SUNRISE: An Architecture for Billing, Rating and Information Services in a Telephone Network

H. V. Jagadish* — AT&T Laboratories — jag@research.att.com

Rama Kanneganti — Lucent Bell Laboratories — rama@research.bell-labs.com

Inderpal Singh Mumick* — AT&T Laboratories — mumick@research.att.com

Abraham Silberschatz — Lucent Bell Laboratories — avi@research.bell-labs.com

Abstract

SUNRISE is a data architecture for the usage and billing information management in a telephone network. Its aim is to provide *real-time* rating and information services, in addition to on-line billing services. The SUNRISE system is based on the chronicle data model [JMS94] that provides for summary information to be maintained about the calling patterns of each customer. A logically integrated view is maintained of the customer profiles managed by different business units or service providers. The summary information and a portion of the customer profile information permanently resides in a main memory storage system (such as the Dali system [JLR+94] or the HP Smallbase system). A wide range of queries can be answered in real-time using the summary information and the customer profiles. Such queries can also be asked by a telephone switch during call set-up. More complex queries are supported by an on-line call detail database. Billing can be provided by programs that access the call detail database and/or the summary information. Bill summaries can be generated efficiently. The architecture is scalable, and new services can be added easily.

1 Introduction

Billing plays a major role in communication *services* provided over a network. Rating and billing information, if available in real-time, can enable new rating and information services, as well as other useful non-service functionality, such as fraud control. A flexible billing system that permits quick introduction of new services can result in a competitive advantage.

The SUNRISE system defines an architecture for rating telephone calls as soon as they are recorded, and for providing information services based upon real-time access to a summary of the rating information. For example, the query "How many dollars have I spent on International Calls

*This paper represents work performed in AT&T Bell Labs prior to the creation of Lucent Technologies. The current document represents a futuristic scenario developed at that time. Nothing in this document is intended to reflect the views of AT&T or any other corporation.

this month?" can be answered within a delay of about 1 millisecond. This delay is small enough that such queries can be made as part of call setup. For instance, such a query can be used to detect potential fraud in real time, and prevent it by call denial. The complete telephone call records are also kept on-line for detailed billing and complex querying.

There are two major hurdles in building such a system. First, a company's customer data is typically fragmented across multiple business units, with no cohesive notion of an "integrated customer profile". Second, the sheer volume of customer and telephone call data makes it difficult to store, rate, and query the data in real-time.

The SUNRISE system accumulates summary information as each individual call record is received and rated in real time. The nature of the summary information depends upon the services subscribed to by the customer. Queries can be posed against the accumulated summary information during call set-up, and such queries will be answered in real-time. Queries can also be asked by a billing system, or by an operator.

SUNRISE uses an "integrated customer profile" that is generated by software tools internal to the SUNRISE system from component service specific customer databases. The tools maintain efficiently the up-to-date value of the integrated customer profile on account of updates to service specific databases, using algorithms for "view maintenance." Thus, little change is required to the current fragmented and distributed ownership of the base data – we derive the required integrated view, and provide tools to manage inconsistency where it exists.

The processed call records are stored in a call detail database, and include the rate assigned to each call in real time. Billing is done by billing programs that have access to the call detail database and to the summary information kept with the integrated customer profile. It is thus possible to generate bills with only the summary information database, with only the call detail database, or with both. It is possible to do billing at any frequency desired by the customer. The call detail database contains processed call records and supports arbitrary queries.

The SUNRISE architecture is scalable, and can grow with increases in calling volume, number of customers, and number of services. A graphical service creation tool facilitates creation of new services and detection of conflicts between services. New services can be defined dynamically, even as the system remains up and continues running.

In summary, SUNRISE is a system that

- Enables *information*, *rating*, and *billing* services, including those that require
  - up-to-date rating and customer information in real-time, and
  - data from multiple sources;
- Supports the high throughput and the low response time needed by the high volumes of telephone calls in a major telephone network; and
- Facilitates the rapid introduction of new services.

The remainder of this paper is organized as follows. The Sunrise architecture for processing call records and managing customer profiles is presented in Section 2. The system features, and the key enabling technologies are discussed. We rate calls using a customized rating function that takes into account the list of features subscribed to by the billed customer. Our notions of customized rating and billing are discussed in Section 3. Section 4 explains the derivation and maintenance of an integrated customer profile, and the associated summary fields. The real-time rating server is explained in Section 5. A system configuration to meet the billing needs of a telephone service depends upon the size and volume of the service. We illustrate how a system configuration may be derived using an example in Section 6. Section 7 discusses the service creation facilities and mechanism in SUNRISE for creation of new rating, information, and billing services. Section 8 discusses the new options and challenges for the actual bill generation. A few example services, and their specification, are presented as examples in Section 9. We conclude with several examples of rating and information services enabled by SUNRISE (Appendix A).

2 The Architecture

The SUNRISE architecture is outlined in Figure 1. The three major components in the architecture, which are marked in bold font, are the real-time rating and information (RTRI) server, the call detail database, and the integrated customer profile.

The RTRI server processes the incoming stream of call records, and for each call record, (1) rates the call record according to a *customized* rating function that depends on the list of features and services subscribed to by the customer, (2) computes summary fields associated with the customer, and (3) generates and compresses a processed call record that includes the rating information. Additionally, the RTRI server responds to queries about customer profile and summary data.

The processed call records are stored in a call detail database. Billing programs can access summary information from the RTRI, and can access the detailed call records from the call detail database. We can support the (current) mode of operation where each service provider provides a separate billing program that generates separate physical bills for its customers, or we can support an integrated billing program that generates one bill per customer.

The integrated customer profile represents a logical integration of service specific customer profiles maintained by distinct service providers. In addition, summary fields can be defined for each customer and stored alongside the integrated customer profile. Summary fields summarize data from all the call records billed to the customer. Summary fields are defined by service providers for the customers who subscribe to individual services.

2.1 Features

We now consider the key features of the SUNRISE system.

