# Performance Modelling of Distributed Caching of Credit Information in Pre-paid Telecommunications Systems

Student: Patrick Hayden (W20026687). Supervisor: Dr. Brendan Jennings

Waterford Institute of Technology

**Abstract.** Telecommunications networks offering pre-paid communications services contain centralised databases in which customers' call credit information is stored. This information is incremented as customers purchase top-ups and periodically decremented as they use services. Increasingly network operators are offering prepaid data services, which typically require increased numbers of credit decrements and the ability to reserve amounts of credit for particular service sessions. To support this operators are introducing into their networks pre-paid platforms which cache customer credit information, leading to the problem of how to synchronise cache information with the "master" information stored in the network database. This paper discusses the development of algorithms/processes to maximise credit information synchronisation subject to the constraint of maintaining network performance at desired levels.

## 1 Introduction

A prepaid communications service is one where the subscriber pays for the use of services in advance. As the consumer makes use of services, the credit balance available for their continued use is decremented. When the credit balance reaches zero, no more services can be used until the subscriber replenishes their credit balance, a procedure known as *topping-up*. Due to the popularity of prepaid telecommunications services [2], operators are making more services available to prepaid customers in order to generate additional revenue. As the prepaid service offering becomes more complex, operators are introducing *Prepaid Service Platforms* (PSPs) to their networks in order to handle the increasing complexity.

In order to decide if a customer has sufficient credit available to continue using a service, the PSP must contact the *Service Control Point* (SCP). SCPs are very high performance computers with a very high throughput of service requests. They are responsible for managing and distributing services as well as routing calls. They allow service logic to be stored externally to switching systems and were originally provided on a per-service basis (i.e. one SCP per service) although this is no longer the case. The SCP accesses credit information stored in a *Service Data Point* (SDP) which is a database that stores subscriber data. It may be co-located with the SCP and for the purpose of this paper no distinction is made between the SCP and the SDP i.e. subscriber data is accessed using the SCP. Whether it is stored in the SCP or on a dedicated SDP is of no relevance to the topic under discussion.

When dealing only with voice calls these service requests to the SCP do not represent a substantial overhead since the frequency with which the remaining balance has to be checked does not need to be very high (for example it could be every 30 seconds) and the rate at which the balance decreases is linear and broadly known (e.g. if the call is to another mobile on the same network then the cost is $x$, if it is to a fixed line then the cost is $y$, etc.).

Other, non-call, service transactions may make more frequent requests for smaller amounts of credit than call transactions. Consider for example the case where the service that is being consumed is 3G access to the internet. If the cost is 1¢ per Mb to download data from the internet and a large page with multiple HTTP GET requests for images that appear on the page is accessed then multiple requests for small amounts of credit must be sent to the SCP in order to load that one page.

The consequence of frequent credit database access for small amounts is that the SCP may become overloaded while handling a multitude of service requests. Since the SCP is on the voice call transaction path, prevention of overloading is a key concern to a network operator. For this reason a caching mechanism may need to be implemented in the PSP to prevent SCP overload from occurring. Figure 1 shows subscribers accessing both voice and data services on a network using the Prepaid Service Platform.

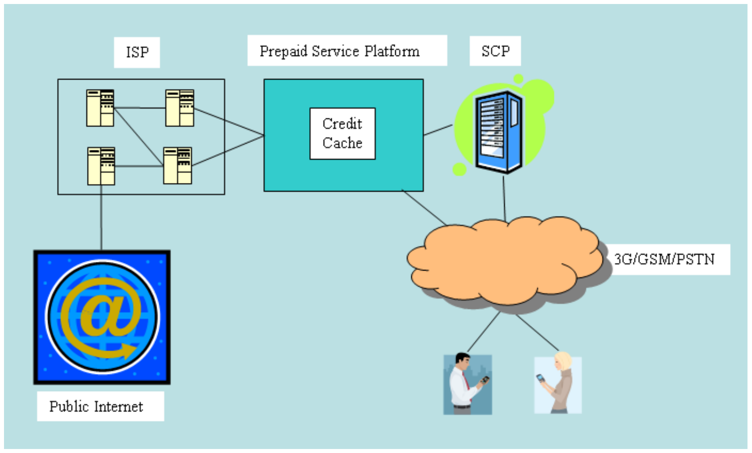

**Fig. 1.** Use of pre-paid services

This paper discusses a mechanism for synchronising a cache of credit balances in the PSP with the master data in the SCP in such a way that the SCP is not overloaded and that synchronisation between the cache and the master data is maximised. It is stuctured as follows: in section 2 we describe the research areas that are relevant to the subject under discussion; section 3 provides a detailed description of the problem; section 4 proposes an algorithm to control the flow of requests between the cache and the SCP; section 5 describes the experimental testbed; section 6 presents and analyses the results and section 7 concludes and summarises the work.

## 2 Background and related work

The problem described in this paper encompasses the areas of charging, load control and caching. An overview of these topics is provided with reference to existing publications.

### 2.1 Charging

Charging is an area in which mobile telecommunications networks and the internet differ greatly but as these two networks converge due to the move from circuit switched to packet switched networks and the encroachment of internet services into areas that were traditionally the preserve of the telecommunications network there is potential for overlap in the implementation of charging mechanisms[13].

**Mobile Telecommunications Network Charging Mechanisms** The mobile telecommunications network has well established charging systeems. It is owned and maintained by the telecommunications companies, access is limited to subscribers, use of services is subject to charges and a good Quality of Service (QoS) is guaranteed i.e. if a voice call is connected then call quality will be maintained at an acceptable level for the duration of that call, regardless of how busy the network becomes because the components of the network are designed to reject access to services if they become overloaded. The charging models can be broadly divided into *pre-pay* where the subscriber buys credit before using services on the network and where the subscriber's credit balance is reduced as they use network services and *contract* where the subscriber uses the services of the network and is charged periodically (for example, on a monthly basis) for their usage. Sometimes a contract includes an amount of service usage during the billing period for a fixed price (for example 90 minutes of voice calls or 50Mb of data) and usage outside these limits is added to the subscriber's bill. Unlike the current IP based systems, telecommunications networks have well established protocols (RADIUS, Diameter) for authentication, authorization and accounting (AAA) and these are linked to sophisticated charging systems. Mobile Network operators rely on billing, charging and accounting systems to operate their network[14].

The roll-out of 3G mobile telecommunications networks has led to research in how to apply new charging models to 3G networks and their associated services[7,14].

