avpName << " transmitted by client with value: " << *datarec
<< std::endl;
            // Map the content to the variable value
            value = *datarec;
            // Data successfully received
            valueOK = true;
          }
        }
        // If te AVP-Datatype is Unsigned32
        else if(avpType=="Unsigned32") {
          //std::cout << "Unsigned32" << std::endl;
          // Create Unsigned32AVP-Container for the data
          AAA_UInt32AvpContainerWidget dataAvp(avpList);
          // Get the content of the AVP
          diameter_unsigned32_t *datarec = dataAvp.GetAvp(avpName);
          // If there is content (if the AVP was present...)
```

```
          if (datarec) {
            // Send some informations tn console...
            std::cout << "AVP "<< avpName << " transmitted by client with value: " << *datarec
<< std::endl;
            // Convert the content to string and map them to the variable value
            stringstream ss;
            ss << *datarec;
            ss >> value;
            // Data successfully received;
            valueOK = true;
          }
        }
        // If te AVP-Datatype is Unsigned64
        else if(avpType=="Unsigned64") {
          //std::cout << "Unsigned64" << std::endl;
          // Create Unsigned64AVP-Container for the data
          AAA_UInt64AvpContainerWidget dataAvp(avpList);
          // Get the content of the AVP
          diameter_unsigned64_t *datarec = dataAvp.GetAvp(avpName);
          // If there is content (if the AVP was present...)
          if (datarec) {
            // Send some informations tn console...
            std::cout << "AVP "<< avpName << " transmitted by client with value: " << *datarec
<< std::endl;
            // Convert the content to string and map them to the variable value
            stringstream ss;
            ss << *datarec;
            ss >> value;
            // Data successfully received
            valueOK = true;
          }
        }
        // If te AVP-Datatype is something else...
        else {
          // Send some informations tn console...
          std::cout << "Unable to proccess this type of value..." << std::endl;
        }
        // If all the required data was successfully received
        if(userOK && acctApplOK && valueOK) {
          // store the received data into the database...
          database.StoreIntoDatabase(*userIDrec, avpName, value);
        }
        else {
          // Send some informations to console...
          std::cout << "An error occured while proccessing data..." << std::endl;
        }
      }
    }
  }
};

// the class AAA_AcctModuleServer is derived from AAA_ServerAcctSession.
// It provides all the functionality of a diameter accounting server session.
// The server session factory is responsible for instantiating  this object.
// AAA_ServerAcctSession is also a template class that requires an
// AAA_ServerAcctRecStorage derived class (in our case AAA_AcctModuleRecStorage
// as a parameter. We implemented some parameter changes and some intelligence
// in case of failures.
class AAA_AcctModuleServer : public AAA_ServerAcctSession<AAA_AcctModuleRecStorage> {
 public:
   // Constructor whicht dictates whether this session is stateful
   AAA_AcctModuleServer(AAA_Task &task, diameter_unsigned32_t id) :
AAA_ServerAcctSession<AAA_AcctModuleRecStorage>(task, id, true) {
     // empty
   }

   // This function is used for setting realtime required AVP as a hint to
   // the server
   virtual void SetRealTimeRequired(AAA_ScholarAttribute<diameter_enumerated_t> &rt) {
     // The following are possible values:
```

```
      // AAA_ACCT_REALTIME_DELIVER_AND_GRANT
      // ACCT_REALTIME_GRANT_AND_STORE
      // ACCT_REALTIME_GRANT_AND_LOSE
      rt = AAA_ACCT_REALTIME_DELIVER_AND_GRANT;
    }

    // This function is used for setting acct interim interval AVP dictated
    // by the server to client
    virtual void SetInterimInterval(AAA_ScholarAttribute<diameter_unsigned32_t> &timeout){
      // tell client to generate record every 2 sec
      timeout = 2;
    }

    // This function is called in case of an successfully exchange
    virtual AAAReturnCode Success() {
      // notification of successful ACR exchange for all record type
      // store the informations into the AAA-LOG
      AAA_LOG(LM_INFO, "(%P|%t) **** record exchange completed ****\n");
      return (AAA_ERR_SUCCESS);
    }

    // This function is called in case of an failed exchange
    virtual AAAReturnCode Failed(int recNum) {
      // notification that recNum record was not processed properly
      // store the informations into the AAA-LOG
      AAA_LOG(LM_INFO, "(%P|%t) **** record #%d not processed ****\n", recNum);
      return (AAA_ERR_SUCCESS);
    }

    // This function is called in case of an session timeout
    virtual AAAReturnCode SessionTimeout() {
      // notification of session timeout if this session was stateful
      // store the informations into the AAA-LOG
      AAA_LOG(LM_INFO, "(%P|%t) **** session timeout ****\n");
      return (AAA_ERR_SUCCESS);
    }
};

// Server session factory. Unlike AAA clients, server
// sessions need to be created on demand. This factory
// is responsible for creating new server sessions
// based on incomming new request.
typedef AAA_ServerSessionAllocator<AAA_AcctModuleServer> AcctModuleServerAllocator;

// This class is the heart of the Module...
// In this class all the Accounting-Server-Stuff will be initialized and
// startet. So the Server is after instantiating this Class and calling
// the function StartAccountingServer ready for receiving DiameterMessages.
class AAA_AccountingServerModule {
 public :
   // Defines if the Server runs or not
   bool running;

   // This function starts the accounting-server
   int StartAccountingServer(char* serverConfigFile, int accountingApplicationID) {
     // running the server = true
     // this variable could be modified through the implementation of
     // an additional thread and function... => as abort-statement
     running = true;
     // Create an AAA_Task object
     AAA_Task task;
     // call the Start-Method from the new generated task-object
     task.Start(5);

