# Flexible Multi-Service Telecommunications Accounting System

E. de Leastar and J. McGibney

Telecommunications Software Systems Group, Waterford Institute of Technology, Ireland
e-mail: edeleastar@wit.ie

## Abstract

Due to market deregulation and technological advances, a multi-service broadband environment is emerging, generating a requirement for a new approach to pricing systems. This paper presents such an approach and describes a working implementation of an innovative and flexible multi-service accounting system. The system is based on a mature understanding of the nature of charging algorithms, and is implemented as a set of interoperating distributed components. At the heart of the system is a rating engine built around a service portfolio, tying together spreadsheet specification of arbitrarily complex services, charging algorithms and tariff tables. The very wide range of service delivery technologies in existence motivates the design of a system that is independent of proprietary formats and based on a dynamic service portfolio structure and a generic service detail record definition.

## Keywords

Accounting, Charging, Rating, Multiservice

## 1. Introduction

Until fairly recently, communications services have generally been straightforward - a limited number of services on offer, limited scope for customer tailoring of such services, and limited complexity. Now, however, there is considerable growth in the quantity and diversity of services. New communication technologies are enabling the introduction of variable-bandwidth with quality of service guarantees. ATM (ATM Forum, 1999) and Frame Relay already support a mature set of such services, and work is emerging from the IETF on QoS-guaranteed IP[1]. Some services will be provided as standardised network services based on core technologies, and others as higher-level user services that are in effect an amalgamation of network services, information content, and other components.

A new kind of accounting system is needed to provide metering and pricing support for these complex services. It should be capable of supporting the creation new services, charging algorithms and tariffs without the need for frequent upgrades. It should provide support for near real-time online billing, where the customer can very quickly obtain information on the cost of the service just used. And it should hide as much of the complexity associated with charging algorithms as possible, delivering to service providers a mechanism for manipulating these algorithms in a familiar environment.

Recent work on the specification of charging schemes for ATM and emerging quality IP services has been carried out under the European *ACTS* (Advanced Communications Technologies and Services) programme, and forms the basis of the work reported on here.

This paper proposes an architecture for a flexible broadband pricing system and reports on a working implementation that has been used in European trials. An overview of accounting concepts is presented, identifying the major subsystems within broadband accounting. This is followed by analytical discussion of the structure of the charging algorithms. A Service Portfolio is introduced encapsulating the proposed charging approach, and the outline of an implementation is presented, in CORBA IDL.

### 1.1. Accounting Concepts and Subsystems

The overall accounting process can be divided into *metering*, *rating* (or pricing), and *billing* systems, as shown in Figure 1. The role of the metering system is to measure, collect and forward information that identifies and describes the usage of network services. This information can then be used for charging purposes. The metering system retrieves this information from specific measuring points, distributed throughout the network. The measuring points are dependent on the network. For ATM these are switches, for IP these are routers and so on. This service usage information is organised in the form of *Service Detail Records* (SDRs). The SDR is a generalization of the Call Detail Record (CDR) provided by most conventional telephony switches.

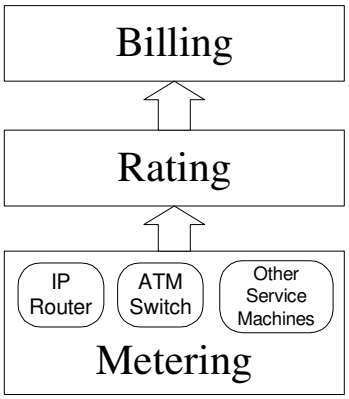


**Figure 1 Accounting Systems**

The main role of rating, the focus of this paper, is to provide support for the billing system so that it can invoice customers for telecommunications services. The rating system *computes the charges* from which the billing system composes the customer bills.

## 2. Multi-service Charging Algorithms

Charging in a multi-service environment is complex. Many different services can be delivered on the same equipment, and a wide variety of charging schemes and tariffs are possible. The charging approach proposed here stipulates that all the necessary and sufficient information for charging be encapsulated in four entities – a charging algorithm, a tariff table, an SDR type specification, and set of service determination rules. Particular reference is made to an implementation of charging for ATM and Quality IP services. However, the architecture is sufficiently flexible for wider applicability to as yet unforeseen technologies.