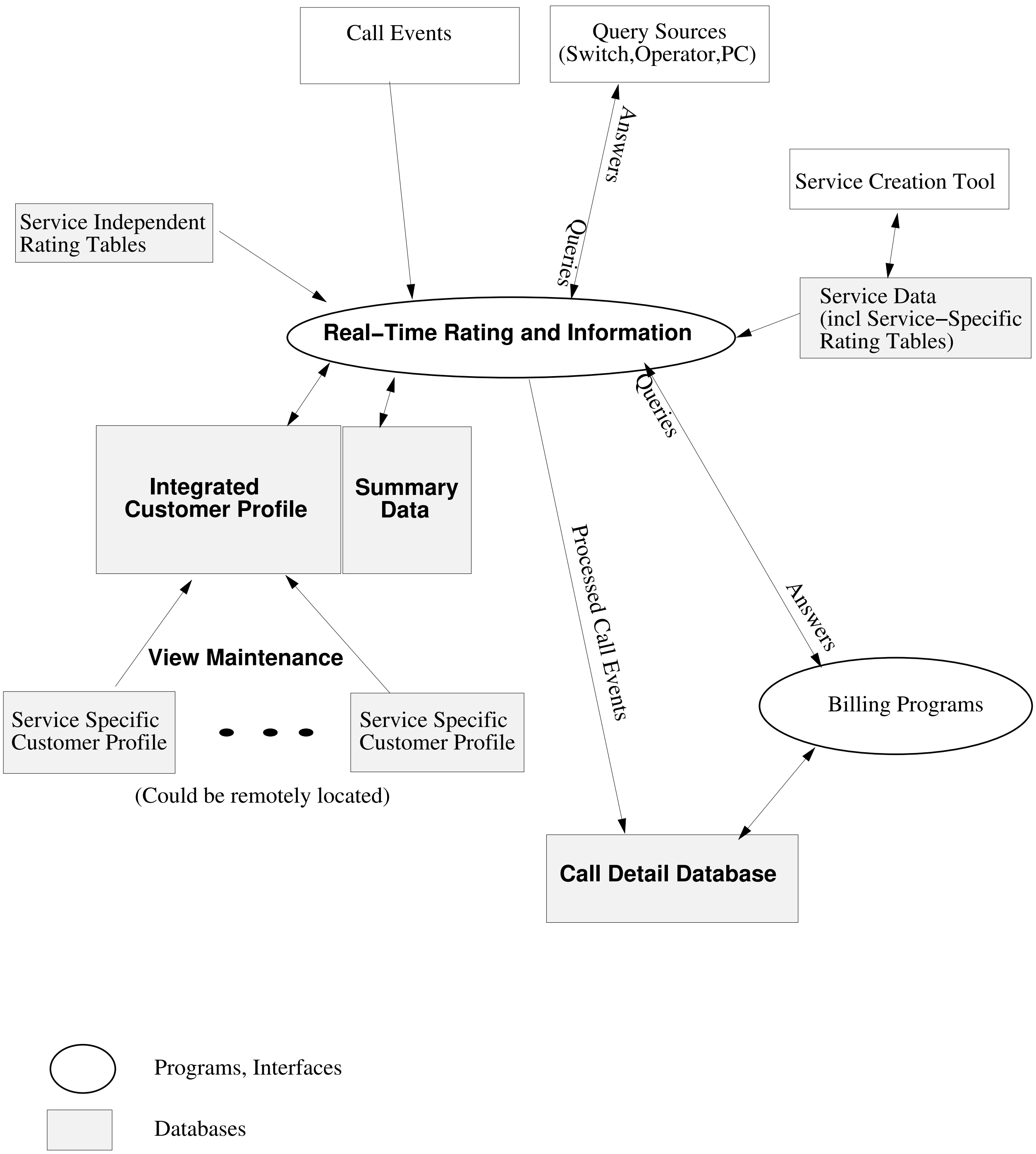

Figure 1: The SUNRISE Architecture.

- A large number of rating and billing services can be supported. Several examples are discussed in Section 9 and Appendix A.
- Triggers can be activated upon observing *interesting* sequences of call records (e.g., Fraud Detection).
- Real-time querying is possible, even during call set-up. Queries can be answered in about 1 millisecond.
- An integrated view of the customer profile is maintained. We permit existing service providers to retain control over their own data. New service providers may choose to build their own database, or to keep their data in our system. In either case, the data needed for offering new services and billing is accessible as a logical database.
- Several billing options (with and without call detail) are available. Billing without call detail can be done at any frequency, with low overhead. It is practical to generate daily bills, for example, and use this to debit an account automatically.
- New services can be created rapidly and efficiently.
- At the time of service creation, assistance is available in determining service interactions.
- The architecture is scalable with respect to increases in number of telephone calls, increases in number of services offered, and increases in number of subscribers.
- It is possible to introduce SUNRISE into an existing telephone network gradually.

2.2 Key Technologies

The proposed SUNRISE architecture is enabled by three key technologies.

- Stream-oriented processing, with real-time computation of useful information summaries. Each customer is allocated storage space that depends upon the number of services subscribed, and is independent of the number of call records billed to the customer. The summary fields are defined for each service at time of service creation, and can be periodically reset (e.g., to initiate new billing cycles). No storage space is allocated for services that a customer does not subscribe to.
- Incremental view maintenance. The integrated customer profile is derived as a view from multiple service specific customer profiles maintained by each service provider. As the underlying service specific data changes, the consequent changes to the integrated customer profile are computed incrementally, without recomputing the entire integrated customer profile.

- Main-memory database. The integrated customer profile and summary information is stored in a main-memory database. Main-memory databases can give transaction rates above 1,000 updates/sec, and above 10,000 simple queries/sec. Consequently, we can support the high throughput rates needed to process call records in real-time, and the low response rates needed to query the customer profile and summary information during call set-up time.

3 Customized Rating and Billing

In SUNRISE we distinguish between *rating* and *billing*. Rating is the rate given to a call when the call is placed, with consideration for subscribed `rating` plans. Note that the call rating thus depends upon (1) Service independent rating tables that take into account the time of day, the day of week, distance, and length of call, and (2) List of services subscribed to by the billed party, and (3) Service dependent rating tables that determine how each service affects the rates for a call. Note that our definition of the term "rating" is broader than the traditional definition, which considers only item (1) above.

Billing is the process of *aggregating over the rate charged for each call* made in some period, usually a month, to determine the final amount to charge. The rate charged for a call depends upon the services subscribed to by the billed person. In addition discounts and fees can be applied based upon the services subscribed to by the billed person, and upon the calling pattern.

In SUNRISE, rating is performed in real-time in the RTRI server. Billing is performed periodically for each customer, usually by looking at the call detail database, though any needed summary information can be pre-computed by the RTRI and looked up at billing time.