**Convergence with the Internet** For a number of reasons mobile telecommunications networks are converging with the internet. These include the changeover of the mobile telecommunications network from a circuit switched network to a packet switched IP network and the convergence of services that are offered on both networks. For example the biggest telecommunications network service, voice telephony is now available over the internet using Voice Over IP (VOIP) while Instant Messaging, and indeed VOIP services which were traditionally internet services are now offered on mobile telecommunications networks. New mobile devices such as the Apple® iPhone® are and Android® phones are also blurring the distinction between internet and mobile telecommunications networks. As mobile telecommunications networks converge with the internet it is argued that effort is needed to align the accounting protocols of both systems[13]. The Internet Engineering Task Force (IETF) has established the Authentication, Authorization and Accounting (AAA) Working Group to focus on the development of standards for the exchange of accounting information. The mobile telecommunications network providers have focussed on specifying the network components that generate charging information, mainly in the form of the charging functions specified as part of the Internet Multimedia Subsystem (IMS)[13].

The convergence of IP based networks with the telecommunications networks will mean that charging mechanisms will migrate from one network to the other. Pre-paid charging is a good example of this (at the end of the third quarter of 2001, prepaid users accounted for 50% of all subscribers worldwide[2]). Kurtansky *et al.* assert that "the customers' favor for prepaid charging will carry over to IP-based services"[8] and they evaluate variants of TICA (the Time Interval Calculation Algorithm) as a means of doing this.

## 2.2 Load Control

Due to a number of large and well publicized telecommunications network outages that occurred in the early 1990's[6], load control in SS7 networks has been the subject of a large amount of research. In his investigation of overload control algorithms for SCPs in the intelligent (SS7) network[15] Kihl contends that all nodes in the network must have an overload control mechanism and that the SCP must control the rate at which the SSPs accept IN requests.

The exchange of several messages between an SSP and an SCP during call setup results in congestion at the SCP due to the aggregration of messages from several SSPs in the SCP[5]. In order to allieviate this problem congestion control mechanisms such as call gapping are used to control the flow of input traffic[5,9,16]. Automatic Code Gapping (ACG) procedures are used in the SCPs to measure the load on the SCP and ACG requests are used to tell the source to apply a rate control[21].

Code gapping (also known as "call gapping") causes queries to the SCP to be blocked during the gap intervals. The gap intervals are determined by the SCP overhead control algorithms and include table driven controls and adaptive control[21]. Table driven control algorithms select gap intervals from a table stored in the SCP and adaptive control algorithms customize the gap settings according the load on the SCP and the transaction rates from the SSPs. Of the two, adaptive control was found to perform better under simulation[21].

An alternative throttling technique to call gapping is "percent blocking". Instead of blocking queries for a determined gap size as call gapping does, percent blocking blocks queries with a given probability. Percent blocking's strengths are it's robustness to changes in number of active sources (SSPs) and robustness to unbalanced loads[1].

Multiple SCPs have been proposed as a solution to the bottleneck of one central SCP and algorithms have been devised to perform load control using multi-SCPs[4] and to synchronise replicated SDP instances across multiple SCPs[19].

## 2.3 Caching

The term *"cache"* is derived from the French verb *"cacher"* meaning *"to hide"*. The Compact Oxford English Dictionary describes a cache as a *"hidden store of things"*[20] or in computing terms as *"an auxiliary memory from which high-speed retrieval is possible"*[20]. The term was first used in a computing publication to describe a high speed buffer in an article in the IBM Systems Journal in 1968[22] and subsequently came into common usage in the computing sphere.

In the case of a single computer a cache could be an area of physical memory where data is stored so that it is available to programs without relatively time-consuming access to the disk. In terms of distributed systems a cache could be a file on a local disk which is quicker to access than remote data that needs to be retrieved over a network. This also has the positive effect of reducing network traffic and reducing the load on servers and for this reason caching is commonly used on the internet to store static and frequently accessed data.

Caching introduces it's own set of terminology and concepts. A *"cache hit"* refers to data that is available in the cache when it is requested by a computer program. The corollary, a *"cache miss"* occurs when data that is retrieved is not present in the cache and must be retrieved from an external store. Data that has been modified in the cache but has not been updated in the external store is referred to as *"dirty"* and will be persisted in the external store when the data is removed from the cache or at intervals when the data in the cache is reconciled with the data in the store.

Caching is effective because computer programs frequently access the same or related storage locations. This is known as *"locality of reference"*[3]. There are various types of locality of reference. Spatial Locality "implies a high probability of reference to neighbouring addresses or pages"[11]. In the case of the distributed caching of credit information the data that are accessed by a program will be accessed again within a very short period of time. This is known as *"temporal locality"*.

*"Pre-fetching"* is closely associated with caching. It is a mechanism to increase the cache hit-rate by speculatively fetching data for the cache. Spatial locality of data is useful when designing pre-fetching algorithms so that data

that are referenced by or close to data that has been previously fetched can be speculatively pre-fetched in order to prevent cache-misses from occurring.

There are many different ways to implement caching and many different pre-fetching algorithms. In their survey of caching and pre-fetching techniques in distributed systems[11] Liu and Maguire identify the following cache design issues:

– Cache Location - *in local memory or on local disk.*
– Cache Unit Size - *the granularity of the cached data.*
– Cache Replacement Policies - *how to replace an existing entry in the cache to make way for a new one.*

Of these, the cache location and cache unit size are not relevant to the problem under discussion since the cache is located in the pre-paid platform and the size of the data are constant i.e. a floating point number representing the balance remaining in a customers account. The cache replacement policy and the size of the cache (number of entries) are of more interest when designing an algorithm that protects SCPs from overload.

One application where caching is used extensively is the world wide web. Indeed, "without caching, the WWW would become a victim of it's own success"[12]. The exponential growth of the world wide web has led to lot of research in the area of web caching. In their paper on web caching architectures[18] Rodriguez *et al.* describe two web caching architectures - hierarchical and distributed caching. In hierarchical caching, caching occurs at a number of different levels from local browser caches at the bottom through institutional level to regional level to national level. In distributed caching there are no intermediate caches, only institutional caches which are at the edge of the network and which serve each others misses using such protocols as the Inter Cache Protocol (ICP) (although this particular protocol has very high overhead as the number of proxy caches increases[10]).

While it is clearly a very useful concept, caching is not without it's problems. The existance of many copies of the same shared data in distributed caches introduces the problem of maintaining consistency across multiple copies of the data [11]. This is known as the *"cache consistency problem"*. The problem of mainainting cache consistency between the pre-paid platform and the master data in the SCP while minimising balance reconciliation is an important issue in the distibuted caching of credit information in pre-paid telecommunications systems.