## 2.1. Charging Algorithms

The central task of rating is to evaluate the charge for a given service usage so that the user can be billed appropriately. This involves the application of a charging regime to SDR data to yield a monetary charge. Charging for a service can be considered as containing elements relating to *subscription* and usage *sessions* (CANCAN, 1997). Of most interest to us is the session charge. This is the variable component of charging that is normally highly dependent on the contents of the relevant SDR(s). Generally, the session charge can be expressed as:

$$C = a_1X_1 + a_2X_2 + \ldots + a_nX_n, \qquad \textit{(Equation 1)}$$

where $X_1, \ldots, X_n$ are charging parameters or functions of charging parameters that do not depend on numeric tariffs,
$a_1, \ldots, a_n$ are charging scheme numeric coefficients, known as tariffs, and,
$C$ is the charge for the session.

Thus $X_1, \ldots, X_n$ represent the commodities that are being charged - e.g., duration, packet counts, bit rates, etc., and it must be possible to express them in terms of SDR field parameters. The tariff coefficients $a_1, \ldots, a_n$ are then the price factors that are applied to the chargeable commodities. For example, a proposed charging scheme (SUSIE, 1999) for MPLS real time service looks like this:

$$C = a_1*P*T, \qquad \textit{(Equation 2)}$$

where $T$ is the duration of the connection, $P$ is the peak data rate (i.e. the guaranteed bandwidth) and $a_1$ and $C$ are as above

Table 1 gives an analysis of this charging scheme and how its terms derive from SDR fields

| Parameter | Parameter Type | SDR field(s) | Units |
|---|---|---|---|
| $C$ | | | [$] |
| $a_1$ | Tariff | | [$/kbit] |
| $P$ | Charging | *ContractedPeakDataRate* | [kbits/sec] |
| $T$ | Charging | *(CollectionTime + CollectionTimeOffset) - (CallStartTime + CallStartTimeOffset) – DisruptionDuration* | [sec] |

**Table 1: Example charging scheme expressed in terms of SDR Parameters**

## 2.2. Tariff Tables

Tariffs are the coefficients of a charging scheme; they are the price component of each term. Consider again the charging schemes above (equation 2). Here $P$ and $T$ are charging parameters as they relate directly to values that can be retrieved from the SDR. $a_1$ is viewed as the tariff as it is effectively the price that can be altered as the market demands. This tariff is not necessarily a fixed value however but may depend, for example, on time of day or geographical location of called party. Such variable tariffs are traditionally organised in tariff tables, examples of which are the rate cards available from most telephone companies and published in telephone directories.

### 2.3. Service Determination Rules

The third, and perhaps most significant, key to charging in a multi-service environment is to have a means of specifying service determination rules. The purpose of such rules is to allow us to take an SDR and unambiguously identify the service that was used to produce this SDR. Then the correct charges and tariffs can be applied. For maximum flexibility it is allowed to use any combination of SDR parameters to identify the service. An example service determination rule is as follows:

| | |
|---|---|
| Service: | "ConstantBitRate, local, night, no quality guarantees" |
| Condition: | if ServiceCategory is "CBR"<br>and CallingNumber and CalledNumber have the same first 3 digits<br>and CallCreationTime is between 00:00 and 08:00<br>and CollectionTime is between 00:00 and 08:00 on the same day |

It is important that the service identification rules unambiguously identify a single service from an SDR. It would be too much of a burden on the user to expect all service rules entered to be mutually exclusive, and to cover the entire “SDR space”, ensuring that any SDR causes one and only one identification rule to be satisfied. Thus the user is allowed to assign a priority level to each service rule, and rules of higher priority are tested first.

### 2.4. Representing the Schemes

The challenge for the implementation of the accounting system to support these schemes is to define a representation that is flexible and user-friendly, but sufficiently powerful to permit the most complex of charging regimes to be captured. Specifically, the representation format must:

- be easy to manage.
- permit arbitrarily complex formulae.
- allow multiple SDR types to be manipulated
- be consistent with the world of accounting and billing.