     // The Application core is responsible for providing
     // peer connectivity between the AAA entities
     // So let's create an AAA_Application object
     AAA_Application appCore(task, serverConfigFile);
     // Create an AcctModuleServerAllocator object which represents the
     // ServerSessionFactory
```

```
    AcctModuleServerAllocator allocator(task, accountingApplicationID);
    // Now register the ServerSesscionFactoryObject by the ApplicationCore
    appCore.RegisterServerSessionFactory(allocator);

    // Send some informations to console
    std::cout << "AAA-Accounting-Server started..." << std::endl;
    // Send some informations to console
    std::cout << "Waiting for Client-Sessions..." << std::endl;

    // Loop until we tell the Server to stop waiting on client-sessions
    // Should be implemented in later versions to define an abort-statement
    // eg user interaction or something else...
    while (running) {
      // Send some informations to console
      std::cout << "Just wait here and let factory take care of new sessions..." << std::endl;
      // wait some seconds
      ACE_OS::sleep(10);
    }

    // Close/Terminate the ApplicationCore
    appCore.Close();
    // Stop the AAA_Task
    task.Stop();
    // Return termination code (unix-style)
    return (0);
   }
};

// This is the main-function of the accounting_server-Module
// With this function we create the needed AAA_AccountingServerModule object
// Also some config-stuff will be sent to the object...
int main(int argc, char *argv[]) {
 // Define which config-file should be used
 char* serverConfigFile = "config/server.xml";
 // Define which application (see also dictionary.xml) should be used
 int accountingApplicationID = 29999;
 // Create an object from type AAA_AccountingServerModule
 AAA_AccountingServerModule aaaAcctSrvModule;
 // Send some informations tn console...
 std::cout << "Starting AAA-Accounting-Server..." << std::endl;
 // Call the function StartAccountingServer from the new generated
 // object aaaAcctSrvModule with the configuration-parameters
 aaaAcctSrvModule.StartAccountingServer(serverConfigFile, accountingApplicationID);
}
```

D.2 accounting-client.cxx

```
/* BEGIN_COPYRIGHT Open Diameter                                          */
/*                                                                        */
/* Open Diameter: Open-source software for the Diameter and               */
/*                Diameter related protocols                              */
/*                                                                        */
/* Copyright (C) 2002-2004 Open Diameter Project                          */
/*                                                                        */
/* This library is free software; you can redistribute it and/or modify   */
/* it under the terms of the GNU Lesser General Public License as         */
/* published by the Free Software Foundation; either version 2.1 of the   */
/* License, or (at your option) any later version.                        */
/*                                                                        */
/* This library is distributed in the hope that it will be useful,        */
/* but WITHOUT ANY WARRANTY; without even the implied warranty of         */
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU      */
/* Lesser General Public License for more details.                        */
/*                                                                        */
/* You should have received a copy of the GNU Lesser General Public       */
/* License along with this library; if not, write to the Free Software    */
/* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307    */
/* USA.                                                                   */
/*                                                                        */
```

```
/* In addition, when you copy and redistribute some or the entire part of */
/* the source code of this software with or without modification, you     */
/* MUST include this copyright notice in each copy.                      */
/*                                                                       */
/* If you make any changes that are appeared to be useful, please send   */
/* sources that include the changed part to                              */
/* diameter-developers@lists.sourceforge.net so that we can reflect your */
/* changes to one unified version of this software.                      */
/*                                                                       */
/* END_COPYRIGHT Open Diameter                                           */
/*                                                                       */
/* BEGIN_COPYRIGHT Accounting Module                                     */
/*                                                                       */
/* Copyright (c) 2005 by Adrian Bachmann, adrian@bachmann.lu             */
/* This Accounting Module was programmed as a semester thesis            */
/* DESIGN AND PROTOTYPICAL IMPLEMENTATION OF AN ACCOUNTING SYSTEM FOR    */
/* AN AAA SERVER                                                         */
/* at the Institute of Informatics, University of Zurich                 */
/*                                                                       */
/* Author: Adrian Bachmann, adrian@bachmann.lu                           */
/* Supervisor: Cristian Morariu, morariu@ifi.unizh.ch                    */
/* Director: Prof. Dr. Burkhard Stiller, stiller@ifi.unizh.ch            */
/*                                                                       */
/* END_COPYRIGHT Accounting Module                                       */

// ACCOUNTING CLIENT

// INCLUDE THE DIAMETER-API-HEADER-FILE
#include "diameter_api.h"
// INCLUDE THE STRING LIBRARY
#include <string>
// INCLUDE THE STRINGSTREAM LIBRARY FOR CASTING THE DATA TO STRING
#include <sstream>

// initialization of strings
using namespace std;

// This class represents AVPs as objects
class AAA_AVP {
 public:
   // This function sets the values of an AVP
   void setValues(char* n, string t) {
     // Set the Name of AVP
     name = n;
     // Set the Type of AVP
     type = t;
   }

   // This function returns the name of the object (AVP)
   char* getName() {
     // return the name of the AVP
     return name;
   }

   // This function returns the type of the object (AVP)
   string getType() {
     // return the datatype of the AVP
     return type;
   }
 private:
   // Name of AVP e.g. "Destination-Host"
   char* name;
   // Type of AVP e.g. "DiameterIdentity"
   string type;
};

// The class AAA_AVPDatabase is used to store all the AVP-names and AVP-datatypes
// which corresponds to an AVP-ID
class AAA_AVPDatabase{
 public:
```

