Design Document

Offline Charging Server (Offline CS)

Version 1.0

Objective

Document Scope

The information provided in this document specifies the design details of Operations of Offline Charging Server (Offline CS). For complete scope of Offline CS please see the Project Proposal.

Table of Contents

1. References & Abbreviations

References

Following is the reference document list, which is related to the information present in this document:

[1] 3GPP TS 23.299 V8.3.0: “Telecommunication management; Charging Management; Diameter charging application”.

[2] IETF RFC 3588: “Diameter Base Protocol”.

[3] 3GPP TS 23.240 V8.3.0:“Telecommunication management; Charging Management; Charging Architecture and Principles”.

Abbreviations

Following are the abbreviations that have been used in the document:

IMS	IP-Multimedia Subsystem
Offline CS	Offline Charging Server
API	Application Programming interface
AMPS	Asynchronous Middleware for Protocol Serve

2. Introduction

Offline charging is a process where charging information for network resource usage is collected concurrently with that resource usage. Offline Charging Server is providing this service to the IMS network entities. These entities include P_CSCF, I_CSCF, MRFC and S_CSCF. On Offline CS these charging information are passed through a chain of logical charging functions for further processing and functionality. At the end of this process, CDR files are generated by the network, which are then transferred to the network operator's Billing Domain for the purpose of subscriber billing and/or inter-operator accounting (or additional functions, e.g. statistics, at the operator’s discretion). The BD typically comprises post-processing systems such as the operator's billing system or billing mediation device.

In conclusion, offline charging is a mechanism where charging information does not affect, in real-time, the service rendered.

Offline CS Architecture

The Offline Charging Server (Offline CS) is written using the AMPS framework, which is an Asynchronous Event Based framework that provides two major facilities to us as application writers:

- Hides the implementation details of Operating System specific API, thus making any code written on top of AMPS to be portable across various operating systems.
- Provides extensive support for asynchronous and event driven programming paradigm specifically tailored for writing protocol servers for networking and telecommunication domains.

Both of the facilities above make AMPS a prime candidate to be used in the development of Offline Charging Server, hence the choice to use AMPS in the Offline CS design and development.

AMPS framework is implemented as a shared library, and lies at the bottom of the architecture . The diameter base protocol stack has been implemented as a shared library, and lies on top of AMPS. The communication between Diameter Base Protocol stack and AMPS is through direct function calls and through callback functions.

Offline CS itself is implemented as an AMPS module. The Rf interface which is part of Offline CS functionality, is implemented as a standalone AMPS module. For functionality requiring access to DBMS, a standalone AMPS module is implemented, known as Database Module. Any access to DBMS required by any constituent component of Offline CS will be serviced through the Database module and never through direct DBMS interaction. Please refer to figure 1 for a visual display of the broad term Offline CS Architecture.