Flexible representation of charging algorithms, satisfying the above requirements, is achieved by providing the user with a spreadsheet-like user interface. Usage data is specified for selection *a priori* to be placed into cells in the worksheet, and other cells can be used by the user to enter formulae or conditions. Sample usage data can be entered for testing purposes, but actual usage data is taken from the SDRs and placed into the specified positions at runtime, and the calculations are performed automatically.

## 3. Service Portfolio

A service provider will typically offer a range of services, some at the bearer communications level and others that are termed "value-added". The term service portfolio is used to represent this service offering. A present day public fixed line network provider might have the following service portfolio:

- PSTN residential
- PSTN business
- Virtual Private Network (VPN)
- Freephone
- Call Answering
- Telephone equipment rental
- Directory assistance
- ISDN

Some of these services are entirely different from one another (e.g. Directory Assistance vs. ISDN) while others have little technical difference and only differ in how they are charged (PSTN business vs. PSTN residential).

For the purposes of accounting, the service portfolio can be defined as the collection of service information that is necessary to support the rating process. Formally, we define service portfolio as a set of *service templates* that are offered by a service provider, where each template contains the following:

- SDR type - format of usage data from the service machine
- Service Determination Rule - logic to identify the service
- Tariff Table - algorithm to generate coefficients of the charging scheme; i.e, the *prices*
- Charging Scheme - algorithm to generate charge for billing

There is no restriction on reuse of any of these elements across multiple service templates. For example, the same charging algorithm could be applied to different services, but the tariffs tables could be different.

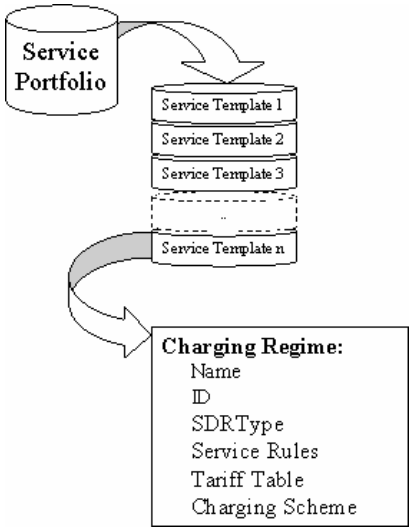


**Figure 2: Service Portfolio**

The Service Portfolio is modelled as a set of CORBA IDL types. This facilitates the centralised management of the service portfolio, with remote access across a CORBA infrastructure. Client applications include Graphical User Interface (GUI) applications to create, edit, modify and delete service portfolio entries (the *service templates*), generalised report generators, and of course the rating process itself, which will need to match service templates against incoming SDRs and compute charges accordingly.

The Service Portfolio IDL is divided into three categories, defined as separate modules. The first category, *SDRTypes*, specifies a comprehensive generalised representation for usage data gathered by the metering system. The second category, *Schemes*, defines the structure of the algorithms maintained within the service portfolio, specifically the service rules, tariff tables and charging algorithms. The third category, *ServiceTemplate*, models individual entries within the service portfolio in terms of the *SDRTypes* and *Schemes* modules.

### 3.1. SDR Types

The structure of the SDR is at the root of the accounting model, as SDRs are ultimately the origin of the charges generated by the system. The model developed in IDL is completely generic, catering for current and future metering technology and service types. Struct *t_ChargeParameter* defines a meta-type, describing the name, type, default value, and unit denomination of one element within an SDR.

```
struct t_ChargeParameter {
  string   name;
  any      defaultValue;
  string   units;
};
typedef sequence<t_ChargeParameter> t_SDRType;
typedef string                      t_SDRId;
```

**Figure 3: SDRType Structure**

The default value and type are represented as a single *any* value, which by definition has a *typecode* field. For instance a charge parameter for an MPLS-SDR might consist of:

*"ContractedPeakCellCount", 0 (long long), "kbits"*

An SDRType is simply a collection of these structures; i.e. a sequence of *t_ChargePatameter* objects. SDRType is thus regarded as a meta-type, describing a category of SDRs. This allows us to define an MPLS-SDR for MPLS traffic, or an IP-SDR for IP based transport, or we may define an SDR for content delivered. Defining different SDRType representations for each of these types of service technology is critically important to the composition of charging algorithms. Essentially the SDRType for a particular category of service defines a vocabulary in which the charging algorithms can be expressed. This vocabulary permits the composition of these algorithms in an intuitive manner; with charging formulae manipulating named charging parameters derived from the SDRType associated with the service.

