CAP Limits in Telecom Subscriber Database Design

Javier Arauz
Ericsson
Via de los Poblados 13
Madrid, Spain, 28033
34913392997
jesus.javier.arauz@ericsson.com

ABSTRACT

While the notion of a Distributed DBMS has been familiar to the IT industry for several decades, within telecom networks the subscriber data management based on DDBMS technology is a novel addition to a service provider's infrastructure.

Service providers are used to telecom networks that are efficient, reliable and easy to maintain and operate, in part thanks to the node model used in designing such networks. A DDBMS spanning a large geographical area however incurs into distributed systems issues not previously seen in telecom networks.

Identifying and delivering the right set of trade-offs that satisfies the service providers' needs while staying within the known physical bounds of a distributed system is therefore crucial if DDBMS are to conquer the subscriber management space within telecom networks.

1. INTRODUCTION

Telecom networks have used databases to hold subscriber data since the advent of the Global System for Mobile (GSM) standard [1]. In GSM networks, the Network Function (NF) known as Home Location Register (HLR) performs the task of managing the data of the subscribers of the network. The HLR, in addition to holding subscriber data, cooperates in many network procedures with multiple NFs in order to provide the services the users demand from the network. Two of the most relevant procedures are authentication/authorization and location management, but there are many others.

In modern IP Multimedia Subsystem (IMS) [2] and 4G System Architecture Evolution/Long-Term Evolution (SAE/LTE) [3] networks, the subscriber management NF is realized by the Home Subscriber Server (HSS), an extension of the HLR supporting the additional or improved services these networks provide to their users. As opposed to HLR, HSS is capable of supporting fixed networks in addition to mobile cellular networks.

Both HLR and HSS manage massive amounts of data about all the subscribers in a telecom network. Following the traditional node-based design of a telecom network, every NF is realized by multiple independent nodes, such that failure of a single node does not affect the rest of the network. Each node holds one partition of the whole subscriber data space, thus when one node fails the subscribers whose data are held in the failing node lose access to the network. This node-based design has allowed telecom networks to keep pace with technological advances and new services while retaining the reliability properties users of those networks have learned to expect.

But not everything in a node-based network is good news for the service provider that operates it. According to the description above subscriber data are spread across vertical silos, each silo owning just one partition of the subscriber space. These silos pose two major challenges to the service provider:

1. Managing all these silos implies increased operating costs and complexity. Data duplication, where pieces of data with the same meaning are stored across different silos (e.g. HLR and HSS), is of special concern since it requires coordinated data management across silos.
2. Performing business intelligence and operative research over subscriber data becomes a formidable task, since there's no standardized way of fetching subscriber data from the silos

To overcome the above challenges, 3GPP (the standardization body behind the highly-successful 3G and 4G standards) [4] came up with a User Data Consolidation (UDC) architecture [5]. In this architecture, all subscriber data are consolidated in a single subscriber database – the User Data Repository (UDR), while participation in network procedures is left for application front-ends (FE)[1]. This database is mandated to support an LDAP-based interface [6] to read/write subscriber data. The structure and semantics of subscriber data are not detailed by the UDC specifications.

Since the UDR has to be accessed by every application FE in a service provider network, in most cases it cannot be a centralized database located at a single place. Instead, it becomes a distributed database, with data spread across multiple storage sites and Points of Access (PoA) wherever FEs are located. In between the storage sites and the PoA a distributed middleware creates the illusion that one single data space exists. The distributed nature of the UDR combined with the particularities of a telecom network is what poses the issues that are described in this paper.

Proceedings of the VLDB Endowment, Vol. 7, No. 13

[1] Examples of application front-ends are the HLR-FE and the HSS-FE, named after their non-DLA counterparts. These front-ends cooperate in the same network procedures as their non-DLA versions, but when a FE needs subscriber data to execute its part of a network procedure it reads those data from the UDR.

Nailing down those issues applying the proper choice of database technologies and associated trade-offs is crucial for UDC to become the next-generation target architecture in present and future telecom networks.

2. DESCRIPTION OF THE PROBLEM

2.1 The 3GPP telecom network

As has been mentioned in the introduction, traditionally telecom networks have been built following a node-based design, where a NF, e.g. HSS, is realized by multiple, independent instances. These instances are spread across the geographic area of coverage of a service provider, such that there is always a NF instance close to a user of the network service. Telecom networks are designed this way for resiliency reasons: when one instance of a NF fails, only the users making use of that instance are affected. While it can be argued this is a property also of well-designed distributed systems, as long as some interaction between the parts of a system exist it cannot be guaranteed that a severe problem in one of the parts will not bring the whole system down. Spreading NF instances geographically is done for efficiency reasons: traditionally, long-distance backhaul of IP traffic has been both limited in bandwidth and expensive, hence deploying NF instances close to the users decreases the need of backhauling control plane IP traffic to a central control plane processing data center[2].

The following figure illustrates the traditional building practice for a telecom network operated by a multi-national service provider.

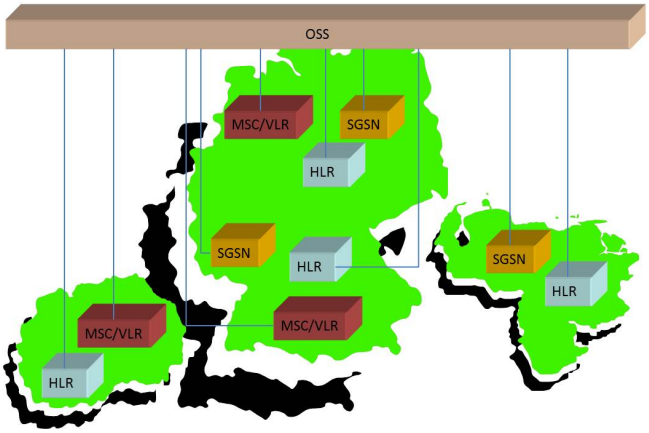

Figure 1. Traditional building practice of a multi-national telecom network.