Ideally, the exact cost of a call should be computable as the call record is generated. However, there are some services that cannot compute the final charge for calls as the call records are generated. For instance, consider a volume discount plan where a 10% discount is applied if more than 15 hours of calls were made in one month. If the 10% discount is given on all *subsequent* calls after the first 15 hours of calls, then the discount can be computed during the rating process. However, if the 10% discount is to be applied to *all* calls, including the calls made within the first 15 hours, then the processed call records already generated cannot reflect this discount in their rating. (However, subsequent processed call records can reflect this discount, if desired.) The rate given to the first 15 hours of calls at the time the calls were placed has to be adjusted by applying a discount at billing time. However, summary information about the total charge for the calls can be adjusted at the time it is noticed that the total usage has gone beyond 15 hours, by taking a 10% on the charge accumulated so far. Thus, even with this plan, "correct" summary information regarding total charge is accessible in real time.

It is important that the summary information available in real-time be consistent with the information actually present in the bill, particularly the total charges. SUNRISE itself cannot guarantee this — depends on what a billing program does — but it does facilitate the maintenance of consistency, through providing a single logical store for all processed call records.

4 Integrated Customer Profile and Summary Fields

A logically *integrated customer profile* (ICP) is central to the SUNRISE architecture. The ICP includes a logical record for every customer of the telephone service. Customers include those that are individually billed, and those that are specified additional accounts (such as sub-accounts for a business, or the neighborhood account for a neighborhood billing plan). In other words, there is an ICP for every accounting unit to which calls are charged.

An indexed access to the integrated customer profile through telephone numbers and billing-ids is provided. Such access must be possible even when the customer is a large business and "owns" numerous telephone numbers, too many for all of them to be listed in the profile.

There is a notion of customer "type", specified by means of a type field in the customer profile. This type can be used to distinguish business customers from residential customers, "neighborhood" customers from actual customers, billed entities from sub-accounts, etc. Since the type information is simply a field in the profile, new types can freely be introduced at run-time, simply by overwriting this type information for the affected customers. If a single customer subscribes to multiple services, there is still a single type associated with the customer. Thus customer type is not equivalent to service type. This type field enables different kinds of processing to be performed for different customers depending on type. For example, the type field could have values: "residential", "small business", "medium business", "large business", or "virtual"[1].

4.1 Deriving the Integrated customer profile

We do not assume that a single integrated customer profile exists. Rather, we consider the case where a separate *service-specific customer profile* (SSCP) is maintained by each service provider within a telephone company. For instance, the providers of residential long distance, calling cards, international calling card, personal 800 service, business 800 service, and local service may all maintain their own specific customer profiles. The SUNRISE architecture allows each service provider to have control over its own data. We do provide for, and indeed it is our hope for new providers, that service providers can store their data within our system. However, it is unreasonable to expect that all current and future service providers will give up control over their data.

The integrated customer profile (ICP) is obtained as a view over SSCP's, as illustrated in Figure 2.

The view is nominally defined as a join of the SSCPs along customer-ids in a high level declarative language, such as SQL [ISO93]. Since the SSCP's may not use the same customer-id's (due to historical or independence assumptions), we need auxiliary mapping information to map the SSCP id's to the customer-id in the ICP.

[1] The virtual type applies to a non-billed accounting entity, such as "neighborhood" or "sub-account".

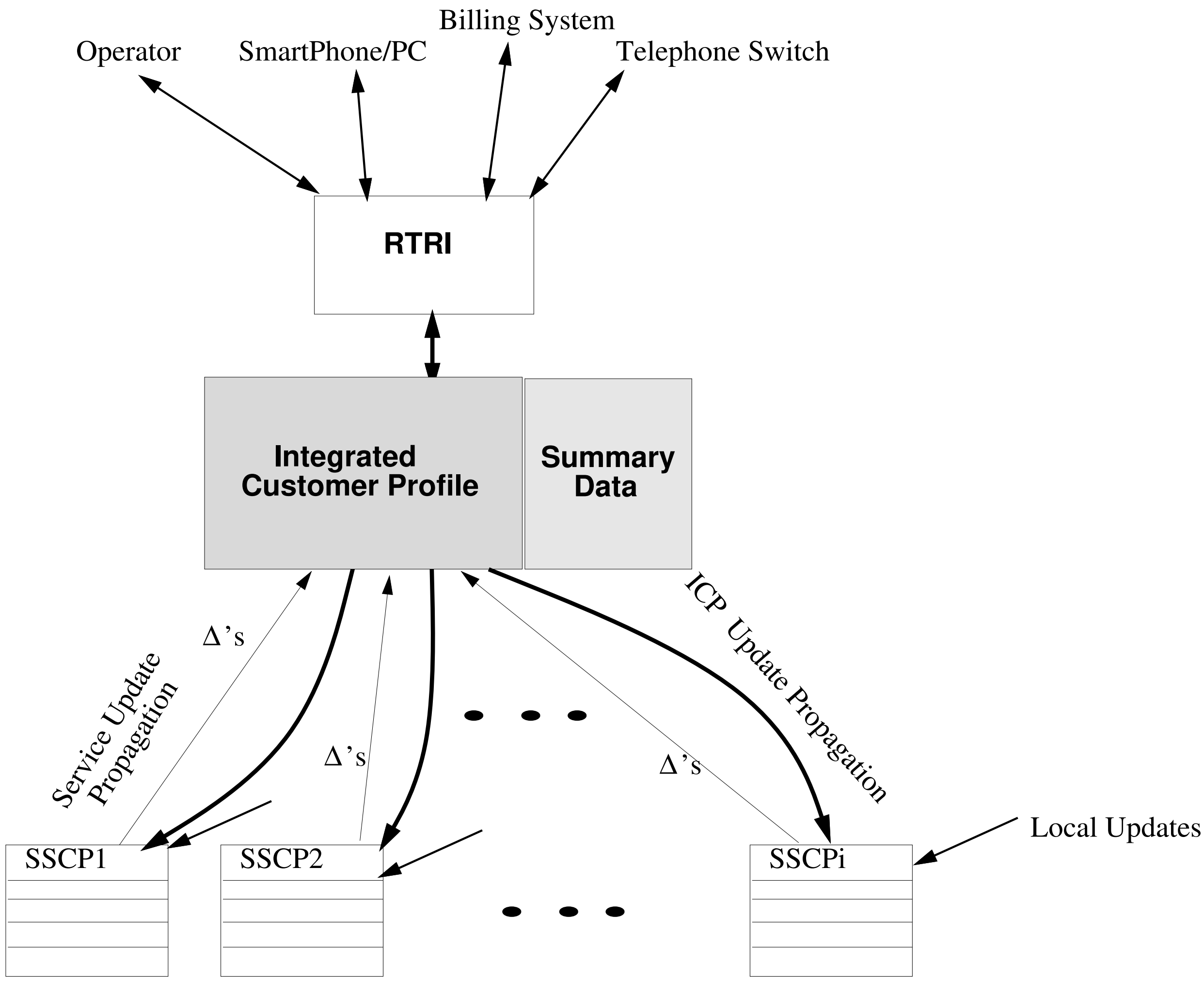

Figure 2: Customer Profile Database.