The actual usage data itself, as measured during the delivery of a service, is represented as a collection of *t_MeterParameter* structs. Each meter parameter consists of an identity, locating a descriptive *t_ChargeParameter* in a corresponding SDRType sequence, and the actual data itself, encoded as a CORBA *any*.

```
struct t_MeterParameter {
  octet identity; //attribute identifier
  any   value;    //Value
};
typedef sequence<t_MeterParameter> t_UsageData;

struct t_SDR {
  t_SDRId             type; // Type + version information
  t_UsageData         data;
};
typedef sequence<t_SDR> t_SDRList;
```

**Figure 4: SDRs**

Thus for an MPLS-SDR, a meter parameter might simply be:

*12333314, 5*

The first number is some parameter gathered from a service machine, and the second

associates this measurement with some *t_ChargeParameter* meta-type within an SDRType sequence (say *ContractedPeakDataRate* from the example above). A *t_SDR* is defined as a sequence of these charge parameters, along with an id of the *t_SDRType* of which this SDR is an instance.

### 3.2. Schemes

Each of the three algorithms that compose a service template: the service determination rules, the tariff table and the charging algorithm, are modelled in IDL as a struct *t_Scheme*. Using structs, as opposed to interfaces, is a strategic decision, as client applications will be capable of preloading the full service portfolio without having to continually invoke remote functions to access scheme attributes (Mowbray et al, 1997). This reduction in remote invocations is an important consideration for GUI applications manipulating the portfolio, but absolutely crucial for the rating process, which will be unable to tolerate network latency implied by remote invocations during charge computation.

```
typedef unsigned long   t_SchemeId;
typedef string          t_Name;
typedef sequence<octet> t_SchemeFormulae;
struct t_Scheme
{
  t_SchemeId       ID;
  t_Name           Name;
  t_SDRId          SDRtype;
  t_SchemeFormulae Algorithm;
};
typedef sequence<t_Scheme> t_SchemeList;
```

**Figure 5: Scheme IDL**

Each Scheme consists of an ID, a Name, an SDRType and an Algorithm. The SDRType identifier maps to an SDRType structure defined above, defining the vocabulary of the algorithm.

Modelling the algorithm itself in IDL poses a dilemma. Spreadsheet technology has been introduced as the appropriate mechanism for defining and manipulating charging algorithms. In this scenario, the SDRType entities are mapped to specific cells within a spreadsheet, and a set of interrelated algorithm formulae can be assembled using these cell references to compose a “result”, or set of results for the algorithm. A spreadsheet is a complex structure, with multiple internal dependencies between cell contents, and sophisticated formulae of arbitrary length and structure. Modelling this in IDL would be counter-productive, pushing a degree of complexity into a domain fundamentally unsuitable for it.

The dilemma was resolved by completely abstracting the entire algorithm within the *t_Scheme* structure as a sequence of octets – essentially a binary stream. Within the database management system hosting the Service Portfolio, the stream is stored as a Binary Large Object (BLOB) and placed in a relational table. Within the concrete classes that manipulate schemes, this binary stream is serialized into its native format. For the GUI applications this native representation is attached to a visual component rendering it as a traditional spreadsheet. During the rating process, they are similarly serialised into native worksheet format, but no GUI is attached. Here the calculating engine is employed to recalculate all cells, and generate results employed in the rating process.

### 3.3. Service Templates

The Service Template can now be defined as a structure encapsulating naming information + IDs of the Charging Scheme, Tariff Table and Service Rules objects as defined above.

```
typedef unsigned long   t_ServiceTemplateId;
typedef string          t_Brand;
typedef string          t_Description;
struct t_ServiceTemplate {
  t_ServiceTemplateId ID;
  t_Name              Name;
  t_Brand             Brand;
  t_Description       Description;
  t_SchemeId          ChargingScheme;
  t_SchemeId          TariffScheme;
  t_SchemeId          RulesScheme;
};
typedef sequence<t_ServiceTemplate> t_ServiceTemplateList;
```

**Figure 6: Serviceportfolio Servicetemplate IDL**

A Service Template embodies a *charging regime,* i.e. a complete set of algorithms which, when applied to some usage data gathered from a service machine (an SDR), will compute a charge for that service. The Service Portfolio is thus modelled as a sequence of *t_ServiceTemplate* objects – a collection of charging regimes that can be used to compute charges for all the offerings a service provider may choose to make available.