2.2 Telecom networks and UDC

In 3GPP UDC networks, subscriber management NFs become stateless application front-ends, while actual subscriber data management takes place at the UDR NF. This is a major departure from the pre-UDC building practice, since due to distribution of subscriber data across the network in a UDC network there's a high chance (up to 100% depending on UDR NF architecture) that for every network procedure, control plane IP traffic has to be back-hauled to a remote location. This turns the IP back-bone network into an integral part of the telecom network, and leads the service provider to put more effort in IP network resiliency and latency than in NF reliability and deployment in order to keep the service levels unchanged with respect to a pre-UDC network.

This increased investment in IP network resiliency and latency is compensated by operating expense savings thanks to higher flexibility across the telecom network –i.e. application FEs become state-less processing nodes hence any FE instance can serve any user, enabling statistical multiplexing and resource sharing- and subscriber data consolidation into one single NF – the UDR, removing the need for data redundancy control and coordinated data management-.

Additionally, the prospects of improved operational efficiency and business management enabled by data mining over the subscriber data stored in the UDR is propelling service providers to move to a DLA telecom network. This aspect however remains to be realized mainly due to business and regulatory issues.

2.3 Architectural framework of a UDR NF realization

The UDR NF is the UDC network's Single Point of Access (SPoA) for subscriber data. Since a telecom network may grow up to hundreds of millions of subscribers and span whole continents while keeping a high Quality of Service (QoS) level any realization of the UDR NF must fulfill a number of requirements:

1. It must provide enough storage for the service provider's full subscriber base
2. It must be able to accommodate a growing subscriber base with little operating expense associated
3. It must be resilient, on average any given subscriber's data must be available 99.999% of the time
4. It must be fast, with a target average response time of 10ms (excluding network delays) for index-based single subscriber queries
5. It must be cheap and easy to operate, which implies it must provide ACID guarantees

The above can be summarized in that the UDR NF must be *fast*, *resilient*, *ACID*, *scalable* and *huge*, abbreviated *FRASH*. Those requirements led us to an architectural framework having the following characteristics:

- To be *fast*, data are stored in volatile media (RAM memory)
- To be *resilient*, all the elements in the architecture are redundant and geographically spread
- To be *ACID*, storage provides support to grouping operations into transactions that execute atomically
- To be *scalable*, storage is split into elements of a limited size so growth can be tackled in small steps
- To be *huge*, the architecture is able to accommodate a high number of storage elements

[2] This aspect is less relevant every day with the advent of massive bandwidth optical networks. Currently service providers consider that centralizing NF instances at a single data center is superior in terms of operating costs to the traditional geographically-spread deployments.

The diagram in figure 2 shows a representation of a multi-national UDR NF following the just described architectural framework.

In the UDR NF shown, every SE holds a primary copy of one partition and one or two secondary copies of other partitions. For instance, if subscriber data space were spread across three partitions, the top-most SE in figure 2 might hold the primary copy of partition 1 and secondary copies of partitions 2 and 3; the middle SE might hold the primary copy of partition 2 and secondary copies of partitions 1 and 3; and the bottom SE should then hold the primary copy of partition 3 and secondary copies of partitions 1 and 2. Given that data distribution policy, the UDR from figure 2 can continue providing service for 100% of the subscriber base as long as one PoA and one SE are reachable.

Partitions are further split into sub-partitions in order to address the scalability properties of the UDR. A single subscriber data partition typically amounts to *circa* 200 GB. The 200 GB size comes imposed by the available RAM memory in one SE's HW platform, so it can be expected to double with every new HW generation.

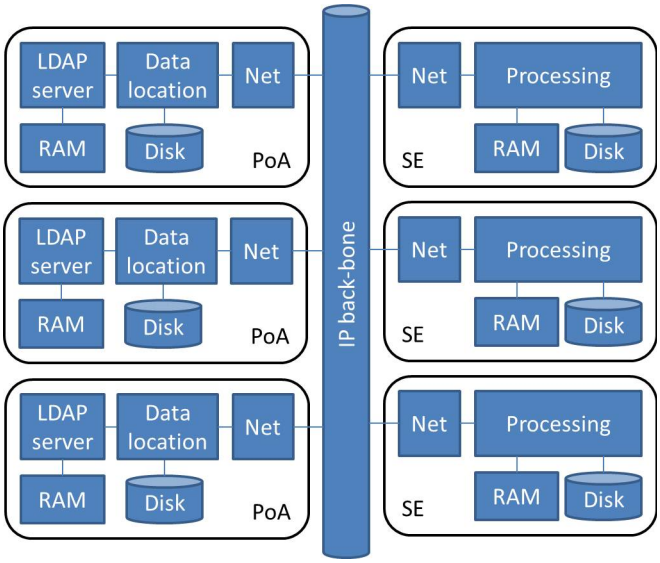

Figure 2. Architecture of a UDR NF realization

.

2.4 Operation of the Telecom Network

A UDC Telecom Network is operated exactly the same way as any other telecom network, i.e. following the Telecom Management Forum (TMF) specifications [7] with the help of an Operations Support System, OSS. All Operation and Maintenance (OaM) on the network is performed through the OSS, which offers the operator a consolidated view of all the nodes in the network.