Often, the same information may be stored in slightly different forms in the different SSCPs, and sometimes, there could even be outright conflicts. The view definition for the ICP provides an explicit statement about how to construct the ICP from the underlying SSCPs. In particular, arbitrary functions can be applied to fields in SSCPs to "massage" them into the proper format for the ICP, and arbitrary conflict resolution strategies can be used where an ICP field is derivable from multiple SSCPs. While we do not provide any mechanisms ourselves to resolve these conflicts, we do provide a central place that forces recognition of such conflicts and requires the relevant system designers to create a resolution scheme. Where policy decisions are required to resolve conflicts, the need for them is highlighted.

Performance requirements dictate that queries applied to the ICP be evaluated with the information actually stored in the ICP, and without reference to the underlying SSCPs. To permit this, the ICP is maintained as a "materialized" view. That is, it replicates information contained in the SSCPs, which remain the primary repositories of most customer data.

SSCPs can be updated directly, by individual services. Such updates are called *local updates*. We require that knowledge of such updates be made available to the integrated profile, so that the effect of these updates may be propagated to the ICP. Ideally, an active database approach will be used to keep track of the changes, and to trigger re-computation of ICP on these changes. Incremental view maintenance techniques [GMS93] are then used to update the integrated profile, and to recompute the new fields. Incremental view maintenance is possible for views specified in the SWORD declarative language used in SUNRISE. If SSCPs are stored in traditional passive databases, as is likely to be the case for the immediate future, we require that a periodic log of updates be sent to the ICP from each SSCP. A "batched incremental" update of the ICP view is carried out, each time such a log is received. There will be a lag in the information showing up in the ICP, corresponding to the periodicity of the batched update. This lag will have to be made known to customers. For instance, if the period is one day, customers must be told that the new service they have subscribed to will be operational "by tomorrow" rather than immediately. Some customers may want to control the exact time at which a service is turned on – this can be done by performing the SSCP update enough in advance, and setting an "alarm" in the ICP to make the change at the appropriate time. Since the ICP is an active database, such alarms can be provided through triggers.

Restricted forms of updates can be supported directly on the ICP. These would then be translated into updates to the individual SSCP's.

The integrated customer profile includes certain computed attributes, defined by various services to make each customer record self contained for the service(s) being offered.

4.2 Summary Fields

The ICP is associated with summary data that captures, for each customer, a summary of his call records needed to support the rating and information services subscribed to by the customer. The summary data is potentially updated after each telephone call.

Examples of summary data are number of call minutes since the last bill, total cost of calls made on any particular day, and so on. We require that the amount of summary data be independent of the number of calls made or billed to a customer. This ensures that each customer has bounded space allocated to him in the summary part of the database. Bounding the space helps in ensuring that queries on a customer record can be answered in a few milliseconds, and in restricting the size of the database referenced by the RTRI server.

Customers may subscribe and unsubscribe to services requiring subscription. Their current subscription status for any service is recorded in the integrated customer profile. Summary fields associated with a feature are allocated in a record only if the customer subscribes to the feature.

4.3 Limiting the size of ICP and summary fields

We store the entire ICP and summary data in a main-memory database to get the throughputs and response times needed for call processing and queries to the RTRI. We ensure that the customer profile is not too large and can be accommodated in main-memory of a small number of machines. In particular, as features proliferate the size of the customer profile should not explode.

We have two techniques for managing the storage requirements. First, customer profiles consume storage only for the features available to a particular customer. Features are available to a customer on account of an explicit subscription by the customer or because it is a "global" feature available to all customers (of a certain type). We recommend that the number of features of the latter variety be kept small, particularly those that require significant amounts of storage. The effort and cost to subscribe to special features can be expected to keep down the number of features per customer

Second, we suggest only using features that consume a fixed pre-determined amount of storage. Thus, the following features would *not* be supported within the RTRI server (these would be supported from the call detail database instead, where real-time responses are not guaranteed):

1. **Unbounded Destination Billing Summary:** How many minutes of calls have I made to a number n in the last 1 week, last 1 day, since last billed, on any particular day in the last month. (Where the number n was not pre-specified).

2. **Unbounded Daily Summary:** How many calls have I made on any one day in the past.

4.4 Distribution

The RTRI server consists of a cluster of machines operating in parallel. Section 6 considers the system parameters in light of the performance requirements. To achieve parallelization, we partition the ICP and associated summary data based upon the billing-id. Further, by introducing auxiliary call records (see below), we ensure that each call record can be processed by looking up exactly one customer profile record. Consequently, there is no need for communication between processors during call processing. Real-time queries on the RTRI are similarly restricted to access only one customer profile record. A demultiplexing function can be used to assign call records to the appropriate processor based upon the billing-id.

4.5 Auxiliary call Records

There may be multiple overlapping notions of who a customer is, and there may be individual customer profiles for each. For instance, a law firm may have a separate identification for each client account, and request a single comprehensive bill with discounts computed on the total but with charges to the individual client accounts, after taking the overall discounts into consideration, indicated individually. Similarly, the "Friendly Neighborhood" plan discussed in Sec. 9 requires a customer profile for the "neighborhood" in addition to one for each individual customer. For such services, the call record identifies one notion of a customer. The profile record for this primary customer can specify a secondary customer unit with which to compute the rating. The RTRI server rates the primary customer record, and then creates an auxiliary call record with the secondary customer unit as the billed entity, and feeds it back into the RTRI server as a part of the call stream.

Thus, the call recording workflow comprises a stream of call records received by the RTRI rating system, generating a stream of processed call records for the call detail database, and a stream of auxiliary call records that are fed back into the RTRI server, as shown in Figure 3.