## 3 Problem Description

A high load of service requests for customer credit decrements from the Prepaid Service Platform (PSP) to the SCP has the potential to overload the SCP. In order to prevent SCP overload from occurring a cache of credit balances may be implemented on the PSP. The implementation of such a credit cache means that it is necessary to also implement a mechanism to synchronise the credit values stored in the cache with the master data stored in the SCP. Cache synchronisation must be implemented in such a way that the following goals are achieved.

1. The size of the cache should be sufficient that the SCP does not become overloaded when the service request load on the PSP exceeds the specified processing capacity of the SCP.
2. The cache balances must be reconciled in such a way that reconciliation requests do not overload the SCP.
3. The cache balances must be reconciled in such a way as to maximise synchronisation between the cache and on the SCP.
4. The cache balances must be reconciled in such a way that the customer should not be able to overdraw available credit by simultaneously consuming credit in the cache and on the SCP in parallel service sessions.
5. The cache balance reconciliation algorithm must allow for credit top-ups in parallel service sessions on the SCP.

To realise the above goals a traffic flow algorithm must be designed that schedules reconcile requests in such a way that the SCP is protected from overload whilst simultaneously processing service requests. A number of strategies for prioritising which entries in the cache should be reconciled in any reconcile interval also need to be investigated.

## 4 Credit Cache/SCP Flow Control Algorithm

An algorithm is required to control the traffic flow between the credit cache and the SCP. The objectives of such an algorithm are twofold:

1. ensure that the SCP does not overload
2. maximise the degree to which entries in the cache are synchronised with the master copies of the entries as held by the SCP

### 4.1 Assumptions

In specifying the algorithm we make the following simplifying assumptions. First we assume that service sessions require only a single credit query (for example, if the subscriber has credit over a certain threshold then the service can proceed, even if there is some chance that the actual credit consumed by the session will exceed that amount or that the credit will be independently consumed by another service session executing in parallel). We assume that service sessions arrive according to a Poisson arrival process (this is a valid assumption given that we can consider that there are a large number of subscribers who can make a service request at any given time).

We assume that all requests incur the same, constant service time at the SCP and that the maximum SCP service rate, in terms of requests per second, is known at the credit cache.

## 4.2 Algorithm Specification

We have two types of request that are sent to the SCP:

– service requests which result in the creation of a new cache entry
– reconciliation requests relating to the reconciliation of a cache entry with the corresponding SCP entry

Service Requests *must* always be serviced – if they are not then the user will not receive service even if he/she has sufficient credit. On the other hand reconciliation requests are discretionary, in the sense that they do not relate to realtime service session requests. Instead, they serve to minimise the degree to which cache entries deviate from the corresponding entries at the SCP.

Given this our algorithm must perform four tasks. Firstly, it must control the number of reconcilation requests forwarded to the SCP in a given time interval so that the SCP does not overload. Secondly, given the number of reconciliation requests it is allowed to forward in a given time interval it must select which entries in the credit cache to reconcile with the SCP entries. Thirdly, if the cache size is limited then it must select which entries should be selected for removal from the cache. Fourthly, it should ensure that the stream of requests sent to the SCP is relatively smooth, to avoid bursts of requests that could result in transient overloads with consequent violation of maximum delay targets for processing of service session requests.

Let the algorithm be executed very $T$ seconds, where $T$ is a control interval. We now describe the four tasks that are performed every time the algorithm executes.

**Task 1: Estimation of Allowed number of reconciliation requests** Given the assumption of a Poisson arrival process for service requests, at time $t$ the number of such requests that arrived in the time interval $(t, t-T)$ is as valid an estimate as any for the number of arrivals that will arrive in the interval $(t, t+T)$. When a service request arrives, *if* there is already an entry for the subscriber in question in the cache, the request can be handled locally. However, if no cache entry exists then a request must be sent to the SCP, the response to which will result in the creation of a new cache entry.

Therefore, the number of requests sent to the SCP in the interval $(t, t+T)$ can be estimated by the number of service requests in interval $(t-T, t)$, denoted $r(t-T, t)$, times the probability that a cache entry did not already exist for the subscriber in question. This probability can be estimated as the mean number of cache entries for the interval divided by the total number of subscribers. This is calculated as follows.

Let $N_{cache}(t)$ denote the number of entries in the cache at time $t$. The estimate mean number of entries in the cache during interval $(t, t+T)$, denoted $\hat{N}_{cache}(t, t+T)$, is then given by:

$$\hat{N}_{cache}(t, t+T) = N_{cache}(t) + \frac{N_{cache}(t) - N_{cache}(t-T)}{2} \qquad (1)$$

The probability of a cache entry existing for a subscriber $s$, denoted $e_(s)$, in the interval $(t, t+T)$, denoted $P_{e(s)}(t, t+T)$, is estimated as:

$$P_{e(s)}(t, t+T) = \frac{\hat{N}_{cache}(t, t+T)}{N_{sub}} \tag{2}$$

where $N_{sub}$ denotes the total number of subscribers that can make service requests and for which the SCP holds credit information. Given this, the estimate for the number of requests to be sent to the SCP in the interval $(t, t+T)$, denoted $\hat{R}_{SCP}(t, t+T)$, is:

$$\hat{R}_{SCP}(t, t+T) = r(t-T, t)\big(1 - P_{e(s)}(t, t+T)\big) \tag{3}$$

Finally the number of reconcilation requests that can be sent to the SCP during $R_{recon}(t, t+T)$

$$R_{recon}(t, t+T) = C_{SCP} - \hat{R}_{SCP}(t, t+T) \tag{4}$$

where $C_{SCP}$ denotes the total number of requests that can be handled by the SCP in the interval T.

**Task 2: Proiritisation of cache entries for reconciliation** A number of different metrics can be used as the basis for prioritisation of cache entries for reconciliation. These include the following which were selected for analysis.

- *Ascending Service Request:* ascending order of the time since the subscribed last generated a service request – on the basis that users having recently used services that changed the credit amount stored in the cache are more likely to have entries whose cache entries are at variance with the SCP values.
- *Descending Reconcile:* descending order of the time since the cache entry was created / last reconciled – on the basis that the longer it is since the value was reconciled the greater the chance that the credit values differ.
- *Ascending Credit:* ascending order of the amount of credit stored in the entries – on the basis that there is greater risk that customer top-ups are not reflected in cache entries when users attempt to access services.
- *Descending Credit:* descending order of the amount of credit stored in the entries – on the basis that there is a greater risk of revenue leakage if the user is depleting a significant amount of credit via the cache whilst simultaneously depleting the same amount of credit via the SCP.
- *Random:* purely random prioritisation of cache entries.
- *Greatest Variation:* greatest variation between the current cache balance and the last known SCP balance – on the basis that divergence between the cache balance and the SCP balance should be as little as possible.
- *Cache Order:* the order that the entries occur in the cache.