```
   // Constructor which initializes automaticaly all the AVPs through the
   // funcion initAVPs()
   AAA_AVPDatabase() {
     // Initialize all the AVPs (manually)...
     initAVPs();
   }
 private:
   // Lets create an array with all the mapped AVPs (as objects) in it
   // The index corresponds to the AVP-ID
   AAA_AVP AVPs[100000];

   // Fills all the AVPs into the AVPs-Array
   // In this prototype we fill them all manually... later that should
   // be done based on the dictionary.xml-file automaticaly and directly
   void initAVPs() {
     AVPs[1].setValues("User-Name","UTF8String"); // DIAMETER BASE PROTOCOL
     AVPs[99999].setValues("Test-AVP","UTF8String"); // ACCOUNTING MODULE TEST-AVP
     AVPs[10000].setValues("Accounting-CPUUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10001].setValues("Accounting-DiskUsage-AVP","Unsigned64"); // ACCOUNTING MODULE
     AVPs[10002].setValues("Accounting-EndTime-AVP","UTF8String"); // ACCOUNTING MODULE
     AVPs[10003].setValues("Accounting-HostName-AVP","UTF8String"); // ACCOUNTING MODULE
     AVPs[10004].setValues("Accounting-JobName-AVP","UTF8String"); // ACCOUNTING MODULE
     AVPs[10005].setValues("Accounting-MachineName-AVP","UTF8String"); // ACCOUNTING MODULE
     AVPs[10006].setValues("Accounting-MemoryUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10007].setValues("Accounting-NetworkUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10008].setValues("Accounting-NodeCount-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10013].setValues("Accounting-ServiceLevelQuality-AVP","UTF8String"); // ACCOUNTING
MODULE
     AVPs[10014].setValues("Accounting-StartTime-AVP","UTF8String"); // ACCOUNTING MODULE
     AVPs[10015].setValues("Accounting-Status-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10016].setValues("Accounting-SubmitHost-AVP","UTF8String"); // ACCOUNTING MODULE
     AVPs[10017].setValues("Accounting-SwapUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10018].setValues("Accounting-TempUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10009].setValues("Accounting-ProcessId-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10010].setValues("Accounting-ProcessorCount-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10011].setValues("Accounting-QueueName-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[10012].setValues("Accounting-ScratchUsage-AVP","Unsigned32"); // ACCOUNTING MODULE
     AVPs[20000].setValues("Accounting-Application-Id-AVP","Unsigned32"); // ACCOUNTING MODUL
   }

 public:
   // This function delivers back the name of an AVP in respect on the AVP-ID
   // e.g. getNameOfAVP(20000) ==> Accounting-Parameter-Id-AVP
   char* getNameOfAVP(int id) {
     // Return the Name of the AVP with the index id
     return AVPs[id].getName();
   }

   // This function delivers back the parametertype of an AVP in respect on the AVP-ID
   // e.g. getTypeOfAVP(20000) ==> Unsigned32
   string getTypeOfAVP(int id) {
     // Return the Type of the AVP with the index parameter
     return AVPs[id].getType();
   }
};

// This class represents an single datarecord like a username
class AAA_Record {
 private:
   // this variable represents the AVP-ID
   int avpid;
   // this string represents the value
   string value;

 public:
   // standard constructor
   AAA_Record() {
     // EMPTY
   }
```

```
   // constructor for string-values
   AAA_Record(int id, string val){
     // set the avp-id
     avpid = id;
     // set the value
     value = val;
   }

   // constructor for int values
   AAA_Record(int id, int val) {
     // set the avp-id
     avpid = id;
     // cast the int-value to a string-value
     // and set them to value
     stringstream ss;
     ss << val;
     ss >> value;
   }

   // this function returns the AVP-ID of the object
   int getAvpid() {
     // return the AVP-ID
     return avpid;
   }

   // this function returns the value of the object
   string getValue() {
     // return the value
     return value;
   }
};

// this class represents all the records which should be sent through a client
class AAA_Records {
 private:
   // Array in which all the Records will be stored
   // HINT: The maximum number of records is 100
   AAA_Record records[100];
   // saves the number of records represented with this object
   int numofrecords;

 public:
   // constructor which sets the initial value to zero
   AAA_Records() {
     // initial value => no records stored
     numofrecords = 0;
   }

   // this function is used to add new records with a int-parameter
   bool addRecord(int id, int param) {
     // if there are already more than 99 records
     if(numofrecords>99){
       // return an error
       std::cout << "Too many records in this message..." << std::endl;
       return false;
     }
     // everything ok
     else {
       // store the new record
       records[numofrecords] = AAA_Record(id, param);
       // increment the number of records
       numofrecords++;
       // return success
       return true;
     }
   }

   // this function is used to add new records with a string-parameter
   bool addRecord(int id, string param) {
     // if there are already more than 99 records
```

```
     if(numofrecords>99){
       // return an error
       std::cout << "Too many records in this message..." << std::endl;
       return false;
     }
     // everything ok
     else {
       // store the new record
       records[numofrecords] = AAA_Record(id, param);
       // increment the number of records
       numofrecords++;
       // return success
       return true;
     }
   }

   // this function returns the number of stored records
   int getNumberOfRecords() {
     // return number of records
     return numofrecords;
   }

   // this function returns a defined record
   AAA_Record getRecord(int num) {
     // return record with index num
     return records[num];
   }
};

// This class is responsible for collecting all the data needed to send a
// diameter message
class AAA_AcctModuleClientAcctRecCollector : public AAA_ClientAcctRecCollector {
 public:
   // create an AAA_Records-object
   AAA_Records records;