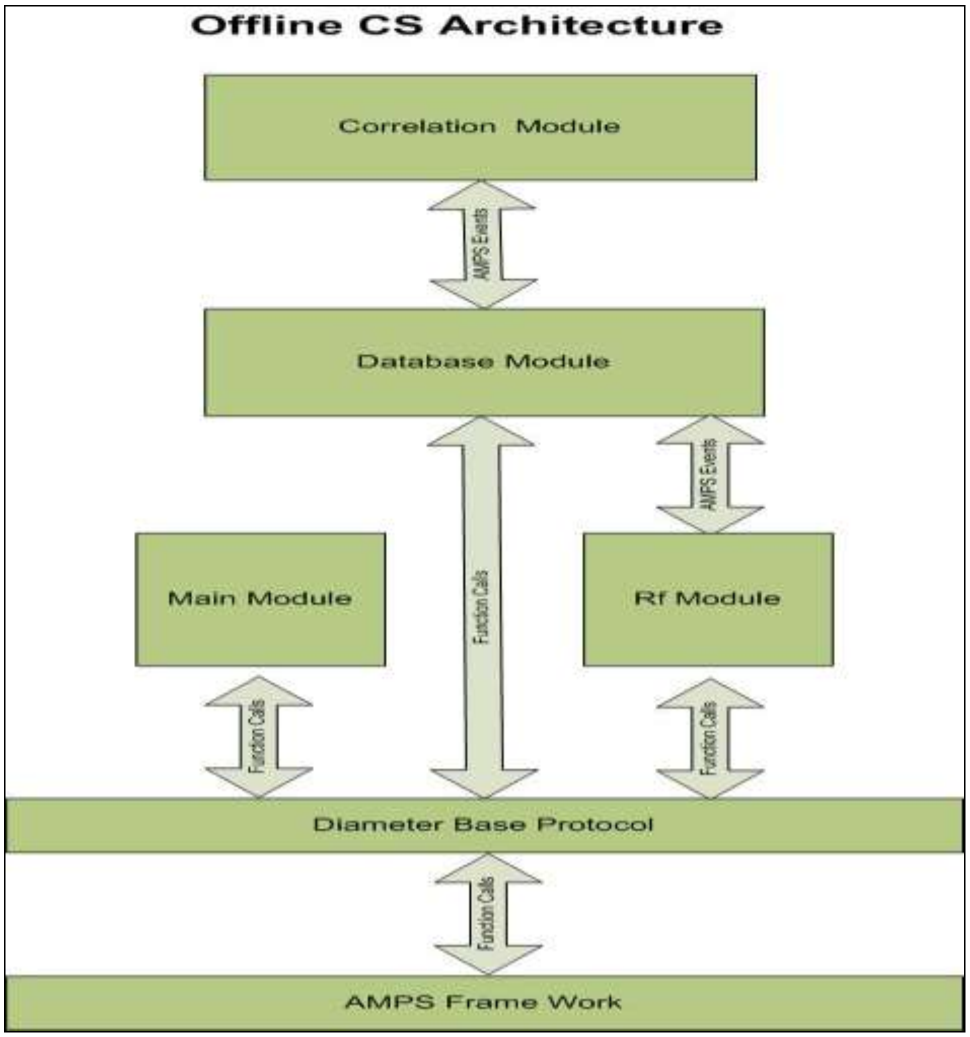

Figure 1. Offline CS Architecture

4. Components of Offline CS Architecture

Each component of the Offline CS architecture, as illustrated above, is described in detail in the following section.

4.1 Asynchronous Middleware for Protocol Sever (AMPS)

Asynchronous Middleware for Protocol Servers (AMPS) is an open source server development platform supporting event driven, asynchronous programming paradigm. AMPS is ideally suited for development of high performance application protocol servers in telecommunications and networking domains.

AMPS, with its asynchronous, event-driven programming model and fully abstracted OS dependent layer provides the end-user with following features:

- High performance.
- Supports multi – Operating system environment.
- Requires substantially lesser time to develop and market

AMPS is a powerful middleware to implement protocol servers. Due to above mentioned features AMPS was selected as the underlying middleware framework to be used in the design and development of Offline CS. AMPS exists in the form of a shared library to be linked with the main application that is to be developed.

4.2 Diameter Base Protocol Stack

The Diameter Base Protocol Stack is a complete implementation of IETF RFC 3588 specifications, utilizing the AMPS framework. This exists in the form of a shared library that needs to be loaded and configured by its clients. The Offline CS will function as a client of the Diameter Base Protocol Stack. The interaction of Diameter Stack with other components of Offline CS is further explained below item wise:

4.2.1 Interaction of Diameter Stack with AMPS

The Diameter Stack is a client of AMPS, and makes direct function call into the shared library of AMPS. Various functionalities i.e. notification of sockets data reception, Diameter Stack registers its callback functions with AMPS. These callback functions are in turn called by AMPS when relevant events or triggers are fired. So, the interaction of Diameter Stack and AMPS is that of mutual direct function calls into each others' shared libraries.

4.2.2 Interaction of Diameter Stack with Main Module

The main module of Offline CS is an AMPS application. It is client of Diameter Stack. The main module is responsible for the configuration of Diameter Stack shared library. This configuration happens through direct function calls into the Diameter Stack Shared library, using special configuration API provided by the Diameter Stack.

4.2.3 Interaction of Diameter Stack with RF Module

Rf Module is a client of Diameter Stack. Rf module makes direct function calls into the Diameter Stack shared library. The Diameter Stack also calls one function from this module. During its initialization, RF module registers a Message Processor function with the Diameter Stack. During registration, they provide their Application ID as well as the function pointer for the message processor callback function to the Diameter Stack. The Diameter Stack maintains an association between the Application ID and the Message Processor function associated with the application ID. Whenever a Diameter Message arrives from the network, which is to be processed locally and which is destined for an application ID for whom an Application has registered a message processor callback function – then the Diameter Stack fires the relevant callback function giving it the newly arrived Diameter Message, in the parameter. This way, the Diameter Stack also makes direct function calls into the Rf module as well.