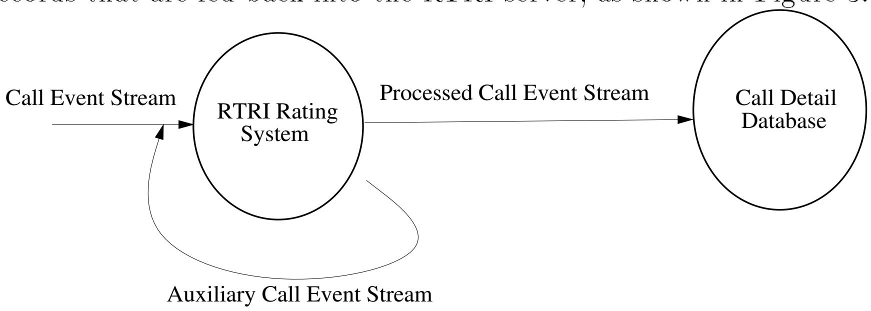

Figure 3: Call Recording Workflow.

5 The RTRI Server

The RTRI server consists of a cluster of identical machines working in parallel. Each RTRI machine handles a horizontal partition of the call records, and a horizontal partition of the ICP and summary data, partitioned by billing-id. A call distribution system distributes call records between the RTRI machine using the same criteria used to distribute the ICP and summary data.

Each RTRI machine runs the following processes, prioritized according to the order shown:

1. Real-time Querying Process. This process accepts queries submitted to the RTRI system by a switch during call set-up, and generates a subprocess (thread) to execute the query. All

such queries are written and compiled during service creation, and are stored as a part of the service data, as functions to be executed to answer the queries.

2. Call Processing Process. This process looks at the call record buffer, and processes each call record in turn. Each call record causes a lookup into the ICP and summary data, and subsequent execution of feature functions. The feature functions compute the rate for the call, the new values for summary fields, and possibly an auxiliary call record. A processed call record containing the rate is written to the output buffer in compressed form.

3. Lower Priority Querying Process. This process accepts queries submitted to the RTRI system by sources other than the switches, and generates a subprocess (thread) to execute the query. All such queries are written and compiled during service creation, and are stored as a part of the service data, as functions to be executed to answer the queries.

4. ICP Maintenance Process: This process updates the ICP in response to changes to SSCPs. The changes to service specific customer profiles may be communicated to this process individually or in a batched mode, at any frequency desired by the SSCP.

5. Rating Table Update Process: This process accepts updates to the RTRI rating tables.

6 System Parameters and Performance

We illustrate how a system configuration implementing the SUNRISE architecture can be selected by considering a hypothetical large provider of telephone services, named GCP - Generic Communication Provider.

We assume the following sizing parameters for the GCP network:

- Up to 100 million calls per day (1157 calls/sec), with a peak rate of up to 10,000 calls per second. Most calls causes an update of exactly one ICP. Only a small fraction cause multiple ICPs to be updated.
- Up to 100 million queries into the RTRI every day (1157 queries/sec), with a peak of 10,000 queries/sec. This number allows each call to make one query during call set-up. Other query sources work at much lower rates.
- Queries made by a switch during call set-up need a response time of less than 10 milliseconds.
- 50 million customers.
- A call record has an average size of 100 bytes.
- 0.1% of customer records are updated every day. Thus, there are 50,000 updates every day, or less than 1 update/sec.

In our prototype system, using Sparcstation 10 machines, we are able to handle about 500 calls/sec on each Sparcstation 10 machine. It is our projection that 20 Sparcstation-10 machines will suffice for real time rating for the entire workload. Each machine will need about 2GB of main-memory. When building a system of this size, it is however advisable to use large server machines. We believe that four large Sun Sparc or equivalent server machines will be able to handle the needs of GCP.

Since our architecture is scalable, we can easily add machines to deal with larger workloads.

Integrated Customer Profile Size. We assume an average customer needs 500 bytes. Large corporate customers can have up to 10^6 bytes, smaller residential customers may have less than 200 bytes. Since we assume 50 million customers, the integrated customer profile database has 25 GB. The indices may add another 5GB. The ICP presents the main requirement for memory on the system. Therefore a bank of 20 machines with 2 GB of main memory can accommodate the entire customer profile, along with a few indices, in main memory, with room to spare for other data sets and processes.

Some of the information in a customer profile, such as customer name and address, may not require real time access at main memory speeds. Such information can be moved to disk, and some savings in storage achieved thereby.

RTRI Transaction Rate. Allowing for a 10% overhead due to auxiliary call records generated by our system, the system-wide average call traffic coming into the RTRI is 1300 records/sec, with a peak of $11,000$ records/sec. Similarly, the query rate has an average of 1300 queries/sec, with a peak of $11,000$ queries/sec. The update rate due to view maintenance is about 1 transaction per sec. Since the call records are partitioned amongst the 20 workstations, we need an average transaction rate of 130 transactions per second a peak transaction rate of 1100 transactions per second, with about half the transactions being queries, and half being updates. The amount of work done by a transaction to process each call record is kept simple in our model – access a customer profile record, do local computation, and update the record. Experiments using the main-memory storage system show that we can support query rates of up to 2,000 transactions per sec, and update rates of up to 800 transactions per sec per SparcStation 10. These performance numbers show that the performance requirements of SUNRISE will be met comfortably with workstation sized machines. However, we emphasize, that an application of this size is better done on a smaller number of larger server machines.

Query Response Time. Each query requires an indexed lookup into the integrated customer profile stored in main-memory, and can be executed in about 1 millisecond. Further, the queries and updates can be processed faster than their peak arrival rate. Up to 550 queries/sec may be initiated by switches for the RTRI system, as the above analysis shows. These rates appear to be comfortably within our limits, including the requirement of a response time of under 10 milliseconds for queries initiated by the switches.

RTRI Transfer Rates. Assume that the incoming call records into the RTRI have an average size of 100 bytes. With the assumption that the average call traffic is 1200 records/sec, with a peak of $10,000$ records/sec, the average incoming transfer rate into the RTRI system is 0.12 MB/sec, with a peak of 1 MB/sec. Similarly, the output transfer rate from the RTRI system has an average of 0.12 MB/sec, and a peak of 1 MB/sec. We can obtain close to uniform distribution between the 20 machines, so these transfer rates can be supported easily.