   // this function is called by the client to set the records, which should be transfered
   void setRecords(AAA_Records rec) {
     // send some informations to consoole
       std::cout << "Records overtaken from AAA_AcctModuleClientAcctRecCollector..." <<
std::endl;
     // set the object to the just arrived object
     records = rec;
     // send some informations to consoole
     std::cout << "Number of records received for transmission: " << records.getNumberOf-
Records() << std::endl;
   }


   // This function is called by the library to ask the client application
   // to generate records that will be stored in vendor specific AVPs. These
   // AVPs will then be added to the avpList before sending the ACR
   virtual void GenerateRecord(AAAAvpContainerList &avpList, int recordType, int recordNum) {
     // this variable stores the number of recors to proccess
     int length = records.getNumberOfRecords();
     // this variable stores temporarily the ID of the current proccessing AVP
     int avpid;
     // this variable stores temporarily the value of the current proccessing AVP
     string value;
     // this variable stores temporarily the type of the current proccessing AVP
     string avptype;
     // this variable stores temporarily the name of the current proccessing AVP
     char* avpname;

     // create an AVPDatabase-object to get all the needed informations about the AVPs
     AAA_AVPDatabase avpdb;

     // For all records (see object records)
     for (int i = 0; i<length; i++) {
       // get the ID
```

```
      avpid = records.getRecord(i).getAvpid();
      // get the value (always as string)
      value = records.getRecord(i).getValue();
      // get the avp-type
      avptype = avpdb.getTypeOfAVP(avpid);
      // get the avp-name
      avpname = avpdb.getNameOfAVP(avpid);

      // If the avp-type is UTF8String
      if(avptype=="UTF8String") {
        // Create AVP Widget for the data of AVP
        AAA_Utf8AvpWidget dataAvp(avpname);
        // set the data
        dataAvp.Get() = value;
        // add the AVP to the avpList
        avpList.add(dataAvp());
        // send some informations to consoole
        std::cout << "AVP #" << i << " Name: " << avpname << " Data: " << value << std::endl;
      }
      // If the avp-type is Unsigned32
      else if(avptype=="Unsigned32") {
        // create AVP Widget for the data of AVP
        AAA_UInt32AvpWidget dataAvp(avpname);
        // convert the data from string to *char to int
        int intvalue = atof(value.c_str());
        // set the data
        dataAvp.Get() = intvalue;
        // add the AVP to the avpList
        avpList.add(dataAvp());
        // send some informations to consoole
        std::cout << "AVP #" << i << " Name: " << avpname << " Data: " << value << std::endl;
      }
      // If the avp-type is Unsigned64
      else if(avptype=="Unsigned64") {
        // Create AVP Widget for the data of AVP
        AAA_UInt64AvpWidget dataAvp(avpname);
        // convert the data from string to *char to int
        int intvalue = atof(value.c_str());
        // set the data
        dataAvp.Get() = intvalue;
        // add the AVP to the avpList
        avpList.add(dataAvp());
        // send some informations to consoole
        std::cout << "AVP #" << i << " Name: " << avpname << " Data: " << value << std::endl;
      }
      // In all other cases
      else {
        // send some informations to consoole
        std::cout << "Unknown datatype... couldn't proccess record #"<< i << std::endl;
      }
    }
  }

  // Checks the client app if there is a a record
  // that has temporarily been stored due to prior
  // transmission failure. The check should be
  // based on application specific storage
  virtual bool IsLastRecordInStorage() {
    // Not yet properly implemented
    return (false);
  }

  // Checks the client app if it will be able
  // to store a record in case transmission
  // fails.
  virtual bool IsStorageSpaceAvailable() {
    // Not yet properly implemented
    return (true);
  }
```

```
   // Asks the client app to temporarily
   // store the last known record. This maybe
   // due to transmission and the library is
   // attempting to retry. Applications MUST
   // provide thier own storage here
   virtual AAAReturnCode StoreLastRecord(int recordType) {
     // Not yet properly implemented
     return (AAA_ERR_SUCCESS);
   }

   /// Asks the client app to delete the
   /// last temporarily stored record if any
   virtual AAAReturnCode DeleteLastRecord(int recordType) {
     // Not yet properly implemented
     return (AAA_ERR_SUCCESS);
   }
};

// AAA client session derived from AAA_ClientAcctSession.
// It provides for all the functionality of a diameter
// client accounting session. The ClientAcctSubSession
// class is a template function that requires a proper
// version of a record collector as a paramter. Note
// that the application is responsible for instantiating
// this object
class AAA_AcctModuleClientSubSession : public
AAA_ClientAcctSubSession<AAA_AcctModuleClientAcctRecCollector> {

 public:
   // Constructor
   AAA_AcctModuleClientSubSession(AAA_ClientAcctSession &parent) :
AAA_ClientAcctSubSession<AAA_AcctModuleClientAcctRecCollector>(parent),
m_HowManyRecProcessed(0) {
   }

   // optional override, called by the library to
   // set the destination host. Note that this
   // overrides applications sending a destination
   // host AVP
   virtual void SetDestinationHost(AAA_ScholarAttribute<diameter_identity_t> &dHost){
     // set the destination host of this client-module
     dHost = "server.aaaserver.net";
   }

   // optional override, called by the library
   // to set the destination realm. Note that
   // this overrides applications sending a
   // destination realm AVP
        virtual void SetDestinationRealm(AAA_ScholarAttribute<diameter_identity_t> &dRealm){
     // set the destination realm of this client-module
     dRealm = "aaaserver.net";
   }

   // This function is used for setting realtime required
   // AVP as a hint to the server
   virtual void SetRealTimeRequired(AAA_ScholarAttribute<diameter_enumerated_t> &rt){
   }

   // This function is used for setting acct interim
   // interval AVP as a hint to the server
   virtual void SetInterimInterval(AAA_ScholarAttribute<diameter_unsigned32_t> &timeout){
   }