With respect to subscriber management, a UDC network is very different from its predecessors. In pre-UDC networks subscriber management is performed directly on the nodes involved. For instance, in order to create a subscription in the network, also known as "provisioning" a subscription, many write operations need to be issued on multiple nodes:

- Subscription data are created on one instance of the subscriber data management NF, e.g. the HSS
- Data location information is created in all instances of signaling routing NF, e.g. the Subscription Location Function (SLF). This information consists of binary tuples containing the identities associated to the subscription, e.g. Mobile Subscriber Integrated Services Digital Network (MSISDN) number and IMPU (IMS Public Identity), and the address of the NF instance containing the data for that subscription

Provisioning operations are executed with the support of a Provisioning System, PS. All the operations associated with a single provisioning procedure need to be handled as a transaction. Since NF instances do not provide support for transactional operations this turns into very complex PS logic that needs to take care of executing provisioning operations transactionally across multiple nodes.

In a UDC network however, the PS has one single place that needs to be written (the UDR), which provides support for handling a provisioning procedure as a transaction. This allows simplification of the PS logic to a large extent, and solves corner cases that could not be solved in pre-UDC networks and that normally end up requiring manual intervention on the nodes to restore the network to a consistent state.

Figures 3 and 4 illustrate the difference between pre-UDC networks and UDC networks respectively with regards to provisioning. Notice that in figure 4, every UDR element is a part of the whole distributed UDR NF, being all parts inter-connected through the multi-national IP back-bone network.

2.5 Conflicting Demands on Data Management in UDC Networks

Unfortunately the *FRASH* requirements on the UDR NF are conflicting, as the reader savvy in DDBMS technology will have noticed. Conflicts arise when trying to maximize two or more of the *FRASH* characteristics. For instance, *fast* data access enabled by storage in volatile media very often conflicts with *resilient* data access since on unplanned events contents of volatile media may vanish, thus rendering data of a number of subscribers unavailable. As another example, *huge* and *scalable* data storage by means of a high number of limited-size storage elements may go against *fast* data access since data location management becomes more complicated.

The best known theoretical framework to understand and analyze the relationships between conflicting characteristics in a distributed system is the CAP (Consistency-Availability-Partition tolerance) theorem [8]. In our discussion of the UDR NF, Consistency is represented by *ACID* properties; while Availability and Partition-tolerance is represented by the *resilience* property. The other properties –*fast*, *scalable*, *huge*- are not linked by the CAP theorem, although they are in turn constrained by the *acid* and *resilient* properties. For instance, the larger the data set stored by the UDR, the harder is to maintain a fast and scalable data location stage, ACID properties of transactions spanning many storage elements and 99.999% available data in the presence of multiple storage element failures.

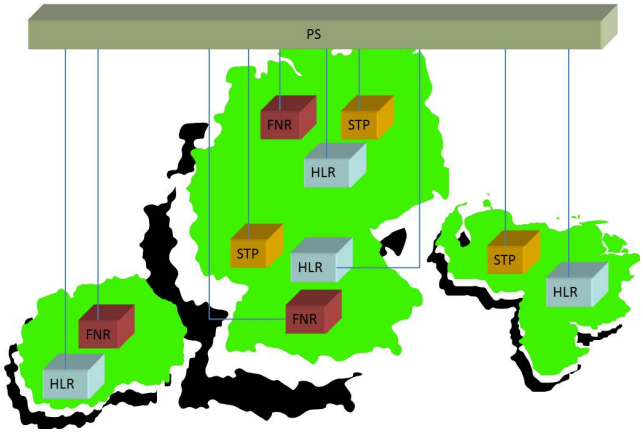

Figure 3. Provisioning in pre-UDC networks

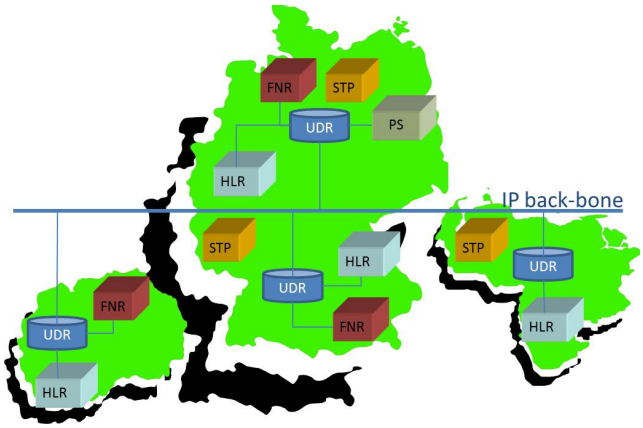

Figure 4. Provisioning in UDC networks

Figure 5 shows the relationships between the *FRASH* characteristics of the UDR NF. The grayed oval in the figure represents the scope of the CAP theorem. An arrow between any two characteristics represents a restriction, such that increasing one of them forces to decrease the other. In the coming sections we will illustrate where along those arrows the UDR NF realization described in this paper sits.

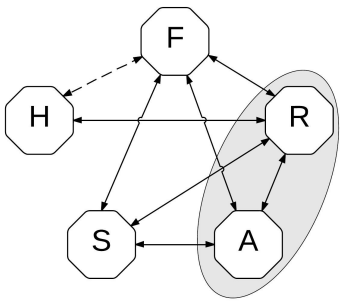

Figure 5. Relationships between the UDR NF *FRASH* characteristics

With the advent of grid databases like HBase [9] and a myriad of NoSQL[3] databases like Cassandra [10] and MongoDB [11], new trade-offs not covered by the CAP theorem have arisen. A new term that covers this kind of databases is PACELC [12]. PACELC stands for "On a Partition be either Available or Consistent, Else favor either Latency or Consistency", thereby introducing latency in the equation. Scalability and size have not been included in any theoretical framework to the best of our knowledge.