Call Distribution. The call records have to be switched to the correct machine within the RTRI engine that is responsible for rating the call record. The call distribution may be done by a machine that reads the incoming call record, hashes on the telephone number, and outputs the records onto an ATM LAN operating at 100 Mb/sec. The RTRI machines can pick up the call records destined for the machine from the ATM LAN. With 100 byte call records at a peak rate of 1 MB/sec,the peak transfer rate on the ATM LAN will be 8 Mb/sec. Further, since the call records would typically be fed from several switches, we could distribute the hashing function between several small machines. Each of these machines hashes the call record and puts it on the ATM LAN with the appropriate destination address.

Call Detail Database. A processed call record is about 100 bytes, perhaps even smaller after compression. Thus the 100 million calls/day generate less than 10 GB of processed call data per day. Let us assume that data needs to be stored on-line for 90 days on disk. The total disk space requirement is 0.9 TB, with an input rate of 0.12 MB/sec on the average, and 1 MB/sec at peak. Special main-memory storage techniques are used for buffering, and grouping, to reduce the number of disk writes.

Besides processed call data, images of bills sent to customers are also stored on-line, in a compressed form. The call detail database is partitioned by a billing identifier across 20 machines, each of which is expected to have 50 GB of disk.

Detail on older telephone calls can be stored in optical jukeboxes. 3.6 TB of data is generated every year.

7 Service Creation

An important design goal is for new components to be added into the system by a service for integration into the ICP with minimal work by the service, and without stopping the system.

To support these, we must be able to evolve the scheme of a database while a system is running. We achieve such run-time schema evolution by the use of two devices. First, operations involved with any particular feature are not directly invoked as functions. Instead, a vector of function pointers is used, making it easy to link in new functions on-the-fly. Second the ICP object class is defined as a list of pointers to fields, with the pointers themselves being interpreted based on the services subscribed by the specific customer. The actual fields for the services subscribed are stored separately, with storage allocated only for fields that are used. Since relevant fields are linked through a list of pointers for each individual instance of the ICP, new fields can be introduced by a simple change to the pointer list.

An important side benefit of such a pointer based organization is that it is not necessary for each customer to allocate space for the fields required by every possible service. When a customer subscribes to a service, some additional fields may be required in the customer profile. These fields are added on simply by extending the pointer list appropriately.

In addition to traditional network usage services, it is possible in SUNRISE to support third party service providers as well. All that is required is that a "call " record of the appropriate type be generated for the service type. The new format need be registered at only one place, namely the SUNRISE system, and new ICP fields created for this service just as for any other network service.

There are a few standard functions that each feature must specify: a subscription function, an unsubscription function, definitions for zero or more fields in the ICP required by this feature, functions to compute call specific fields affected by this feature, and specification of additional fields in the processed call record required on account of this feature, if any. In addition, if auxiliary call records are to be generated by the rating process, the function to generate these must also be specified. A billing function, if applicable to this feature, such as a monthly fee, must also be specified. Subscription and unsubscription functions are not required for features that are universally provided to all customers of a particular type.

We use a graphical service creation interface. Since there is a fixed set of functions to be created for each feature, and since we anticipate the number of primitive operations required by these functions to be small, we believe that it will be possible to use a graphical specification mechanism, with no need for the feature creator to write the complete code.

7.1 Feature Interaction

ICP and summary field definitions are registered with our system when a new feature is defined. These fields can then be used by other features in turn. Moreover, the definition of one field can use another, previously defined field, provided that no cycles of dependency result. Our system keeps track of the features that use particular fields. A feature may modify the definition of a field only if no other feature is using it, directly or indirectly. Otherwise, the other affected features must have the service provider owning that feature approve the change.

Since fields are independently specified by features, and previously specified fields used only if they provide exactly the desired functionality, there is no possibility of a conflict between features as far as field definitions go. An optimization is to discover the use of equivalent fields by two features, with slightly different computation procedures even though their values are identical. Also we want to discover overlaps between fields. We know how to do this in some simple cases, and are working on a more general understanding. As a related matter, integrity constraints can be defined between field values, where appropriate, automatically maintaining one field as the sum of two others, etc.

Feature functions can read and write fields in the customer profile, as well as temporary variables created so that different features can cooperate, for instance, to accumulate charges on account of different features used in a call. As far as the summary fields in the customer profile are concerned, we simplify the problem by ensuring that each feature function always reads the old value, and if it writes into the profile, it writes the new value of the field. Multiple features are not allowed to update the same summary field in the customer profile.

As far as the vector of temporary variables is concerned, several different features can read and write these. We require that each feature specify a read set and write set of variables. These are used to determine conflicts between features. Since so many features read a field and write back the field after adding a quantity to it, and since such increments (by function specified amounts) commute rather than conflict, we suggest that feature functions explicitly declare fields that will be incremented. Knowing this will decrease the number of spurious conflicts flagged.

For each pair of features that (potentially) conflict, a check is performed to see if the conflicting field accesses can actually take place on a given call record. If two features can never co-occur, for instance, because one applies only for daytime calls and the other applies only to off-peak calls, there is no conflict.

If the system detects a potential conflict, it will flag the conflict to the service creator, and request a resolution. The service creator may realize that the pair of features can never conflict, and may so declare. On the other hand, if the features do conflict, one or the other feature can be modified to remove the conflict. For instance a "special area code" plan may be applicable to a call that also qualifies for a block discount plan. The feature definition of the block discount plan can be modified to state explicitly that calls covered by the "special area code" plan cannot be included in the block. Alternatively, the service creator can specify an order in which to execute the feature functions. For instance, a volume discount is computed *after* the call has been fully rated, including charges for features such as charge to credit card and directory assistance. An employee discount is applied after the volume discount has been computed.

To aid in the ordering of different feature functions, the computations performed by each feature are typically divided into two steps – one that computes the vector of temporary variables, and a second step that updates summary fields in the ICP. The final requirement is that the union of all the pairwise order specifications be an acyclic graph. The system can then apply the associated functions in topological sorted order.

7.2 Creating Information Services

Information services are created in a manner similar to rating services, except that there is no rating function, and no fields are added to the processed call record. Thus the tasks involved include (1) writing queries over the information maintained in the ICP, (2) defining extra fields to be maintained in the ICP to compute the extra information, (3) specifying the functions required to compute the new fields, and (4) specifying how information requests can be made, and who can make them.

7.3 Creating Billing Services

Billing services are created at two levels. A billing program is written with access to the call detail database, the ICP, and the summary data. For accessing the ICP and summary data, a billing service behaves just like any other information service. We define the fields to be added to the ICP, and the summary fields to be computed. This is done as described for information services above.