   // This function is used for setting RADIUS acct
   // session id for RADIUS/DIAMETER translations
   virtual void SetRadiusAcctSessionId(AAA_ScholarAttribute<diameter_octetstring_t> &sid){
   }

   // This function is used for setting multi-session
   // id AVP
```

```
   virtual void SetMultiSessionId(AAA_ScholarAttribute<diameter_utf8string_t> &sid){
   }

   // notification of successful ACR exchange for all record type
   virtual AAAReturnCode Success() {
     // write some informations into logfile
     AAA_LOG(LM_INFO, "(%P|%t) **** record exchange completed ****\n");
     // increment the number of processed records
     m_HowManyRecProcessed ++;
     // return SUCCESS
     return (AAA_ERR_SUCCESS);
   }

   // notification that recNum record was not processed properly
   virtual AAAReturnCode Failed(int recNum) {
     // write some informations into logfile
     AAA_LOG(LM_INFO, "(%P|%t) **** record #%d not processed ****\n", recNum);
     // return SUCCESS
     return (AAA_ERR_SUCCESS);
   }
   // returns the number of proccessed records
   int NumberOfRecordsProcessed() {
     return m_HowManyRecProcessed;
   }
 private:
   // stores the number of proccessed records
   int m_HowManyRecProcessed;
};

// This class is the heart of the Application-Module
class AAA_AccountingClientModule {
 private:
   // placeholder for the config-file
   char* cfgFile;
   // placeholder fot the application-ID
   int acctApplID;
   // create an AAA_Task-object
   AAA_Task task;
   // create an AAA_Application-pointer
   AAA_Application *appCore;
   // create an AAA_ClientAcctSession-pointer
   AAA_ClientAcctSession *parent;
   // Create an array of subsessions to store all the created Subsessions
   AAA_AcctModuleClientSubSession *subSessions[100000];
   // Stores the actual subsession-number
   int subSessionId;

 public:
   AAA_AccountingClientModule(char* configFile, int accountingApplID){
     // set the variable cfgFile
     cfgFile = configFile;
     // set the variable acctApplID
     acctApplID = accountingApplID;
     // Start the task
     task.Start(5);
     // Application core is responsible for providing
     // peer connectivity between AAA entities
     appCore = new AAA_Application(task);
     // If the config-file could be opened
     if (appCore->Open(cfgFile) == AAA_ERR_SUCCESS) {
       // Wait for connectivity (until we have conenctivity)
       do {
         // send some informations to console
         std::cout << "Waiting till this AAA-Client has connectivity" << std::endl;
         // wait one second...
         ACE_OS::sleep(1);
       } while (appCore->NumActivePeers() == 0);
       /// Each accounting session can have multiple
       /// sub-sessions. The main/parent session has
       /// the Session-Id and each sub session has
```

```
      /// an Accounting-Sub-Session-Id
      parent = new AAA_ClientAcctSession(task, acctApplID);
      // send some informations to console
      std::cout << "Connection successfully etablished... " << std::endl;
    }
    // if we were unable to open the config-file
    else {
      // send some informations to console
      std::cout << "An error occured while etablishing the connection..." << std::endl;
    }
  }
  // Destructor
  ~AAA_AccountingClientModule() {
      // send some informations to console
    std::cout << "Destroy AAA_AccountingClientModule..." << std::endl;
    // close the Application-Core
    appCore->Close();
    // stop the task
    task.Stop();
  }

  // This function creates a new subsession and returns the number of the subsession
  int createNewSubSession() {
      // if there are not more subsessions created than space available
    // in the subsessions-array available
    if(subSessionId < 100000) {
      // create new subsession at the given index
      subSessions[subSessionId] = new AAA_AcctModuleClientSubSession(*parent);
       // Increment the subsession-number for a next subsession
      subSessionId++;
      // return the number of the new created subsession
       return subSessionId-1;
    }
    // else
    else {
      // send some informations to console
      std::cout << "An error occured while creating an new subsession: there are too many sub-
sessions created!" << std::endl;
    }
  }

  // This function removes an open subsession
  void removeSubSession(int session) {
      // close the subsession with index session
    subSessions[session]->End();
  }

  // This function is used to send the data stored in a AAA_Records-object
  bool sendData(AAA_Records records, int session) {
    // Set the records in the RecordCollector in a given subsession
    subSessions[session]->RecCollector().setRecords(records);
    // Start the given sub-session
    subSessions[session]->Begin(true);
    // we have only one message to process
    int numMsgToWaitFor = 1;
    // wait until all messages are processed
    do {
      // send some informations to console
      std::cout << "Waiting till records are processed" << std::endl;
      // wait one second
      ACE_OS::sleep(1);
    } while (subSessions[session]->NumberOfRecordsProcessed() < numMsgToWaitFor);
    // return success if we arrive here
    return true;
  }

  // this function is used to create an AAA_Records-object in which
  // the client should store all the collected data...
  AAA_Records createRecords(int applId, string username) {
    // defines the AVP-ID of the Application-Id-AVP
```

```
    int applIdAvp = 20000;
    // defined the AVP-ID of the User-Name-AVP
    int usernameAvp = 1;
    // Create an new AAA_Records-object
    AAA_Records records;
    // Add the Application-ID to records
    records.addRecord(applIdAvp, applId);
    // Add the username to records
    records.addRecord(usernameAvp, username);
    // Return the records-object which already contains two AVPs...
    return records;
  }

};