**Task 3: Removal of entries from the cache** Where the cache size is limited it is necessary to remove entries in order to create space for newer cache entries. Strategies for removal of cache entries could include the following:

- removal of older entries from the cache – on the basis that newer entries are more likely to represent current service sessions.
- removal of entries in descending order of time since the last service request was received from that customer – on the basis that a longer time since the last service request indicates that the service session is no longer active
- removal of entries in decending order of the amount of credit stored in the cache – on the basis that entries with more credit are less likely to become overdrawn.

**Task 4: Dispatching of reconciliation requests** As noted above all service requests where there is no cache entry must be immediately forwarded to the SCP. We assume that no load controls are applied to these requests, thus the forwarding of these requests to the SCP in and of themselves will not result in overload of the SCP.

Our algorithm must control the rate at which reconciliation requests are forwarded to the SCP. The simplest approach is to distribute these requests evenly in time across the control interval. Let $\tau_{recon}$ denote the delay between successive dispatches of reconciliation requests. We then set:

$$\tau_{recon} = \frac{R_{recon}(t, t+T)}{T} \tag{5}$$

Thus, the first reconciliation request of the control interval will be dispatched at time $t + \tau_{recon}$, whilst the last one will be sent at time $t + T$.

## 5 Experimental Testbed

### 5.1 Simulation Approach

A simulation approach to modelling the problem was used. The proposed solutions were not tested in a real test-bed and no real customer data were used.

Simulation involves modelling real world processes, in this case using software, and specifying inputs to generate an history of how that system performs over time. By specifying different inputs, a series of histories containing the response of the model to different inputs can be built up. By analysing these histories, inferences can be made about how the real world system that is being modelled would respond if the same inputs were applied.

Discrete event simulation is the analysis of the response of the system to a chronological sequence of events. It is the most common form of simulation and is particularly suited to simulating telecommunications systems where events tend to be of a discrete nature, for example, user dials a telephone number.

In the case of the subject of this research topic, an example of a discrete event could be “increase load to 150% when maximum cache size is 50%”.

OPNET Modeler® [17]simulation software was used to model the real-world process. OPNET Modeler® is designed specifically for modelling communication networks. It is an object oriented development environment with a graphical interface for modelling the network. Process models are expressed in a language called Proto-C which is based on the C/C++ programming language and incorporates state transition diagrams and a library of over 300 high level commands known as *Kernel Procedures*.

## 5.2 Simulation Model

The simulation model contains two nodes, one representing the Prepaid Service Platform (PSP) and the other representing the SCP. The PSP, which contains the cache, generates service requests that are serviced by the SCP and the results sent back to the PSP. The services are of three different types, A, B and C, each with a different associated cost. The service request generators are configured to produce the load profile shown in Figure 2.

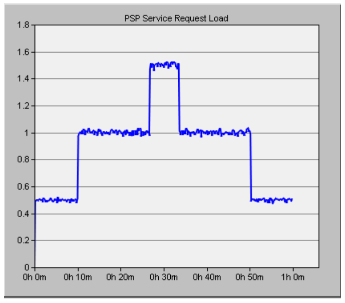

**Fig. 2.** Simulated load profile. A value of 1 represents 100% SCP load

This is a stepped load profile which simulates a situation where the normal load was 50% of SCP capacity with a usage spike resulting in a rise to 100% of SCP capacity and for a short period to 150% of SCP capacity. This load profile causes SCP overload in the situation where no caching is present and can be used to analyse how effective the cache is in preventing SCP overload and to evaluate the performance of the various prioritisation strategies in maximising synchronisation between the cache and the SCP.

Credit top-ups and service requests are also generated at the SCP in order to demonstrate the effect of parallel service sessions which adjust the credit balance at the SCP independently of the cache. These parallel service sessions are ignored for the purposes of calculating SCP load. They are only used to measure the effect of parallel service sessions on cache synchronisation.

When developing the simulation model, the following assumption was made:

– the problem space is limited to the link between the credit cache and the SCP. The impact on the larger telecommunications network is not included in the problem space.

### 5.3 Running the Simulation

The simulation runs for a simulated time of one hour (3600 seconds). The following independent variables are used to conduct experiments using the simulation model:

1. The maximum cache size.
2. A flag indicating whether parallel service sessions are active.
3. The prioritisation strategy used.

In addition there are a number of controlled variables which are not changed. If required, these could be treated as independent variables in order to investigate different scenarios that are not covered here. These are:

1. The control interval, $T$, which was set to 1 second.
2. The SCP Capacity which is set to 500 packets per second.
3. The generation of service requests which creates the load profile. The load profile remains constant.
4. The total number of subscribers, $N_{sub}$, which is set at 10000.

While the simulation is running, statistics are gathered by the various processes at intervals of 10 seconds.

## 6 Results and Analysis

### 6.1 No Cache

When no cache is used, the usage spike in the stepped load profile clearly demonstrates SCP overload as shown in figure 3.

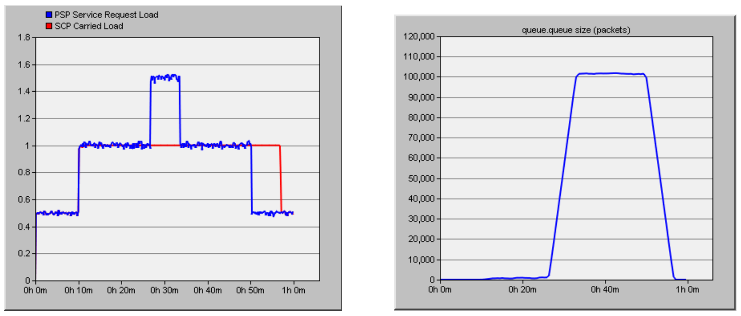

**Fig. 3.** SCP load when no cache is used

The results show how the SCP load remains at 100% (since it has a finite processing capacity) as the offered load from the PSP increases to 150%. The consequence of this is that service request packets from the PSP are put on the SCP's internal queue which grows in size for the duration of the usage spike, levels off when offered load drops back to 100% and finally reduces when offered load drops back to 50% and the backlog of requests in the queue can be processed. Note how the SCP continues operating at 100% load until the backlog has been cleared.

### 6.2 Unlimited Cache Size

In order to service the usage spike without overloading the SCP a cache of unlimited size is introduced on the PSP. Entries are reconciled using a control interval of one second with the flow control algorithm that has been described. At this stage the prioritisation strategy that is used to select the cache entries that should be reconciled is not important as we are only interested in the effect of the load on the SCP and not on the level of synchronisation between the cache and the SCP. The results are shown in figure 4.