8 The Billing System

A billing function b, computes a final bill amount for a customer based on the integrated customer profile C and the collection of calls charged to the customer. The billing process, in addition, generates a bill including information from C and as much call detail data as required by the customer. (This data could be a complete listing of all calls, using the processed call records or only summary totals.) Billing is done in a system separate from the RTRI engine.

We store processed call records in a call detail database. Billing programs make the necessary selections of processed call records from the call detail database to compute the bill. Multiple billing programs may access records of a customer, potentially at different billing times. Each billing program has a start and an end pointer on the sorted list of records within a customer. The pointers identify the calls in the current billing period. Finally, the billing cycles could be coordinated and a single integrated bill provided.

Often, the billing function does not care about the details of specific calls, but instead merely is interested in certain summary information. The summaries required can be specified *a priori* and computed by the rating system. The billing system can query for the values of these summaries at the end of each billing cycle. For this purpose, billing programs can register with RTRI as information service applications. They can define certain fields they would like maintained, e.g., number of minutes during plan hours, total number of minutes, total rated calls. A query to access each one of the fields is defined.

For customers who require only summary data in their bills, all the information required to produce a bill may be found through such querying. Since such querying is much cheaper than querying the call detail database, this also presents us with an opportunity to offer a discount to customers who trust us and do not require call detail on every bill. The detail information is still available in the database should there be an error or a dispute.

Queries to summary data from a billing system poses a new challenge. Billing cycles typically are staggered over several days in a month, but we assume that they must always be coordinated with midnight every day. If billing cycles can have their phases staggered to be evenly spread through the day, we will have better balancing of load. However, we do not consider spreading the billing activity over the day in this paper. Our architecture assumes that roughly 2.5 million bills have to be computed every day, with summary values taken exactly until midnight. Our solution is to "double buffer" summary fields that must be zeroed at midnight: call records pertaining to calls placed before the deadline are aggregated into one version of these fields, and those pertaining to calls placed after the deadline are aggregated into the other version of these fields. Separate pointers from the customer id go to each set of fields. In the course of each call record processing, the correct set of fields to process is determined. No assumptions are made regarding the sequence in which call records arrive – out of order arrival is acceptable. The billing system queries must be delayed enough after the "zero" point to make sure that all call records for calls before the zero point have been recorded.

To further minimize the overhead of this determination, the records that are going to pass a "zero" point have a special bit marked indicating that it is necessary to determine the correct pointer based on the time of the call. This marking is done as late as possible before the "zero" point is reached. After the "zero" point is past, the billing system queries for the summary values with the old pointer structure, and at this time also deletes the old pointer structure, and erases the bit indicating that a time check is required.

9 Examples of Services

We consider three examples of rating, billing, and information services, and consider how they would be implemented in the SUNRISE system.

9.1 A Block Usage Discount Plan

Consider a plan with the following characteristics, which we have selected to loosely represent the "Reach Out" family of plans offered by AT&T.

- A monthly fee.
- 60 free minutes during "off-hours".
- A flat charge for additional minutes during "off-hours".
- "off-hours" = 7pm to 8am M-F, and all day SS.

Block Usage Plan: Creation

- Define new fields (GUI).
  - `Offhour_minutes`. Initialize – 0.

- Subscription function (GUI).
  - Add field `Offhour_minutes`, and initialize it.
  - Send subscription record to billing system.
- Unsubscription function (If data defined in ICP) (GUI).
  - Drop field `Offhour_minutes`.
  - Send unsubscription record to billing system.
- Billing date update (GUI).
  - `Offhour_minutes` = 0.
- Rating function (GUI).
  Check call time, if off-peak, check `Offhour_minutes`. Rate minutes up to 60 at zero charge, rest at flat charge.
- Compile and dynamically link in above fields and functions.

Block Usage Plan: Operation

- call record billed to customer c comes in.
- Check all subscribed services.
- Invoke rating functions of each service.
- Reach Out America's rating function gets applied.
- On Billing date – backup and apply billing date update.
  `Offhour_minutes` = 0.

9.2 A "Friendly Neighborhood" plan

All residents (or offices, shops) in a building, or on a block, can collect together and form a "friendly neighborhood". This "neighborhood" need not be restricted to a physically contiguous set of customers. The total number of minutes of calls made by all customers in that neighborhood is tracked, and at the end of the month, a discount is given to the neighborhood. The discount may be disbursed to the building community fund, or reflected in any agreed upon proportion (equally, proportional to usage) in the next monthly bill for each individual resident in the building. Such a service could, to some extent, mimic the function performed by resellers.

For this purpose, a neighborhood is defined as a new customer. The list of call records for the neighborhood is the union of call records for all customers in the neighborhood.

- New fields: `FN_Account`, `Running_Total`
- Subscription function: Create new neighborhood.
  - Create new customer entity - the neighborhood, with field `Running_Total`.
  - Initialize `Running_Total` = 0.
- Subscription Function: Enroll Participant in neighborhood N.
  - Add field: `FN_Account`.
  - Initialize `FN_Account` = N.
- Rating function 1: Create auxiliary call record.
- Rating function 2: Add call.length to `Running_Total`
- Billing date function: Initialize `Running_Total` = 0.
- Information services: Query `FN_Account.Running_Total`

9.3 Split Billing

Static Split Billing: "Split billing" occurs when two customers share the cost of a call. In *static split billing*, a customer c_1 specifies a list of preferred callers, $L = \{a_1, \ldots, a_k\}$, such that the cost of a call from a customer in list L is split equally, or in any predetermined ratio, between the caller and the customer c_1. ("Select-collect" is a special case of this service). When a call is placed by customer a_1 to customer c_1, there is no information in the call record about the call being a split-billed call. The call is processed by looking up the customer profile record for customer a_1, where an indication is stored about customer a_1 being in someone's split billed list. The call is rated as per the profile of customer a_1. Two options exist at this time: (1) Query the customer record for c_1 to check if a_1 is in his split billed list. If yes, the charge is split between a_1 and c_1, and both customers' profile records are updated. If no, the call is handled normally. (2) An auxiliary call record is created with c_1 as the billed number, and an indication that a_1 still has to be billed for the call. The auxiliary call record goes to c_1, where split billing status is checked. A fraction of the rate may be charged to customer c_1, and another auxiliary call record may be generated with a_1 as the billed number, so that the system may charge the remainder of the cost to customer a_1. The first approach is preferable in a centralized system. However, it does not scale well, and the second approach is likely to be necessary in any real implementation.