4.3 Main Module

The Main module is an AMPS module inside which the Offline Charging Server is passed control by the operating system loader. The Offline CS hence begins life inside the main module. The main module is responsible for the following things:

- Loading its own configuration file and implementing the configuration dictated by the configuration file.
- Initializing and configuring the AMPS framework.
- Initializing and configuring the Diameter Base Protocol Stack.
- Initializing and loading each AMPS module required by Offline CS, i.e. RF module, and the database module.
- Passing control over to AMPS framework for the event based logic to take over control and drive subsequent code execution.

The main module therefore makes direct function calls into AMPS and Diameter Stack .

4.4 Rf Module

Rf Module is AMPS module that implements a 3GPP interface component for Offline CS.

As part of its initialization, it registers with the Diameter Stack their Application ID as well as the function pointer of a Message Processor function that they implement. Whenever a message for local processing arrives from the network that is destined for either of them, the Diameter stack calls the callback function of the relevant module. From that point onwards, the control has been passed to the Rf module message processor function. The message processor function then implements the required application logic to process the message that has been passed on to it.

In Rf module a configurable timer is supported to support the reception of ACR [interim] and ACR [STOP].Upon timer expiration it stops receiving further ACRs related to that session and deletes the session as well with error indication.

4.5 Database Module

Database module is an AMPS module that is responsible for providing DBMS related services to Rf Module. Whenever the Rf module require to interact with the DBMS, for getting or setting values in a certain table, they always do it through the Database module and never directly. In that sense, both the Rf module is a client of the Database module. The communication between the Rf module and the Database module is done through AMPS events. When the Rf module need to query the DBMS or set values into the DBMS, they send an AMPS event to the Database module with all the required information. The Database module receives the event, and gets the associated event data, on the basis of which it interacts with the DBMS on behalf of the Rf module. Similarly, the result of the requested Database operation is relayed back to the requesting Rf module through AMPS events.

4.6 Correlation Module

Offline CS collects the charging information from different network entities’ CTF i.e I-CSCF, P-CSCF, MRFC and S-CSCF. The Correlation module will be implementing to combines these charging information generated by CTF while they are belonging to the same bearer/session/service resource usage. The charging information has to be aggregate for the same charging session and correlate for the same service.

The correlation Module will be only interacting to DB Module through AMPS events .The interaction will be for the retrieval of charging information from different tables related to same session/bearer/service for final CDR generation.

5. Typical control and Data Flow in Offline CS

A Typical control and data flow sequence is given below for better architectural understanding:

1. Offline CS Main module reads its configuration XML file. It then loads and initializes AMPS, Diameter Base protocol shared library and required AMPS modules, i.e. Rf and Database module. It transfers control to the AMPS scheduler.

2. As part of its initialization, each diameter Application registers a callback function with the Base protocol library against its Application ID.

3. A raw diameter message is received by Diameter base protocol library. The library converts it into structured format. Then the library passes this structured message to the relevant Diameter Application by calling its callback function. The callback function is found by doing a lookup against the Application ID in diameter message, in a library maintained mapping data structure.

4. The Diameter application's callback function does some processing based on message contents and application defined logic. As part of the application defined logic, a query needs to be made to get/set some data to/from the DBMS. Since this is a blocking operation, an event is generated. The associated parameters) are passed as event data. SessionID is also passed for correlation . The diameter application has already registered for the event it expects the DBMS to generate as a result of its own event. The process Message callback function returns. The application callback also returns control.

5. The Database Module was registered for the type of event generated by the diameter application in step 4 above. Its event handler gets called. The event handler gets the parameters, queries the DBMS and gets back results. It formats the results in the desired standardized format. Then it generates an event and passes the result in addition to SessionID correlation info to the event as event parameter. The event handler returns control.

6. The Diameter Application had registered itself for the type of event generated by Database Module in step 5 . Its event handler gets called. Application does some processing based on the result of the database operation. Then let us say the application needs to send a diameter message to the originator of the message it received. It builds a diameter message by calling different functions (some inside the application and some from the base protocol library). Then it calls the base protocol library's send message function to send the message. The event handler returns control.