The UDR NF is also covered by the PACELC term. The service demands when there is a partition in the IP network deserve special attention. On a partition, the system may decide to be either Available or Consistent (R or A in *FRASH*). In many applications of DDBMS technology it is acceptable to stop business for a few seconds or even minutes on unexpected events. That's not the case of the UDR; as we've already mentioned, on average data of any given subscriber must be available 99,999% of the time[4]. Hence if the observation period is one year, on a network partition lasting more than a few seconds the UDR NF must make adjustments so subscriber data can be accessed despite the network problems. In the following section we'll see how this C vs. A&P conflict materializes under a possible set of design choices for a UDR NF realization.

3. DESIGN CHOICES

Given the architectural framework introduced in section 2.3, in this section we present a possible set of additional design choices for a first realization of a UDR NF. We believe this approach is illustrative of the iterative process typical of the introduction of a technology in a new realm where it hasn't been used before.

3.1 Resilience

Resilience is not negotiable: data availability must remain at or above 99.999% at all times. This has been the trademark of telecom networks for ages and the world is still not ready to trade reliability in communication for other benefits[5].

RAM-based data storage is established in the architectural framework as the premise to provide the speed of access required to keep the network service responsive. However RAM memory is inherently unreliable, so some adjustments to improve resiliency of data are necessary. Since by virtue of the F-R arrow from figure 5 speed of access would be decreased, the adjustments imply a trade-off between these two factors.

Two design decisions make sense in this scope:

1. To protect data against individual storage element failures, every storage element saves data in RAM to local persistent storage (i.e. SATA or SAS hard disk) on a periodic basis

[3] HBase can also be classified as a NoSQL database, although it uses the more traditional concept of tuples stored in tables where the others use the concept of documents stored in a storage pool

[4] This is an average, i.e. if one subscriber's data is not available at all during the observation period but data of 99,999 other subscribers has been available 100% of the time then the average availability for the 100,000 subscribers is 99,999%

[5] In most available disaster movies, during the catastrophe's algid moments when all computer systems are down the telephone system keeps working

2. To protect data against unforeseen events, every piece of data is duplicated in two or more geographically-disperse locations

The storage elements (SE) should guarantee data accessibility on failures of internal parts of the element by means of intra-element redundancy; hence what remains is availability in the presence of complete SE failures or catastrophes affecting multiple elements.

These decisions are a compromise between F and R since they enable decent data accessibility with very little decrease in speed of access; the storage engine is slightly slowed down since it is deprived of some computing resources needed for storage to disk and replication[6].

But the most relevant trade-off in a UDR system materializes due to the second decision above. Introducing data duplication in the system implies some kind of replication between copies, which takes us into the realm of the CAP theorem and the C vs. A&P trade-offs. The number of possible scenarios and the decisions the system has to take in each scenario would take a full paper by themselves. We provide some details once we explain how the multiple copies of every piece of data can be kept in sync, in sections 3.2 and 3.3.

3.2 ACID

Consistency in subscriber data is very important for a service provider; after purchasing a cell phone subscription from a service provider, we all expect that after sliding the SIM card into the phone and powering it on service is up immediately. Also if you set up a pay-call barring for the line, you wouldn't be very happy if you find your kids speaking on it to a hi-toll number.

The above issues may arise if the UDR NF does not fulfill ACID properties for provisioning operations. A partially executed provisioning transaction may render a subscription unusable. An alteration in the order of concurrent transactions from an application front-end may revert changes to the subscription status without the user noticing it.

Compliance to the ACID rules may have a huge impact in speed of access to the data. Thus a compromise along the F-A link from figure 5 is necessary. This is one of the hardest choices since usually a service provider wouldn't compromise on any of F or A. Our telecom network expertise however advocates for not jeopardizing F as the lesser evil: one angry customer switching carriers is less evil than a constant churn due to poor network performance. Hence the following decisions lend the compromise more towards the F end of the link:

1. ACID properties are guaranteed for transactions running on the same storage element only, i.e. SE are transactional. This prevents from having to run consensus protocols like e.g. 2-Phase Commit (2PC) across geographically disperse locations, which may be expensive.
2. The level of isolation between concurrent transactions on a storage element is set to READ_COMMITED [13], to prevent locking from delaying reads on subscription data (reads are mostly issued by application front-ends).

Obviously, for transactions running over multiple storage elements no ACID guarantees are provided, and the level of isolation afforded to those is READ_UNCOMMITTED [13].

To minimize potential consistency problems, clients of the UDR should limit transactions spanning multiple SEs to a minimum. Application FEs always work against one single storage element (the one containing the subscription involved in the corresponding network procedure), and the PS should attempt to group operations into transactions addressed to as few SE as possible. Nevertheless, the PS still needs to contain logic dealing with unexpected outcomes due to the lack of transactionality between SEs.

The second decision from section 3.1 introduces an additional facet to ACID properties. It would be useless to invest computing resources and execution time in guaranteeing atomic transactions if the multiple copies of a data piece are not kept in sync. In database replication setups, concurrent write operations on different copies of data may lead to different outcomes at each copy. Individual copy failures are another source of inconsistency in these scenarios.