## 4. Rating

In order to realize the charging approach embodied in the Service Portfolio, a software component must be designed to compute the charges based on usage data gathered from network elements. This is termed a *rate engine*. This component must be supported by other components: database and GUI applications to store, retrieve and edit the service portfolio itself; agents to gather the usage information (SDRs) from the service machines; gateway components to route these SDRs to rate engine(s), and to route generated charges to database and/or interested client applications (web browsers for online accounting, for instance).

The components are completely decoupled, with minimal dependencies on shared IDL specified interfaces. This is achieved through widespread deployment of the OMG Event Service (Wang et al, 1999), facilitating the asynchronous transmission of the key data structures (SDRs and Charge Records). While the Event Service does not provide Quality-of-Service (QoS) guarantees, is serves as a stepping-stone to next generation event service implementations, specifically those taking advantage of the CORBA 2.3 (OMG, 1998) messaging and notification services (OMG, 1998). These have a range of QoS parameters, which can be tuned to meet the demands of the accounting system.

The rate engine is the most computationally intensive of these components, and its structure is the key to a flexible, performant accounting system. Rate engine components are instantiated within a multithreaded rating server, each engine listening to one or more event channel for SDRs, and pushing charge records to an output channel. The SDR channel is fed by metering agents, typically interfacing with Service Machines (switches & routers) through SNMP. Attached to the Charge Record channel are listeners pumping the records to a DBMS, or distributing them to online client applications.

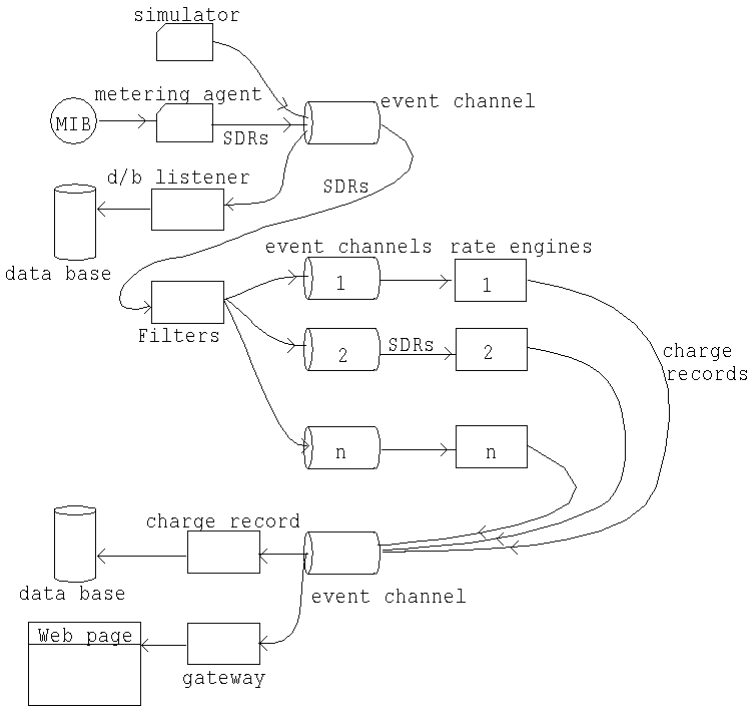


**Figure 7: Rating Components**

## 4.1. The Rate Engine

Within the engine, the actual charges themselves are computed. Essentially the engine, preloaded with the service portfolio using interfaces defined above, applies the appropriate charging regimes to the incoming SDRs, generating the charge records accordingly. This is carried out in five stages (Figure 8):