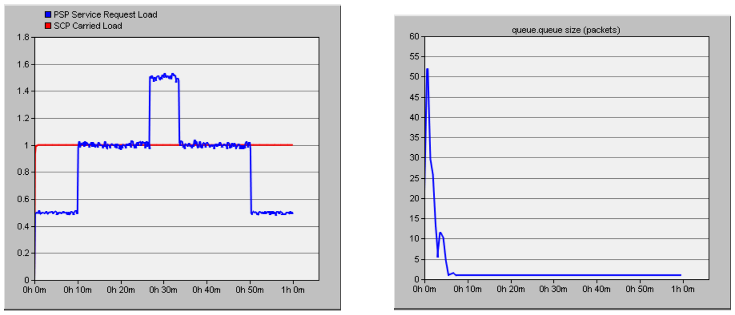

**Fig. 4.** SCP load when cache size is unlimited

The effect of the cache can be seen in the absence of any sizeable queue of unprocessed service requests forming on the SCP. In accordance with the implementation of the flow control algorithm, the SCP load remains constant at 100% with reconcile requests filling the un-used capacity left by the reduced number of service requests. Figure 5 shows the breakdown of the total load offered to the SCP into service request and reconcile request load.

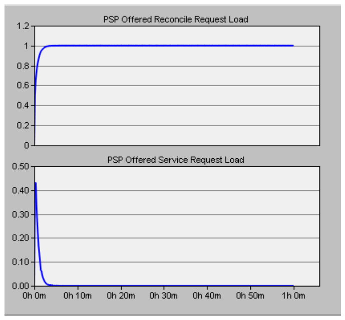

**Fig. 5.** Breakdown of service request and reconcile request load with unlimited cache

The graph shows that the amount of service requests being offered to the SCP declines sharply as the cache is filled which leads to a corresponding increase in reconcile requests. As the cache size is unlimited there are no further service requests to the SCP once the cache has been filled. This shows the success of the flow control algorithm in preventing SCP overload when the cache size is unlimited.

## 6.3 Limiting the Cache Size

In reality, due to the number of subscribers in a telecommunications network it may not be possible to operate with an unlimited cache size. The effect of limiting the cache size to 50% and 20% of subscribers on SCP load was tested under the same conditions as unlimited cache size.

*Note:* In order to limit the cache size, a strategy was necessary to remove entries from the cache. A simple removal strategy was chosen which removes the oldest cache entry (the one with the longest interval since it was last reconciled) which has had no balance decrements while in the cache (the cache balance is the same as the last known SCP balance and a reconcile request is therefore not necessary).

The results for 50% cache are shown in figure 6.

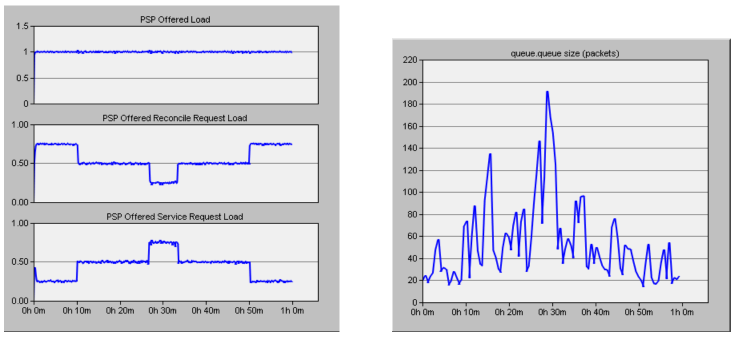

**Fig. 6.** SCP load when maximum cache size is 50%

The service request load offered to the SCP is the same profile that is being received by the SCP from the packet generators but reduced by 50%. For example when the service request load received by the SSP peaks at 150% of SCP capacity, the service request load on the SCP is 75%. At the lowest load of 50%, SCP load is 25%. We can extrapolate from this that a cache size of 50% of subscribers will protect the SCP from overload until service requests arriving at the PSP exceed 200% of SCP capacity. The SCP queue size does indicate that some overloading occurs as the algorithm adjusts to the spike in service requests from the previous interval but the maximum queue size of under 200 packets does not represent a significant overloading of the SCP which can process 500 requests per second.

For 20% cache the results are shown in figure 7.

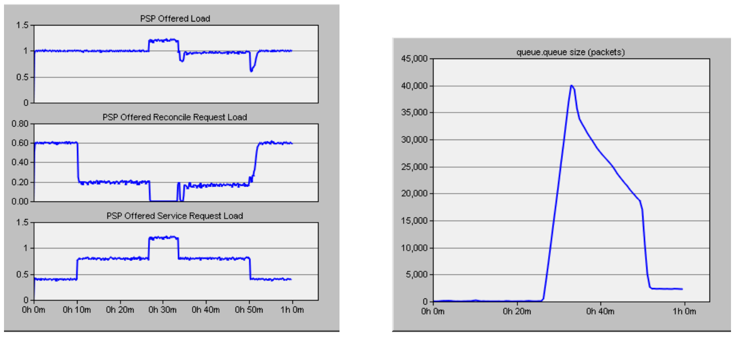

**Fig. 7.** SCP load when maximum cache size is 20%

It is noticable that this cache size does not protect the SCP from becoming overloaded when the service request load reaches 150% of SCP processing capacity. At this point the number of reconciliation requests drop to zero but the service request load increases to 127%, reflecting the amount of cache misses that occur with this cache size. The internal SCP queue contains 40000 packets at the apex of the offered load which indicates significant overloading of the SCP. Using a 20% cache size also shows an under-utilisation of the available SCP capacity at the points where the load decreases.

It might be assumed that this delay in returning the offered load to 100% of SCP capacity when the number of service requests decreases is a feature of the flow control algorithm. However this is not the case. The delay is too long to be caused by the flow control algorithm which is run at a control interval of 1 second and which would therefore result in a delay of only 1 second before $r(t-T,t)$, the number of service requests in the previous interval, would also be an accurate approximation for the number of service requests arriving in the next interval, $r(t+T,t)$. Rather this time lag in returning to 100% utilization is caused by the fact that the SCP needs to clear its internal queue of requests before it will start processing new ones. When the service request load received by the PSP drops, more reconciliation requests are scheduled in the next interval. These reconcile requests are then added at the end of the internal queue on the SCP behind a large backlog of service requests. When a reconcile request is sent for a cache entry, a flag is set in that cache entry to indicate that it is reconciling and should therefore not be scheduled to reconcile again until a response is received from the SCP. Since all reconcile requests will be at the end of the internal SCP queue, most of the entries in the cache will be in state *reconciling* for a relatively long time before a response is received from the SCP and there will therefore be a limited number of entries available in a small cache to schedule for reconciling. When the SCP has finished processing the backlog of service requests in its internal queue it will begin to process these reconciliation requests. This will result in reconcile responses being sent to the PSP which in turn will make more cache entries available for further reconciliation scheduling.