To maximize consistency hence ACID properties across copies, copies are not all equal. At every point in time for each piece of data there is one copy handling all writes to that data. That copy is referred to as the *master* copy. The master copy replicates writes to the other copies of the data, which we will refer to as the *slave* copies. The replication mechanism is such that it guarantees the serialization order of writes replicated to any slave copy is exactly the same as that imposed by the master copy. Since no writes can be applied to a slave copy, the serialization order guarantee above is enough to maintain all copies in sync with respect to the actual data they hold.

What happens when contact between replicas is lost due to network partitions? Since writes can only be applied at a master replica, as long as a transaction can access the master replica it will be executed. This means if the client issuing a transaction is on the side of the partition that contains the master replica for the piece of data the transaction affects the outcome of the transaction will be successful, otherwise it will be failure. The failure cases decrease the overall availability of data, hence sliding the trade-off point along the R-A link towards the A end (or, in CAP terms, we favor Consistency over Availability on a partition).

3.3 Fast

A fundamental property of any telecom network is its *responsiveness*. We all have felt frustrated when it takes more than a few seconds for our cell phone to register in the mobile network. This not to mention the disappointment caused by an outgoing call attempt taking more than one second to start.

Mostly every action a user performs with her cell phone involves some network procedure, and every network procedure involves one or more accesses by the application front-end involved (e.g. HLR-FE) to the UDR NF. Hence it is crucial that processing transactions from application front-ends is as fast as possible, in order to not jeopardize responsiveness of the whole network.

Transactions from the PS are not so fatally affected by excessive delays. However, excessive delay in processing provisioning transactions may have other types of impacts. The average service provider keeps a continuous flow of provisioning

[6] It is possible to configure storage elements to dump transactions to disk before committing for 100% guaranteed durability, but that would slow down storage elements too much (the F-R trade-off point would slide too much towards the R end).

operations going at any one time. There are periods, dubbed *low traffic hours*, during which this flow falls down to a minimum. However, out of those periods long delays in processing provisioning transactions might cause a back-log of operations to grow at the PS. If this back-log overflows for some reason, dropping operations in the way, outcome would be fatal. Some service providers perform batch provisioning, which consists of issuing a huge batch of provisioning operations during a relatively short period of time. Batch provisioning is more likely to cause back-logs in the presence of excessive delays in processing provisioning transactions.

Speed of access is guaranteed from the architectural framework by RAM-based data storage. However, as in any distributed system, network delays play a key role in the response time of the system so additional architectural measures need to be taken in order to keep average speed of access as high as possible.

3.3.1 General for all transactions

In general for any transaction the architecture provides the following mechanisms to enhance speed of access to data:

1. Every point of access to the UDR is capable of resolving data location locally to the PoA, without incurring in long packet exchanges with remote locations over the IP network.
2. Replication of writes from the master to the slave copies is performed asynchronously, so execution of a transaction does not have to wait until the corresponding write(s) have been propagated to the slave replica(s). Since the slave replicas are distant –in geographical terms- from the master replica in order to cater for natural disasters, waiting for a write to be propagated to a slave replica most probably implies a slow exchange of IP packets over the IP back-bone network.

The first decision above favors speed of access (F) despite scalability (S) and size (H); for scalability, on scale-out new data location stages have to be deployed and sync'd with those already in place, which makes the procedure longer and more failure-prone. For size, storage of the identity-location maps deprives storage elements from memory they could use to store more data. Data location uses identity-location maps since the UDR must support multiple indexes (one index per subscriber identity, i.e. MSISDN, IMSI, IMPU etc.) and must support also the selective placement of subscriber data (due to e.g. regulatory or security reasons, where data for subscribers belonging to a country or organization must be located at a predetermined place). Using identity-location maps however has proven to be problematic due to the state-full characteristics of the data location stage; a discussion of possible alternatives comes later in this paper (see section 3.5).

The second decision above moves the trade-off point away from A and closer to F along the F-A link. Asynchronous replication does not guarantee that, in case of failures, all transactions committed at the master are successfully replicated to the slave(s). Hence, a transaction committed on the master with ACID guarantees might not be durable if a severe failure prevents the transaction from being replicated to at least one slave.

3.3.2 Transactions from application front-ends

The following measures guarantee fast access to data from application front-ends:

1. There is always a point of access to the UDR close –in network terms- to any one application front-end, as long as the cost of doing so justifies it. This enables fast IP packet exchanges between application front-ends and the UDR, improving fault detection and enabling efficient processing of erroneous transactions.
2. Read operations on slave copies are allowed. This may be useful if a slave copy of the data being read is co-located with the PoA receiving the read request from the application front-end. If that's the case all IP packet exchanges take place over a fast local network, as opposed to the slower IP back-bone.

The second decision above moves the F-A trade-off even further towards the F end: since asynchronous replication does not guarantee real-time sync between replicas, there's a certain chance that a read operation on a slave replica gets stale data, decreasing the consistency of read operations.

3.3.3 Transactions from PS

For data access from PS we've taken the following measures:

1. An instance of the PS is always co-located with a UDR PoA. This is just not feasible for application front-ends, since there are too many of them, but in a typical telecom network there are just one or two PS instances.
2. Read operations on slave copies are **disallowed**.

The second measure above may seem out of place here, since it moves the speed compromise *away* from the F end of the F-A link. But that's not actually the case. It is still an architectural decision related with speed of access, hence here's where it must lie.

The reason for compromising on F in favor of A for transactions from the PS lies in the fact that provisioning operations must be executed in atomic transactions, or at least as atomic as the architecture allows it. Since the architecture does not guarantee ACID properties on transactions spanning multiple storage elements, it is not possible to read from a slave replica and write on the master replica within one atomic transaction. Combining that with the eventual consistency afforded by asynchronous replication, the chance of the PS reading stale data is too high.