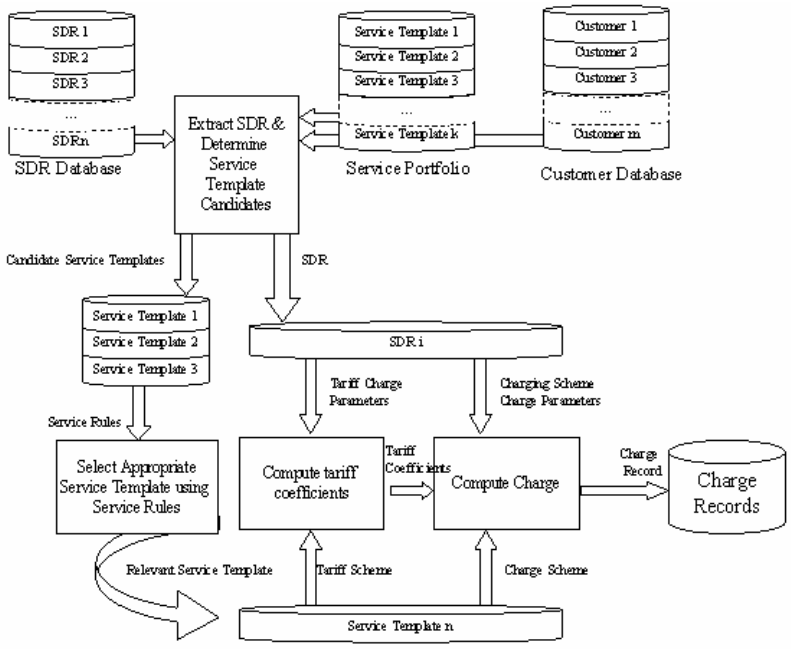


**Figure 8: Rating Process**

1. Identify candidate service templates from the service portfolio
2. Step through these service templates and apply the service determination rules until a rule is satisfied. This is the unique service template applicable to the SDR.
3. Using the tariff worksheet for this service template, compute tariff coefficients based on the received SDR field values
4. Compute the charge by applying the appropriate charging scheme worksheet
5. Write the resultant charge with other relevant information to a Charge Record.

Stages two, three and four involve replacing terms within formulae, and re-evaluating the formulae to yield results. It is analogous to a recalculation that takes place within a spread-sheet, with usage data placed on spreadsheet cells, and cell references within the formulae ensuring that this data percolates through the entire charging algorithm during recalculation.

### 4.2. Spreadsheet Component

Having determined not to publish the format of this algorithm/spreadsheet in the IDL, the issue of structure of the sheet itself can no longer be postponed - or perhaps it can? Spreadsheets are one of the most widely used application categories, and programmable spreadsheet software components are among the most common items on commercial component catalogues[2]. Classical component reuse technology can be deployed here to significantly cut the cost of development, and reduce the complexity of the implementation.

## 5. Conclusion

The accounting system described here has been realised as a set of interoperating components delivering a flexible multi-service accounting system. The system has been trialled primarily over ATM broadband networks, utilising in-house development testbeds, and that of the SUSIE project at Basel in Switzerland. The system has proved robust and performant, delivering a flexible and innovative "laboratory" for devising new types of charging algorithms appropriate for broadband services. Work is ongoing in deploying the system for Voice over IP services, implementing charging algorithms appropriate to this technology.

## 6. Acknowledgements

The authors wish to thank the *SUSIE, Bandwidth 2000* and *FlowThru* consortia for several helpful discussions. The work of these projects is part-funded by the European Commission.

## 7. References

ATM Forum, (1999), *Traffic Management Specification, Version 4.1.*
CANCAN Consortium, (1997), *Final Report on Static Charging Schemes and their Performance, Deliverable 9a.*
SUSIE consortium (1999) *Trials: Accounting* Results, Deliverable 6.
Mowbray,T, et,al. (1997), *CORBA Design Patterns*, Wiley.
Wang,R, et.al. (1999), "Event Bridges Across CORBA Event Service and Programming Language Event Models", *Journal of Object Oriented Programming*, Vol.12, No.4.
Object Management Group (1998), *The Common Object Request Broker Architecture, Revision 2.3.*
Object Management Group (1998), *CORBA Messaging*, Revised Submission.

[1] http://www.ietf.org
[2] http://www.componentsource.com