There is also a residue of entries in the internal queue of the SCP which are not cleared due to the scheduling of new reconciliation requests. The flow control algorithm may need to be changed to use a more adaptive approach to scheduling reconconciliation requests in order to address this problem. An adaptive approach is discussed in section 7.

## 6.4 Comparing the Reconciliation Prioritisation Strategies

In order to compare the performance of the various reconciliation prioritisation strategies described in section 4.2 it is useful to try to measure the level of synchronisation between the cache and the master data in the SCP. In order to do this, the following values were measured:

– **Mean Balance Variance:** The mean variance between the SCP balance and the cache balance.
– **Highest Mean Balance Variance:** The highest value of the above – occurs at the highest service request load.
– **Maximum Balance Variance:** The maximum balance variance in any entry in the cache during the simulation.
– **Longest Unreconciled Cache Time:** The longest time that an entry remained unreconciled in the cache.

The results of running the simulation using the different prioritisation strategies are shown in the following tables.

| Prio. Strategy | Mean BV | High Mean BV | Max. BV | Longest Unrec.(sec) |
|---|---|---|---|---|
| Asc. Service Req. | 0.06 | 0.16 | 2.60 | 588.88 |
| Desc. Reconcile | 0.03 | 0.09 | 1.25 | 20.70 |
| Ascending Credit | 5.61 | 11.02 | 17.25 | 3599.90 |
| Descending Credit | 5.49 | 10.82 | 17.25 | 3599.90 |
| Random | 0.06 | 0.17 | 2.55 | 270.63 |
| Greatest Variance | 0.00 | 0.02 | 0.60 | 3598.92 |
| Cache Order | 0.03 | 0.09 | 1.40 | 39.25 |

**Table 1.** 100% cache; T = 1.0

| Prio. Strategy | Mean BV | High Mean BV | Max. BV | Longest Unrec.(sec) |
|---|---|---|---|---|
| Asc.Service Req. | 0.16 | 0.67 | 5.40 | 739.62 |
| Desc. Reconcile | 0.03 | 0.15 | 1.50 | 29.60 |
| Ascending Credit | 5.25 | 10.56 | 16.35 | 3598.99 |
| Descending Credit | 4.80 | 9.30 | 16.60 | 3598.96 |
| Random | 0.08 | 0.35 | 3.85 | 353.47 |
| Greatest Variance | 0.01 | 0.11 | 0.80 | 422.97 |
| Cache Order | 0.05 | 0.23 | 2.0 | 79.82 |

**Table 2.** 50% cache; T = 1.0

| Prio. Strategy | Mean BV | High Mean BV | Max. BV | Longest Unrec.(sec) |
|---|---|---|---|---|
| Asc. Service Req. | 0.44 | 4.04 | 7.20 | 1413.20 |
| Desc. Reconcile | 0.29 | 3.67 | 6.45 | 486.12 |
| Ascending Credit | 1.62 | 6.95 | 11.40 | 2098.51 |
| Descending Credit | 1.46 | 5.95 | 10.80 | 2098.03 |
| Random | 0.33 | 3.73 | 6.50 | 622.19 |
| Greatest Variance | 0.29 | 3.64 | 6.45 | 503.46 |
| Cache Order | 0.32 | 3.73 | 6.60 | 511.13 |

**Table 3.** 20% Cache; T = 1.0

Looking at the results, the strongest performing prioritisation strategy in terms of limiting the variance between the cached balances and the master balances on the SCP, regardless of the cache sizes used in the tests, is the greatest variance algorithm. The worst performing strategies in all cases are the ascending and descending credit strategies. For poorer performing strategies, a smaller cache size limits the mean balance variance because there are less entries in the cache so a greater percentage of those algorithms are reconciled each time. This is not the case for the better performing strategies.

### 6.5 Effect of Parallel Service Sessions

Having studied the performance of the prioritisation strategies in maximising cache synchronisation in the simple case where parallel service sessions were not considered, it was then necessary to also analyse their performance in situations where the credit balance on the SCP was being adjusted by parallel service sessions. This was done by simulating services that decremented the credit balances on the SCP and also top-ups which replenished the credit balances on the SCP. These parallel service and top-up requests occured at intervals of 0.1 second and decremented a random customer account balance by 0.05 or increased it by 10.0 respectively. In order to measure the effect it was necessary to introduce some new measurements alongside balance variance. These are:

- **Incorrectly Allowed Service Requests (Inc. All.):** These are service requests which are allowed by the PSP based on the cached customer credit balance but which would have been denied by the SCP because its customer credit balance is lower than the cache balance due to balance decrements in a parallel service session.
- **Incorrectly Denied Service Requests (Inc. Den.):** These are service requests which are denied by the PSP based on the cached customer credit balance but which would have been allowed by the SCP because the customer has topped up their account in a parallel service session.
- **Mean Total Overdrawn (MTO):** This is an average of the total amount overdrawn at each statistics gathering interval. An account is overdrawn when the master account balance in the SCP less cache adjustments that have yet to be reconciled is below 0.
- **Highest Mean Total Overdrawn (High MTO):** This is the highest value of MTO that occurred when running the simulation.

The results of running the simulation using the different prioritisation strategies are shown in the following tables.

| Prio. Strategy | Mean BV | High MBV | Max. BV | Inc.. All. | Inc. Den. | MTO | High MTO |
|---|---|---|---|---|---|---|---|
| Asc. Service Req. | 0.49 | 0.65 | 40.15 | 15 | 448 | 0.00 | 0.07 |
| Desc. Reconcile | 0.13 | 0.20 | 30.05 | 3 | 124 | 0.00 | 0.05 |
| Ascending Credit | 19.05 | 35.06 | 132.75 | 0 | 5 | 0.00 | 0.00 |
| Descending Credit | 21.98 | 44.15 | 139.90 | 224 | 87260 | 4.17 | 8.09 |
| Random | 0.26 | 0.39 | 30.45 | 4 | 247 | 0.00 | 0.05 |
| Greatest Variance | 0.48 | 0.99 | 90.00 | 10 | 18558 | 0.01 | 0.16 |
| Cache Order | 0.14 | 0.21 | 30.00 | 2 | 122 | 0.00 | 0.04 |