This all makes sense, since it is a consequence the previous decisions favoring F over A in general for all transactions (see section 3.3.1); if we now want to move the trade-off point closer to A for the PS type of transactions, we need to sacrifice some of F to achieve it.

3.4 Scalable

Scalability is a characteristic most service providers demand. By "scalability" we don't mean just the ability to scale, which is nowadays taken for granted in DDBMS technology, but the ability to do it easy and cheap. Most service providers have embarked in a journey to decrease operating expenses of their networks, therefore they appreciate the ability of the UDR NF to seamlessly scale with transaction load, subscriber base, or both.

When talking about scalability we can refer to more or less "local" scalability (the ability to increase the capacity of every constituent element of a system), also known as "scale-up", and "global" scalability (the ability to add new constituent elements to a system), also known as "scale-out". Scale-up is normally bound by the physical limits of the execution platform where the UDR runs, whereas scale-out is usually bound by practical limits in the UDR NF architecture.

3.4.1 Scale-up

By default, the execution platform of the UDR NF shall be a blade cluster. This allows adding more blades to the cluster when additional capacity is needed. To maintain high availability figures, the cluster should be compliant to the Service Availability Forum (SAF) specifications [14] so it provides Fault Tolerance and High Availability to the UDR processes.

A number of storage elements run on every blade cluster. Every SE is composed of two to four blades to provide for internal redundancy within the SE and shares nothing with any other local or remote SE .

Additionally, the UDR NF runs a distributed, state-less LDAP server providing the northbound interface to clients of the UDR. The LDAP server processes may be deployed to blades devoted to LDAP processing only, or they can share blades with SE. Since LDAP server processes are processor-hungry whereas SE processes are RAM-hungry, combining both kinds of processes on the same blade offers the best resource utilization chances.

The PoA to the UDR might be provided by a L4-capable IP balancer running in a few blades of the cluster. The balancer spreads LDAP traffic over all the LDAP servers available in the local blade cluster.

To scale up LDAP processing capacity, more LDAP servers can be deployed to the blade cluster. LDAP processing capacity of a cluster is thus bound by the number of blades that can be installed in a cluster and the capacity left in those blades by the SE and any platform processes running in the cluster. The IP balancer realizing the PoA to the UDR automatically detects new LDAP server instances deployed to the blade cluster so growth in LDAP processing capacity is automatic.

To scale up data storage capacity, more SE can be deployed to the blade cluster. Since data are stored in RAM memory, data storage capacity is bound by the number of blades that can be installed in a cluster and the amount of RAM left by the LDAP servers and any platform processes running in the cluster.

3.4.2 Scale-out

Scale-out is achieved by deploying additional blade cluster instances, containing LDAP servers and SE as needed. The SE deployed with the additional blade cluster have the same restrictions as those deployed for scale-up.

In every new blade cluster deployed, a data location stage instance is created automatically to increase F. This distribution stage instance syncs its identity-location maps with peer instances in other blade clusters without requiring any additional procedure, thus realizing the first decision from section 3.3.1; however, this synchronization takes some time, during which operations issued on the PoA realized by the new blade cluster cannot be handled. Therefore data availability (R) is affected by the data location sync mechanism introduced to facilitate S. Possible alternatives to identity-location maps are discussed later in this paper (see section 3.5).

3.5 Huge

In order to accommodate data for a huge number of subscriptions, a high number of storage elements needs to be supported by the UDR. The architecture must not set any unreasonable bound on the number of storage elements it can handle, and must provide an efficient mechanism to resolve the key-to-location mapping in the data location stage.

When running on a state-of-the-art execution platform, tests show that a 2-blade SE can hold up to 2·10E+06 subscribers with the average profile so that means that assuming a limit of 16 SE per blade cluster a single blade cluster instance should provide a capacity of 32·10E+06 subscribers (enough for a small country), while assuming a limit of 256 SE per UDR system the total UDR NF capacity sits around 512 million of subscribers. That's more than the population of the USA and roughly half the population in mainland China, and certainly more than any single service provider needs for the time being[7].

Huge data means huge transaction load. Again based on tests, a single LDAP server running on a state-of-the-art blade in a UDR system supports a load of 10E+06 indexed read/write queries of single subscriber per second. Assuming a limit of 32 LDAP servers per blade cluster, this throws a total transaction processing capability of 36·10E+06 LDAP read/write operations per blade cluster per second. If again we choose a limit of 256 blade clusters for a UDR NF, a single UDR NF supports a maximum of 9,216·10E+06 LDAP read/write operations per second.

One interesting conclusion from the figures above is that on state-of-the-art HW the UDR architecture can support around 18 LDAP read/write operations per subscriber per second. Typical mobile network procedures cause between 1 and 3 LDAP operations[8].

Being a distributed DBMS, the UDR contains sharding mechanisms to distribute data across locations. The more distributed data are the lower the chances that one LDAP read/write operation issued by an application front-end finds the subscriber data in a close location. The higher the chance that data have to be brought from a remote location over the IP back-bone, the higher the chance the operation fails (the IP back-bone is inherently less reliable than a local IP network). Hence, if we assume the number of different locations grows with the number of subscribers (which seems reasonable given the physical limits of population per square mile), the more subscriber data are held in the UDR the lower the availability of those data is. This is represented by the H-R link in figure 5.