// This is the main-function of the accounting_client-Module
// With this function we create the needed AAA_AccountingClientModule object
// Also some config-stuff will be sent to the object and some data createt for
// transmission....
int main(int argc, char *argv[]) {
 // Define which config-file should be used
 char* clientConfigFile = "config/client.xml";
 // Define which application (see also dictionary.xml) should be used
 int accountingApplicationID = 29999;
 // Create an object from type AAA_AccountingClientModule
 AAA_AccountingClientModule aaaAcctClientModule(clientConfigFile, accountingApplicationID);

 int testsession = aaaAcctClientModule.createNewSubSession();

 for(int i=0; i<5;i++) {
  // Create an AAA_Recors-object based on the returned AAA_Records-object
  AAA_Records records = aaaAcctClientModule.createRecords(100, "abachmann@access.unizh.ch");
  // Add some data (which creates an new record-entry)
  records.addRecord(10001, 123456);
  // Add some data (which creates an new record-entry)
  records.addRecord(10003, "ARVO.IFI.UNIZH.CH");
  // Add some data (which creates an new record-entry)
  records.addRecord(10008, i);

  // If the data was sent successfully
  if(aaaAcctClientModule.sendData(records, testsession)) {
    // Send some informations to consoole
    std::cout << "Data successfully sent" << std::endl;
  }
  // If an error occured
  else {
    // Send some informations to consoole
    std::cout << "A failure occured while etablishing connection / sending data..." <<
std::endl;
  }
 }

 aaaAcctClientModule.removeSubSession(testsession);

 // retrun UNIX-Style statuscode
 return(0);
}


// This is the main-function of the accounting_client-Module
// With this function we create the needed AAA_AccountingClientModule object
// Also some config-stuff will be sent to the object and some data createt for
// transmission....
int main(int argc, char *argv[]) {
 // Define which config-file should be used
 char* clientConfigFile = "config/client.xml";
 // Define which application (see also dictionary.xml) should be used
 int accountingApplicationID = 29999;
 // Create an object from type AAA_AccountingClientModule
 AAA_AccountingClientModule aaaAcctClientModule(clientConfigFile, accountingApplicationID);
```

```
 for(int i=0; i<10;i++) {
   // Create an AAA_Recors-object based on the returned AAA_Records-object
   AAA_Records records = aaaAcctClientModule.createRecords(100, "abachmann@access.unizh.ch");
   // Add some data (which creates an new record-entry)
   records.addRecord(10001, 123456);
   // Add some data (which creates an new record-entry)
   records.addRecord(10003, "ARVO.IFI.UNIZH.CH");
   // Add some data (which creates an new record-entry)
   records.addRecord(10008, i);

   // If the data was sent successfully
   if(aaaAcctClientModule.sendData(records)) {
     // Send some informations to consoole
     std::cout << "Data successfully sent" << std::endl;
   }
   // If an error occured
   else {
     // Send some informations to consoole
     std::cout << "A failure occured while etablishing connection / sending data..." <<
std::endl;
   }
 }
 // retrun UNIX-Style statuscode
 return(0);
}
```

Anhang E: Übertragungsprotokolle des Prototyps

E.1 Server mit einer neuen Subsession je Diameter-Nachricht

```
./aaa_accounting_server
Starting AAA-Accounting-Server...
(16423|1082546400) Starting diameter core
(16423|1082546400)             Product : Open Diameter
(16423|1082546400)             Version : 1
(16423|1082546400)           Vendor Id : 0
(16423|1082546400)    Supported Vendor : 0
(16423|1082546400)    Supported Vendor : 1
(16423|1082546400)    Supported Vendor : 9999
(16423|1082546400)    Auth Application : 1
(16423|1082546400)    Auth Application : 2
(16423|1082546400)    Auth Application : 19999
(16423|1082546400)    Acct Application : 3
(16423|1082546400)    Acct Application : 4
(16423|1082546400)    Acct Application : 29999
(16423|1082546400)  Vendor Specific Id : Auth=19999, Acct=29999
(16423|1082546400)                        Vendor=9999
(16423|1082546400)          Dictionary : config/dictionary.xml
(16423|1082546400)            Identity : server.aaaserver.net
(16423|1082546400)               Realm : aaaserver.net
(16423|1082546400)          TCP Listen : 1812
(16423|1082546400)          TLS Listen : 0
(16423|1082546400)   Watch-Dog timeout : 4
(16423|1082546400)  Peer Table :
(16423|1082546400)                Peer : Host = client.aaaclient.net, Port = 1811, TLS = 0
(16423|1082546400)  Route Table:
(16423|1082546400)              Route  : Realm = aaaclient.net, Action = 0
(16423|1082546400)                       Application Id=0, Vendor=0
(16423|1082546400)                          Server = client.aaaclient.net, metric = 2
(16423|1082546400)                       Application Id=1, Vendor=0
(16423|1082546400)                          Server = client.aaaclient.net, metric = 2
(16423|1082546400)  Default Route:
(16423|1082546400)              Route  : Realm = aaaclient.net, Action = 0
(16423|1082546400)                       Application Id=0, Vendor=0
(16423|1082546400)                          Server = client.aaaclient.net, metric = 4
(16423|1082546400)       Stateful Auth : 0
(16423|1082546400)          Session(T) : 30
```