**Table 4.** 100% cache; T = 1.0; with parallel service sessions

| Prio. Strategy | Mean BV | High MBV | Max. BV | Inc. All. | Inc. Den. | MTO | High MTO |
|---|---|---|---|---|---|---|---|
| Asc. Service Req. | 0.52 | 1.41 | 34.10 | 2 | 37 | 0.00 | 0.06 |
| Desc. Reconcile | 0.10 | 0.31 | 20.90 | 1 | 33 | 0.00 | 0.00 |
| Ascending Credit | 14.89 | 27.27 | 113.85 | 0 | 1 | 0.00 | 0.00 |
| Descending Credit | 18.95 | 39.04 | 118.55 | 176 | 59335 | 3.37 | 6.70 |
| Random | 0.22 | 0.71 | 30.75 | 4 | 25 | 0.00 | 0.05 |
| Greatest Variance | 0.10 | 0.31 | 20.50 | 2 | 55 | 0.00 | 0.05 |
| Cache Order | 0.14 | 0.46 | 30.20 | 0 | 18 | 0.00 | 0.01 |

**Table 5.** 50% cache; T = 1.0; with parallel service sessions

| Prio. Strategy | Mean BV | High MBV | Max. BV | Inc. All. | Inc. Den. | MTO | High MTO |
|---|---|---|---|---|---|---|---|
| Asc. Service Req. | 1.24 | 9.90 | 45.15 | 5 | 77 | 0.01 | 0.13 |
| Desc. Reconcile | 0.66 | 8.35 | 44.95 | 2 | 25 | 0.00 | 0.01 |
| Ascending Credit | 5.62 | 19.27 | 69.20 | 0 | 7 | 0.00 | 0.00 |
| Descending Credit | 6.60 | 21.90 | 85.00 | 100 | 11666 | 0.85 | 4.15 |
| Random | 0.74 | 8.45 | 34.40 | 4 | 94 | 0.02 | 0.37 |
| Greatest Variance | 0.66 | 8.23 | 44.70 | 2 | 40 | 0.00 | 0.01 |
| Cache Order | 0.72 | 8.71 | 44.95 | 3 | 58 | 0.10 | 0.15 |

**Table 6.** 20% cache; T = 1.0; with parallel service sessions

**Analysis of Results** When the effect of parallel service sessions is included in the simulation, the greatest variance prioritisation strategy still performs best in terms minimising balance variance between the cache and the master data in the SCP except in the case of unlimited cache size where it is outperformed by cache order.

In terms of overdrawn accounts, greatest variance also performs well in keeping them to a minimum. However with regards to overdrawn accounts, the best performing strategy is ascending credit prioritisation, a strategy which has much poorer performance in minimising balance variance. Ascending credit prioritisation is shown never to incorrectly allow a service request and therefore never to cause an overdrawn account regardless of cache size. In addition ascending credit prioritisation is shown to have fewest incorrectly denied service requests when compared to any of the other prioritisation strategies. In comparison, in the worst case, the greatest variance strategy resulted in the incorrect denial of 18558 service requests in a one hour simulation. Of the other strategies, descending reconcile, random and cache order perform well with all cache sizes and produce consistant stable results. The performance of descending credit balance prioritisation remains poor.

The reason for the improvement in the performance of the ascending credit strategy in this set of tests compared to its performance when parallel service sessions are not considered may be that top-ups cause a greater turnover of entries in the reconcile list produced by this strategy compared to the situation where there are no top-ups. Using this strategy, the lowest cache balances are reconciled first. If one of these balances has been replenished by a parallel service session then a higher balance will be returned to the cache as a result of the reconcile request. This will result in the entry which now has a higher balance being moved further down the reconcile list and therefore another entry will have the chance to be reconciled. The fact that entries with low credit balances in the cache are reconciled frequently means that overdrawn accounts are rare since decrements due to parallel service sessions on these accounts will quickly be registered in the cache. Incorrectly denied service requests are also rare because top-ups on accounts with low credit balances are quickly registered in the cache.

## 7 Summary and Conclusions

The data obtained from running the simulations shows that the flow control algorithm was successful in protecting the SCP from overloading using a variety of cache sizes and also that the flow control algorithm successfully utilises the unused service request processing cycles of the SCP in order to effect reconciliation between the cache and the master data.

Selecting the best cache size is a balance of protecting the SCP against overload and maximising cache synchronisation with the master data on the SCP. A larger cache size is the best protection against SCP overload and an unlimited cache size in fact prevents it from occurring at all. The downside of an unlimited cache size is that there may be some revenue leakage due to discrepencies between the balances stored in the cache and the real balances stored in the master database in the SCP, although the use of a good prioritisation strategy can minimise this.

In the case of selecting the correct cache size, a number of factors could be considered.

- What is the normal load on the SCP? If the normal load on the SCP is such that the SCP is operating well within its processing capacity then a smaller cache size might be considered. As has been seen, a cache size of 20% is not sufficient to protect the SCP from overload during a usage spike of 150% of it's capacity. However if subscribers only place the SCP under 25% of it's load under normal circumstances then such a small cache size might be considered.
- How many subscribers are there? If there are a large number of subscribers then replication of a significant portion of the master database in a cache may not be possible due to memory constraints in the PSP.

In relation to the prioritisation strategy for selecting cache entries for reconciliation two different types of scenario were examined. In the first scenario, balances stored in the master database on the SCP were not subject to change in parallel service sessions. In this scenario a prioritisation strategy that promoted the reconciliation of entries with the greatest variance between the cache balance and the last known SCP balance produced the best results in keeping overall variance between the cached credit balances and the master credit balances to a minimum. In the second scenario, the effect of parallel service sessions was introduced and the results measured. In this scenario a strategy which prioritised the reconciliation of cache entries with the lowest credit balances first was shown to result in the fewest overdrawn balances and also the fewest incorrectly denied service requests. For a telecommunications operator, both incorrectly denied service requests and incorrectly overdrawn credit balances result in a loss of revenue and for this reason this prioritisation strategy is the most attractive.

Different strategies for removing entries from the cache where cache size is limited were not examined in any great detail. In order to focus on the performance of the algorithm in reducing load on the SCP and on the perfomance of the different prioritisation strategis, one removal strategy which was sufficient for removing entries from the cache when necessary was implemented.

While implementing the model a possible problem was discovered with respect to the scheduling of reconcile requests. In cases where the total number of requests for new cache entries sent to the SCP during interval $(t, t+T)$ is greater than the estimate value $\hat{R}_{SCP}(t, t+T)$, the SCP will be sent more requests than it can handle. To mitigate against this an alternative approach to scheduling reconciliation requests is outlined in which the length of $\tau_{recon}$ is adapted during the control interval in order to reflect the number of cache entry creation requests that are actually being sent to the SCP.