Dynamic Split Billing: In *dynamic split billing*, the called person (for outbound service), or calling person (for inbound service) punches in a code during the call and gives an indication of the dollar amount/number of minutes/percentage of cost that he is willing to share. The code may be punched in manually or by a computer, is captured by the switch, and is included in the call record sent out by the switch. The code may be dialed during call set-up, or after a call has been set-up. Capturing a code after call set-up may require hardware support not currently available in all telephone switches. We will require that one customer be the "primary" billed customer per call record. Usually this is the customer who would have paid for the call if split billing had not been invoked. The rating system will recognize the call record as needing split billing, will rate the call based upon the primary billed customer, and charge the primary billed number the full cost of the call minus the cost assumed by the secondary billed party. The secondary customer may be charged his share either by a direct update or by generating an auxiliary call record.

10 Conclusions

In this memorandum, we have described a data architecture to capture network usage records, enabling their use for innovative rating, billing, and information services. The proposed architecture holds out the promise of being able to deliver greater functionality in the recording and billing process, while saving cost.

References

[GLS93] Narain Gehani, Dan Lieuwen, and S. Sudarshan. MM-Ode: A Main Memory OODBMS. In *Proceedings of AT&T Database Day*, Oct. 1993.

[GMS93] Ashish Gupta, Inderpal Singh Mumick, and V. S. Subrahmanian. Maintaining Views Incrementally. In *Proceedings of ACM SIGMOD 1993 International Conference on Management of Data*, Washington, DC, May 26-28 1993.

[ISO93] ISO_ANSI. ISO-ANSI working draft: Database Language SQL3, February 1993.

[JLR+94] H.V. Jagadish, Daniel Lieuwen, Rajeev Rastogi, Avi S ilberschatz, and S. Sudarshan. Dali: A high performance main memory storage manager. In Jorge Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, September 12-15 1994.

[JMS94] H. V. Jagadish, Inderpal Singh Mumick, and Avi Silberschatz. The Chronicle Data Model. In PODS 1994.

[MRS93] Inderpal Singh Mumick, Kenneth A. Ross, and S. Sudarshan. Design and Implementation of the SWORD Declarative Object-Oriented Database System. In *Proceedings of AT&T Database Day*, Oct. 1993.

A Enhanced Services

We list several enhanced services based upon customer profile and call record data that SUNRISE is capable of supporting.

A.1 Rating Services

Here are some examples of potentially interesting rating and billing plans that the SUNRISE system can support.

1. **Specified Number Discount:**[2] Each customer keeps a list of twenty US and two International Numbers. The customer gets a 20% discount on all domestic numbers that are dial-1 subscribers. Further, if at least one of the 20 US numbers is a dial-1 subscriber, then the customer gets 20% discount on the two international numbers.

2. **Mutual Specified Number:** Let customer c list a friend f in his profile. Then, if f is a dial-1 subscriber, and lists c in his profile, c gets an extra 10% discount on all calls to f, in addition to the 20% specified number discount.

3. **Transitive Specified Number:** Let customer c list a friend $f1$ in his profile, and let $f1$ list a friend $f2$ in his profile. Then customer c gets a 10% discount on calls to $f2$.

4. **Mutual Transitive Specified Number:** Let customer c list a friend $f1$ in his profile, and let $f1$ list a friend $f2$ in his profile. If $f2$ lists customer c in her profile, than c gets an additional 10% discount on calls to $f1$ and $f2$. (A similar service has also been called "Circle of Friends").

5. **Destination Volume Discount:** A customer specifies a number a and a set of three numbers $\{b, c, d\}$ in his profile. If he makes more than 15 hours of calls to number a in a month, then he will get a 10% discount on all calls to numbers $\{b, c, d\}$.

6. **Select Collect:** A customer specifies a set of numbers or customer ids, such that any call from a number/customer in this set is automatically charged collect to him.

7. **Monthly Budgets:** A customer specifies the maximum dollar amount the customer will spend on calls, very much like one can specify the dollar amount for gas at a gas-station.

8. **Customer Profile based Debit Card:** How long a time slice should be allocated for a fixed dollar cost of a call. Such a query is likely to be made by a switch during call set-up. The projected cost of the call should reflect the customer's billing and feature selection, as well as the billing options of the dialed number (the dialed number may want this call charged collect).

A.2 Information Services

The following services require collection of call records in real-time, as the calls are completed, and the ability to access the customer profile in real-time. Access to real-time information can be used

[2]The *Friends and Family* plan offered by MCI has some similarity to this plan.

by the network to provide additional services, or it can be sold directly to customers. A customer may make these queries using a computer, a smartphone, or through the operator. A switch may make some of these queries during the call set-up process.

1. **Temporal Billing Summary:** How many minutes (or dollar cost) of calls have I made in the last 1 week, last 1 day, since last billed, or on any particular day in the last month.

2. **Last Call Information:** How much did I spend on the last call? What number did I call? What was the last call received by me? What was the last call billed to me?

3. **Destination Billing Summary:** A customer specifies a number/customer id, say C, in his profile, and can ask "How many minutes (or dollar cost) of calls have I made to person C in the last 1 week, last 1 day, since last billed, on any particular day in the last month."

4. **Profile Query:** What billing services do I subscribe to? What calling features do I have? What is my billing date?

5. **Usage Monitoring:** Market analysis groups often wish to study the usage characteristics of a "panel", comprising a large set of statistically selected subscribers belonging to a particular demographic category of interest. Analysts would like timely reports of daily, weekly, and monthly aggregate usage by members of the panel, on which they can do trend analysis.

6. **Fraud Prevention:** What is the total number of minutes of calls to the Dominican Republic billed to a given number within the past week. Such a query can be made during call set-up, and call completion denied if the response is a number above some threshold.

A.3 Billing Services

1. **Billing Summary Options:** Give a billing summary, without call detail information, at any required frequency.

2. **Called Person Name:** A customer can chose to get, in his bill, the name, address, etc. for the called number, provided the called number is a customer known to the system. The information can be provided at rating time through auxiliary call records, or at billing time by making a query into the ICP. Further, a customer can give, in his profile, a list of phone numbers and name pairs he wants used to convert phone numbers on his bill into names. The profile list is very useful for international numbers.

   The 800 customers can get a list of names and addresses of all people calling.