To counter the decreased availability caused by data distribution, the UDR might allow the PS to specify in what SE it wants data of a subscription to be placed, i.e. selective location. This is useful in telecom networks since it is known that users stay within the home region of the subscription most of the time, so if the data of a subscriber can be pinned to a location close –in network terms- to the application front-ends in the home region of the subscription, chances of having to surf the IP back-bone to obtain that subscriber's data decrease enormously. Only when the user leaves her home region (she *roams*), the application front-end serving that user might have to go to a remote location to fetch the subscription data. This ability of the UDR system allows balancing the trade-off point along the H-R link. Notice

[7] The 16 SEs per blade cluster and the 256 SEs per NF are artificial limits used for the calculations. If trans-continental mergers end up taking place, like e.g. a large American service provider merging with a large European provider, the 512M figure might be challenged but the architecture should be able to accommodate more SEs.

[8] Network procedures in the 3GPP IP Multimedia Subsystem (IMS) are somewhat heavier. A single typical IMS network procedure may cause 5 or 6 LDAP read/write operations.

that, as already mentioned above, other criteria like regulatory or security reasons can invalidate this mechanism.

Also with the high number of data pieces distributed across many locations comes increased processing time in the data location stage. The data location stage has not been realized by means of hashing, which grows as *O(1)* in processing cost, since the UDR must support multiple indexes (one index per subscriber identity, i.e. MSISDN, IMSI, IMPU etc.) and must support also the selective placement of subscriber data described above. A state-full data location stage's processing cost typically grows as *O(logN)*, being N the number of subscribers in the UDR NF. Thus a high amount of data has a certain impact in processing time, represented by the H-F link in figure 5. Nevertheless, this impact is very small and can be neglected it in most calculations, hence the link has been represented with a dotted line to mean it's a rather weak link.

The state-full data location stage introduces a subtle trade-off in the F-R-S triangle. If the identity-location maps are provisioned, e.g. by PS, on scale-out the new data location stage needs to copy all the provisioned data from a peer stage in another cluster thus affecting R as was mentioned in section 3.4.2. However if the maps are built on the fly and cached instead, R is not affected but every cache miss implies locating the subscriber data by querying multiple or even all the SE in the system. Those data location queries may become a hurdle to scalability. In this paper we're assuming the maps are provisioned (given F, we favor S over R), which in the end might not be a consistent choice according to the descriptions of R and S provided above. Hence a change to a cached data location stage seems likely in the near future.

There are alternatives to identity-location maps for data location. One such alternative would be to use consistent hashing to index locations. To apply consistent hashing to the UDR, we need multiple replicas being each replica indexed by a different identity. The high number of current and future identities the UDR has to support might render this approach impractical.

3.6 Summary

The following figure depicts the effect of the design decisions described in this section on the *FRASH* relationships graph in figure 5. Red points are for provisioning transactions while blue points represent application front-end transactions. In spite of the figure, we argue that the UDR NF described in this paper is PA/EL for transactions coming from application front-ends but PC/EC for transactions coming from PS instances.

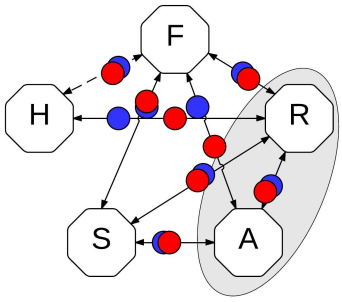

Figure 6. Trade-offs created by the design decisions on UDR NF *FRASH* characteristics

.

4. A CRITIQUE OF DESIGN DECISIONS

Comparing the different considerations and approaches outlined above, the following conclusions may be drawn.

4.1 Resilience vs. ACID

This trade-off falls within the scope of the CAP theorem. By default on a network partition the UDR NF realization described favors Consistency over Availability, for both provisioning and application FE transactions. This decision could raise concerns amongst service providers. On a network partition, while most transactions coming from application front-ends proceed successfully since those transactions are composed of mostly reads, transactions coming from a PS almost always fail since most provisioning transactions involve writes to subscriber data.

Provisioning transactions turn out to be more valuable (or costly, depending on how one looks at the problem) that one might initially estimate. More often each day, activation of a mobile network subscription takes place in an unattended fashion, triggered by the back-office system when a brand new user walks out of the phone shop and activates a device containing a SIM card associated to the subscription. If the activation fails because there's a network partition at that moment, two very bad things happen:

1. The new user gets disappointed with the service (and most probably walks back into the shop and complain to the desk staff)
2. The service provider needs to send someone to check what's happened, wait until network service is restored, and complete the activation manually.

The first event above will cause churn in the long run, hence the provider's income will decrease. But with immediate effect, the provider needs to cover the costs of the manual intervention required to complete the activation.

When using batched provisioning, a network glitch as short as 30 seconds may cause a batch that's been running for hours to fail. At the very best, if the batch is able to finish the provider needs to send someone to check what parts of the batch failed and apply those parts manually. This again incurs a cost for the service provider.

In general, service providers demand that on network partitions the UDR NF keeps taking writes stemming from provisioning transactions, or in other words, do not decrease Availability on a partition. Of course this means jeopardizing Consistency, as we'll see in the next section.

4.2 Fast vs. ACID

This trade-off is outside the CAP limits, but is considered by the PACELC taxonomy. By default, in the absence of a network partition the UDR NF realization described tends to be fast. For provisioning transactions it tries to compensate a bit towards the ACID end, but in the best case a compromise between both is reached. This makes a lot of sense, since from the architectural framework the system has been designed to be this way.

Again, this decision could generate service provider challenges. The reason is that on a failure of a storage element, durability of the latest transactions is not guaranteed. Given the relevance that provisioning transactions have for the service provider (see the previous section), this is an undesirable effect that most service

providers would like to see fixed to some extent, even if it implies losing in some other characteristic.