In the adaptive approach we recalculate the value of $\tau_{recon}$ every time a cache entry creation request is sent to the SCP. Let the new value of $\tau_{recon}$ be denoted $\tau'_{recon}$. Let $t^*$ denote the time the recalculation is made and let $R_{SCP}(t, t+t^*)$ denote the number of cache entry creation requests (including the one that triggered this recalculation) that have arrived to date in the control interval. Finally, let $R_{recon}(t, t+t^*)$ denote the number of reconcilation requests that have been so far during the control interval. $\tau'_{recon}$ is then calculated as:

$$\tau'_{recon} = \frac{C_{SCP} - \Big(R_{SCP}(t, t+t^*) + R_{recon}(t, t+t^*) + \big(R_{SCP}(t, t+t^*)(T-t^*)/t^*\big)\Big)}{T - t^*} \tag{6}$$

Thus the next reconciliation request dispatch is re-scheduled for time $t + \tau'_{recon}$. This approach may provide a better response to changes in load during the control interval particularly where the control interval is in the order of tens of seconds and this is an area for further research.

Other factors outside the scope of the research described in this document could also contribute to a better solution. These include a more detailed investigation of strategies for removing entries from the cache, the application of knowledge of customer usage patterns or context to pre-fetch entries for the cache (for example pre-fetch the most frequent users based on their normal usage at a certain time of the day) and the interaction of these with existing reconcile prioritisation strategies. These factors provide scope for future research in this area.

# Nomenclature

| | |
|---|---|
| AAA | Authentication, Authorization and Accounting. The related tasks of authenticating that subscribers are who they claim to be, authorizing what services they are allowed to access and maintaining an account of their service usage. |
| IN | Intelligent Network. This refers to the fixed and mobile telecommunications network which is based on the SS7 protocol. |
| IP | Internet Protocol. A communication protocol used by the internet. |
| PSP | Prepaid Service Platform. A platform to handle the provision of prepaid services. |
| QoS | Quality of Service. |
| SCP | Service Control Point. This is a high-performance server that contains service control logic. Calls are routed to the SCP by the SSP based on the service access code. |
| SDP | Service Data Point. High-performance database containg subscriber records and other data needed to use a service. The SDP is often co-located with the SCP. |
| SS7 | Signalling System No. 7. An out-of-band signalling protocol that is used by most of the worlds telecommunications networks. |
| SSP | Service Switching Point. Service switching points are located in the local telephone exchanges and route calls to different services in the SCP based on the service access code. For example freephone numbers may begin with 1800. |

## References

1. A.W Berger. Comparison of call gapping and percent blocking for overload control in distributed switching systems and telecommunications networks. *IEEE Transactions on Communications*, 39(4):574–580, April 1991.
2. Wei-Zu Yang; Fang-Sun Lu; Ming-Feng Chang. Performance modeling of integrated mobile prepaid services. *IEEE Transactions on Vehicular Technology*, 56, Issue 2:899–906, March 2007.
3. Peter J. Denning. The locality principle. *Communications of the ACM*, 48(7):19–24, July 2005.
4. Liu Yanming; Ma Yuxiang; Yi Kechu; Ke Guofu. Practical load control algorithm for multi-scps in. In *Proceedings. ICII 2001 - Beijing. 2001 International Conferences on Info-tech and Info-net*, volume 2, pages 760–765, 2001.
5. A.; Lining Gao Hac. Congestion control in intelligent network. *IEEE International Conference on Computing and Communications Performance*, 1:279–283, 1998.
6. Saheban F Houck D.J., Meier-Hellstern K.S. and R.a Skoog. Failure and congestion propagation through signaling controls. In *Proceedings of the 14th International Teletraffic Congress*, pages 367–376, 1994.

7. G.; Vahid S.; Baker N. Khan, F.; Foster. Comparison of charging methods in 3g mobile networks (consumer behaviour versus tariff models). In *Third International Conference on 3G Mobile Communication Technologies*, number 489, pages 382–387, 8-10 May 2002.
8. P.; Stiller B. Kurtansky, P.; Reichl. The evaluation of the efficient prepaid scheme tica for all-ip networks and internet services. In *10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 284–293, May 21 2007.
9. J.S Lee, Y.; Song. Overload control of scp in advanced intelligent network with fairness and priority. In *Proceedings. Sixth International Conference on Computer Communications and Networks*, pages 85–90, 22-25 September 1997.
10. Jussara Almeida Li Fan, Pei Cao and Andrei Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions On Networking*, 8(8):281–293, June 2000.
11. George Liu and Gerald Q. Maguire. Prefetching techniques in distributed systems. Technical report, KTH Royal Institute of Technology, Sweden, October 1994.
12. Ghaleb Abdulla Stephen Williams Edward A. Fox Marc Abrams, Charles R. Standridge. Caching proxies: Limitations and potentials, July 1995.
13. Athanassia Alonistioti Lazaros Merakos Maria Koutsopoulou, Alexandros Kaloxylos and Katsuya Kawamura. Charging, accounting and billing management schemes in mobile telecommunication networks and the internet. *IEEE Communications Surveys & Tutorials*, 6(1):50–58, First Quarter 2004.
14. J.A.G. Marti, A.B.; Ibanez. A prepaid service schema for beyond 3g networks. In *, 2006. IE 06. 2nd IET International Conference on Intelligent Environments*, volume 1, pages 391–399, 5-6 July 2006.
15. M.Kihl and C. Nyberg. Investigation of overload control algorithms for scps in the intelligent network. *IEE Proceedings-Communications*, 144(6):419 – 423, December 1997.
16. Bruce S. Northcote and Donald E. Smith. Service control point overload rules to protect intelligent network services. *IEEE/ACM TRANSACTIONS ON NETWORKING,*, 6(1):71–81, 1998.
17. OPNET. Network simulation, 2009.
18. Christian Spanner Pablo Rodriguez and Ernst W. Biersack. Analysis of web caching architectures: Hierarchical and distributed caching. *IEEE/ACM Transactions on Networking*, 9(4):404–418, August 2001.
19. Thomas E. Heaven W. Jim Jordan Parthasarathy Guturu, Jatinder Pal and Zhengya Zhu. Message replication and consumer database synchronization algorithms and system for highly available high performance intelligent networks. *IEEE Transactions on Consumer Electronics*, 53(2):375–383, May 2007.
20. Oxford University Press. *Compact Oxford English Dictionary of Current English.* Oxford University Press, 3 edition, 2005.
21. Donald E. Smith. Ensuring robust call throughput and fairness for scp overload controls. *IEEE/ACM Transactions on Networking*, 3(5):538–548, October 1995.
22. G. C. Stierhoff and A. G. Davis. A history of the ibm systems journal. *IEEE Annals of the History of Computing*, 20(1):29–35, January 1998.