```
(16423|1082546400)         Lifetime(T) : 360
(16423|1082546400)            Grace(T) : 30
(16423|1082546400)            Abort(T) : 20
(16423|1082546400)            Max Sess : 10000
(16423|1082546400)          Session(T) : 30
(16423|1082546400)    Interim Interval : 5
(16423|1082546400)  Real-Time Required : 1
(16423|1082546400)           Debug Log : disabled
(16423|1082546400)           Trace Log : disabled
(16423|1082546400)            Info Log : disabled
(16423|1082546400)         Console Log : disabled
(16423|1082546400)          Syslog Log : disabled
AAA-Accounting-Server started...
Waiting for Client-Sessions...
Just wait here and let factory take care of new sessions...
===> SessionId: client.aaaclient.net.aaaclient.net;1125151587;0
===> AccountingSubSessionId: 1
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 0
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 0
===> SessionId: client.aaaclient.net.aaaclient.net;1125151587;0
===> AccountingSubSessionId: 2
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 1
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 1
===> SessionId: client.aaaclient.net.aaaclient.net;1125151587;0
===> AccountingSubSessionId: 3
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 2
```

```
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 2
===> SessionId: client.aaaclient.net.aaaclient.net;1125151587;0
===> AccountingSubSessionId: 4
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 3
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 3
===> SessionId: client.aaaclient.net.aaaclient.net;1125151587;0
===> AccountingSubSessionId: 5
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 4
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 4
Just wait here and let factory take care of new sessions...
```

E.2 Server mit einer einzigen Subsession für die übertragenen Diameter-Nachrichten

```
./aaa_accounting_server
Starting AAA-Accounting-Server...
(17063|1082546400) Starting diameter core
(17063|1082546400)             Product : Open Diameter
(17063|1082546400)             Version : 1
(17063|1082546400)           Vendor Id : 0
(17063|1082546400)    Supported Vendor : 0
(17063|1082546400)    Supported Vendor : 1
(17063|1082546400)    Supported Vendor : 9999
(17063|1082546400)    Auth Application : 1
(17063|1082546400)    Auth Application : 2
(17063|1082546400)    Auth Application : 19999
(17063|1082546400)    Acct Application : 3
(17063|1082546400)    Acct Application : 4
(17063|1082546400)    Acct Application : 29999
(17063|1082546400)  Vendor Specific Id : Auth=19999, Acct=29999
(17063|1082546400)                       Vendor=9999
(17063|1082546400)          Dictionary : config/dictionary.xml
(17063|1082546400)            Identity : server.aaaserver.net
(17063|1082546400)               Realm : aaaserver.net
(17063|1082546400)          TCP Listen : 1812
(17063|1082546400)          TLS Listen : 0
(17063|1082546400)   Watch-Dog timeout : 4
(17063|1082546400)  Peer Table :
```

```
(17063|1082546400)               Peer : Host = client.aaaclient.net, Port = 1811, TLS = 0
(17063|1082546400)  Route Table:
(17063|1082546400)             Route  : Realm = aaaclient.net, Action = 0
(17063|1082546400)                      Application Id=0, Vendor=0
(17063|1082546400)                         Server = client.aaaclient.net, metric = 2
(17063|1082546400)                      Application Id=1, Vendor=0
(17063|1082546400)                         Server = client.aaaclient.net, metric = 2
(17063|1082546400)  Default Route:
(17063|1082546400)             Route  : Realm = aaaclient.net, Action = 0
(17063|1082546400)                      Application Id=0, Vendor=0
(17063|1082546400)                         Server = client.aaaclient.net, metric = 4
(17063|1082546400)       Stateful Auth : 0
(17063|1082546400)          Session(T) : 30
(17063|1082546400)         Lifetime(T) : 360
(17063|1082546400)            Grace(T) : 30
(17063|1082546400)            Abort(T) : 20
(17063|1082546400)            Max Sess : 10000
(17063|1082546400)          Session(T) : 30
(17063|1082546400)    Interim Interval : 5
(17063|1082546400)  Real-Time Required : 1
(17063|1082546400)           Debug Log : disabled
(17063|1082546400)           Trace Log : disabled
(17063|1082546400)            Info Log : disabled
(17063|1082546400)         Console Log : disabled
(17063|1082546400)          Syslog Log : disabled
AAA-Accounting-Server started...
Waiting for Client-Sessions...
Just wait here and let factory take care of new sessions...
===> SessionId: client.aaaclient.net.aaaclient.net;1125152239;0
===> AccountingSubSessionId: 1
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 0
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 0
===> SessionId: client.aaaclient.net.aaaclient.net;1125152239;0
===> AccountingSubSessionId: 1
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 1
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 1
===> SessionId: client.aaaclient.net.aaaclient.net;1125152239;0
===> AccountingSubSessionId: 1
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
```

```
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 2
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 2
===> SessionId: client.aaaclient.net.aaaclient.net;1125152239;0
===> AccountingSubSessionId: 1
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 3
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 3
===> SessionId: client.aaaclient.net.aaaclient.net;1125152239;0
===> AccountingSubSessionId: 1
User-Name transmitted by client: abachmann@access.unizh.ch
Application-ID transmitted by client: 100
Datatyp of AVP: UTF8String
Name of AVP: Accounting-HostName-AVP
AVP Accounting-HostName-AVP transmitted by client with value: ARVO.IFI.UNIZH.CH
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-HostName-AVP Value: ARVO.IFI.UNIZH.CH
Datatyp of AVP: Unsigned64
Name of AVP: Accounting-DiskUsage-AVP
AVP Accounting-DiskUsage-AVP transmitted by client with value: 123456
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-DiskUsage-AVP Value: 123456
Datatyp of AVP: Unsigned32
Name of AVP: Accounting-NodeCount-AVP
AVP Accounting-NodeCount-AVP transmitted by client with value: 4
Values stored in database:
User: abachmann@access.unizh.ch Parameter: Accounting-NodeCount-AVP Value: 4
Just wait here and let factory take care of new sessions...
Just wait here and let factory take care of new sessions...
```

E.3 Client