Becoming more ACID would imply being slower (F) and less available (R), ant to some extent less scalable (S). The reasons for this have already been exposed. What is not so clear is how much of F, R and S are service providers willing to sacrifice for guaranteed durability of provisioning transactions.

5. EVOLUTION

In the light of the critique from the previous section, a UDR NF realization as the one described above might need to make some changes to better cater for a service provider's needs.

First and foremost, some sort of multi-master operation would be very convenient so writes can be addressed to more than one single replica. This would allow the provisioning transactions to proceed on network partition events.

The CAP theorem [8] states that if we increase Availability on a partition incident we'll lose some Consistency. This is indeed the case when conflicting write transactions come from clients on different sides of the partition: since communication between the masters is impossible, they have no way of checking that the conflicting transactions are consistent with a single, common view of the data. Hence they'll apply the transactions on their respective views of that data, with such views diverging with every write they receive. Once the partition incident is over, a consistency restoration process must run across the whole UDR NF, trying to merge the different views into one single, consistent view.

Second, the service provider has to be allowed to tune the degree of durability it wants for provisioning transactions. The biggest hurdle here is not technical but human: the service provider's technical staff has to be made to understand that with increased durability (hence consistency) comes decreased speed of access. Actually, the latency penalty for achieving close to 100% guaranteed durability is so high that some unwary service providers might think it twice before going down that way.

One very elegant and powerful durability tuning solution has been implemented in Apache Cassandra [10]. In Cassandra, a client is able to specify the durability guarantees it wants on a per-transaction basis. Under the hood Cassandra uses a consensus protocol across an ensemble of replicas; the more replicas are involved in the transaction, the higher the durability guarantees.

In a UDR NF a similar approach is hardly affordable, since the latency increase would be too high. Instead, most probably the UDR NF should apply provisioning transactions in sequence to two replicas, committing the transaction only when both replicas report success. To avoid incurring the penalties of a consensus protocol, the UDR shall have to work in cooperation with the PS so when a transaction fails to commit, leaving just one of the replicas updated is acceptable. Most probably this limitation forces to restrict the dual-in-sequence replication of transactions to simple transactions that are idempotent or easy to roll-back.

6. CONCLUSIONS AND FUTURE WORK

In this paper we've characterized one possible realization of the 3GPP UDC architecture network function known as UDR. The UDR is a huge distributed DBMS with stringent requirements on latency, resilience, scalability and low cost of operation.

We've described an architectural framework guiding the design of the system, and the set of additional decisions made in order to reach optimum trade-offs within the physical limits of a huge distributed system. We've extended the CAP conjecture and the PACELC taxonomy with size and scalability characteristics to fully understand the limitations and following decisions.

Finally we've exposed weak points in the initial architectural framework and associated decisions, and possible corrections that might have to be made for compensating some not well tuned trade-offs.

Future challenges that any UDR NF realization will probably face include how to compensate for the lack of consistency the increased resiliency service providers demand will bring about. In that regard, one promising alternative to the master-slave replication approach described above lies on efficient distributed agreement protocols like e.g. Paxos [15] or similar solutions [16].

Also a very challenging aspect will be how to increase consistency for transactions coming from application front-ends without heavily impacting the latency those front-ends perceive.

7. REFERENCES

[1] Multiple authors, *GSM/GPRS Specification Series*. 3GPP web site (http://www.3gpp.org/DynaReport/03-series.htm).

[2] Thowle, T. *IP Multimedia Subsystem, Stage 2*. Technical Specification 3GPP TS 23.228, 3GPP web site (http://www.3gpp.org/DynaReport/23228.htm).

[3] 3GPP, UTRA-UTRAN Long Term Evolution (LTE) and 3GPP System Architecture Evolution (SAE). 3GPP web site (ftp://ftp.3gpp.org/Inbox/2008_web_files/LTA_Paper.pdf).

[4] http://www.3gpp.org

[5] Bartolome, M.C. *User Data Convergence, Stage 2.* Technical Specification 3GPP TS 23.335, 3GPP web site (http://www.3gpp.org/DynaReport/23335.htm).

[6] Wahl, M. Howes, T. Kille, S. *Lightweight Directory Access Protocol (v3)*. RFC 2251, Proposed Standard, IETF RFC Database (http://www.rfc-editor.org/info/rfc2251).

[7] http://www.tmforum.org

[8] Gilbert, S. Lynch, N. *Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services*. ACM SIGACT News Vol. 33 Issue 2, June 2002, Pages 51-59 (http://dl.acm.org/citation.cfm?id=564601).

[9] Apache HBase (http://hbase.apache.org)

[10] Apache Cassandra (http://cassandra.apache.org)

[11] MongoDB (http://www.mongodb.org)

[12] Abadi, D.J. *Consistency Tradeoffs in Modern Distributed Database System Design*. IEEE Computer Magazine, 2012 (http://cs-www.cs.yale.edu/homes/dna/papers/abadi-pacelc.pdf).

[13] ISO/IEC Joint Technical Committee 1, *Information Technology – Database Languages – SQL*, International Standard ISO/IEC 9075:1992 (withdrawn, http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=16663)

[14] Service Availability Forum, *Open Specifications for Service Availability*. SA Forum web site (http://www.myassociationvoice.com/page/16627~214723/Service-Availability-Forum-Open-Specifications-for-Service-Availability)

[15] Lamport, L. *Paxos Made Simple*. ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 101, December 2001) 51-58 (http://research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html#paxos-simple)

[16] Apache Zookeeper (http://zookeeper.apache.org)