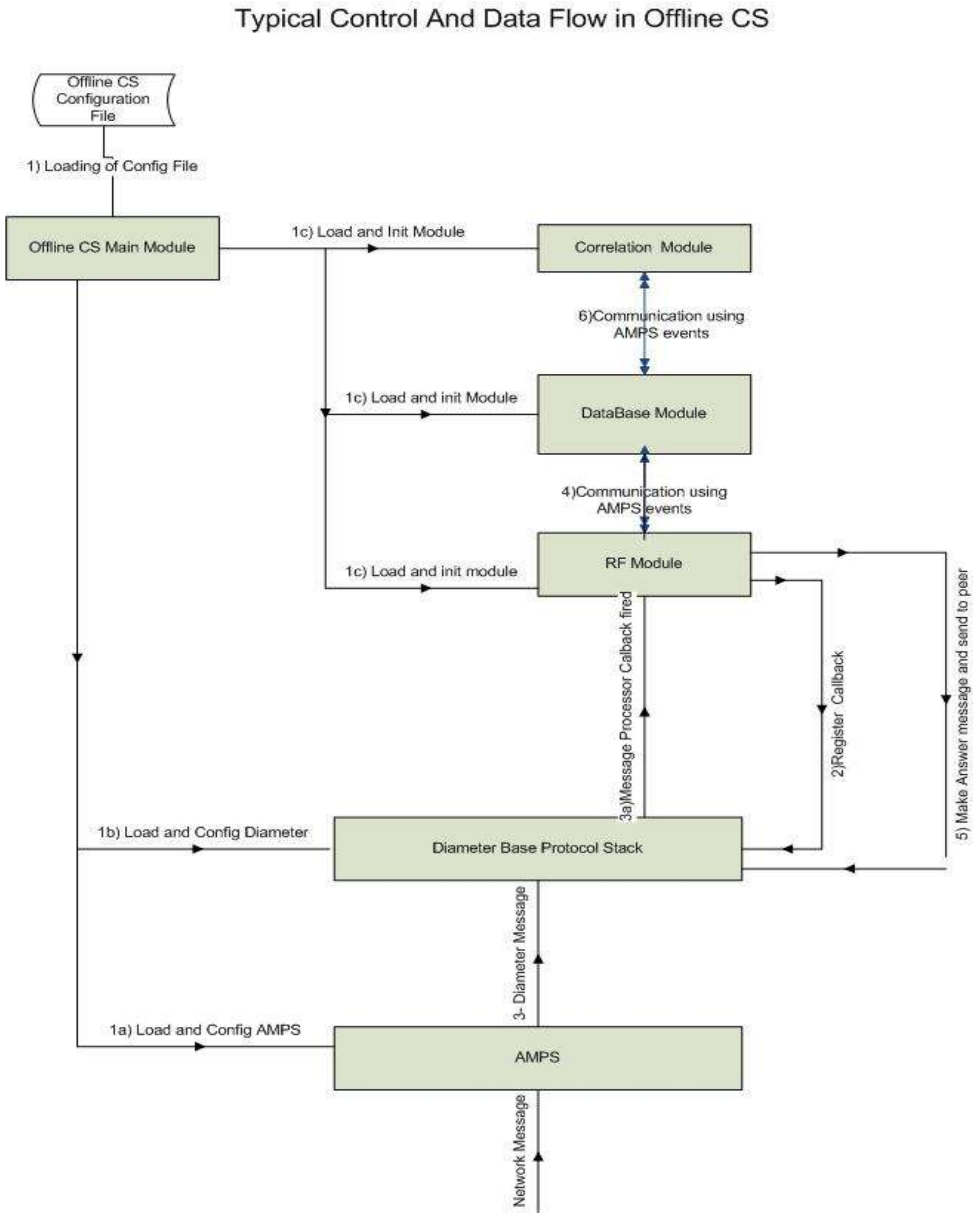
Typical Control And Data Flow in Offline CS
Offline CS Configuration File
1) Loading of Config File
Offline CS Main Module
1c) Load and Init Module
Correlation Module
6)Communication using AMPS events
1c) Load and init Module
DataBase Module
4)Communication using AMPS events
1c) Load and init module
RF Module
3a)Message Processor Calback fired
2)Register Callback
5) Make Answer message and send to peer
1b) Load and Config Diameter
Diameter Base Protocol Stack
3- Diameter Message
1a) Load and Config AMPS
AMPS
Network Message

Fig: Offline Charging System Flow Diagram