```
./aaa_accounting_client
(29931|1082546400) Starting diameter core
(29931|1082546400)             Product : Open Diameter
(29931|1082546400)             Version : 1
(29931|1082546400)           Vendor Id : 0
(29931|1082546400)    Supported Vendor : 0
(29931|1082546400)    Supported Vendor : 1
(29931|1082546400)    Supported Vendor : 9999
(29931|1082546400)    Auth Application : 1
(29931|1082546400)    Auth Application : 2
(29931|1082546400)    Auth Application : 19999
```

```
(29931|1082546400)    Acct Application : 3
(29931|1082546400)    Acct Application : 4
(29931|1082546400)    Acct Application : 29999
(29931|1082546400)  Vendor Specific Id : Auth=19999, Acct=29999
(29931|1082546400)                        Vendor=9999
(29931|1082546400)          Dictionary : config/dictionary.xml
(29931|1082546400)            Identity : client.aaaclient.net
(29931|1082546400)               Realm : aaaclient.net
(29931|1082546400)          TCP Listen : 1811
(29931|1082546400)          TLS Listen : 0
(29931|1082546400)   Watch-Dog timeout : 3
(29931|1082546400)  Peer Table :
(29931|1082546400)                Peer : Host = server.aaaserver.net, Port = 1812, TLS = 0
(29931|1082546400)  Route Table:
(29931|1082546400)              Route  : Realm = aaaserver.net, Action = 1
(29931|1082546400)                       Application Id=1, Vendor=0
(29931|1082546400)                          Server = server.aaaserver.net, metric = 2
(29931|1082546400)                       Application Id=0, Vendor=0
(29931|1082546400)                          Server = server.aaaserver.net, metric = 2
(29931|1082546400)  Default Route:
(29931|1082546400)              Route  : Realm = aaaserver.net, Action = 0
(29931|1082546400)                       Application Id=1, Vendor=0
(29931|1082546400)                          Server = server.aaaserver.net, metric = 4
(29931|1082546400)                          Server = server.aaaserver.net, metric = 5
(29931|1082546400)                              *Duplicate server (replacing previous)
(29931|1082546400)       Stateful Auth : 0
(29931|1082546400)          Session(T) : 30
(29931|1082546400)         Lifetime(T) : 360
(29931|1082546400)            Grace(T) : 30
(29931|1082546400)            Abort(T) : 20
(29931|1082546400)            Max Sess : 10000
(29931|1082546400)          Session(T) : 30
(29931|1082546400)    Interim Interval : 5
(29931|1082546400)  Real-Time Required : 1
(29931|1082546400)           Debug Log : disabled
(29931|1082546400)           Trace Log : disabled
(29931|1082546400)            Info Log : disabled
(29931|1082546400)         Console Log : disabled
(29931|1082546400)          Syslog Log : disabled
Waiting till this AAA-Client has connectivity
Connection successfully etablished...
Records overtaken from AAA_AcctModuleClientAcctRecCollector...
Number of records received for transmission: 5
AVP #0 Name: Accounting-Application-Id-AVP Data: 100
AVP #1 Name: User-Name Data: abachmann@access.unizh.ch
AVP #2 Name: Accounting-DiskUsage-AVP Data: 123456
AVP #3 Name: Accounting-HostName-AVP Data: ARVO.IFI.UNIZH.CH
AVP #4 Name: Accounting-NodeCount-AVP Data: 0
Waiting till records are processed
Data successfully sent
Records overtaken from AAA_AcctModuleClientAcctRecCollector...
Number of records received for transmission: 5
AVP #0 Name: Accounting-Application-Id-AVP Data: 100
AVP #1 Name: User-Name Data: abachmann@access.unizh.ch
AVP #2 Name: Accounting-DiskUsage-AVP Data: 123456
AVP #3 Name: Accounting-HostName-AVP Data: ARVO.IFI.UNIZH.CH
AVP #4 Name: Accounting-NodeCount-AVP Data: 1
Waiting till records are processed
Data successfully sent
Records overtaken from AAA_AcctModuleClientAcctRecCollector...
Number of records received for transmission: 5
AVP #0 Name: Accounting-Application-Id-AVP Data: 100
AVP #1 Name: User-Name Data: abachmann@access.unizh.ch
AVP #2 Name: Accounting-DiskUsage-AVP Data: 123456
AVP #3 Name: Accounting-HostName-AVP Data: ARVO.IFI.UNIZH.CH
AVP #4 Name: Accounting-NodeCount-AVP Data: 2
Waiting till records are processed
Data successfully sent
Records overtaken from AAA_AcctModuleClientAcctRecCollector...
Number of records received for transmission: 5
```

```
AVP #0 Name: Accounting-Application-Id-AVP Data: 100
AVP #1 Name: User-Name Data: abachmann@access.unizh.ch
AVP #2 Name: Accounting-DiskUsage-AVP Data: 123456
AVP #3 Name: Accounting-HostName-AVP Data: ARVO.IFI.UNIZH.CH
AVP #4 Name: Accounting-NodeCount-AVP Data: 3
Waiting till records are processed
Data successfully sent
Records overtaken from AAA_AcctModuleClientAcctRecCollector...
Number of records received for transmission: 5
AVP #0 Name: Accounting-Application-Id-AVP Data: 100
AVP #1 Name: User-Name Data: abachmann@access.unizh.ch
AVP #2 Name: Accounting-DiskUsage-AVP Data: 123456
AVP #3 Name: Accounting-HostName-AVP Data: ARVO.IFI.UNIZH.CH
AVP #4 Name: Accounting-NodeCount-AVP Data: 4
Waiting till records are processed
Data successfully sent
Destroy AAA_AccountingClientModule...
Segmentation fault
```

Anhang F: CD-Rom