Towards HSS as a Virtualized Service for 5G Networks

Hanieh Alipour[#], Fatna Belqasmi[*], Mohammad Abu-Lebdeh[#], Roch Glitho[#]
[#]Concordia University, Canada
[*]Zayed University, UAE

Abstract— **Home Subscriber Server (HSS) is the main database of the current generation's cellular communications systems. It contains subscriber-related information, such as the authentication information and the list of services to which each user is subscribed. The anticipated tremendous increase in the number of subscribers, services and devices (M2M) in 5G networks brings new challenges with regard to HSS provisioning. It calls for more scalability and elasticity regarding information storage, access and management. The current method for increasing the number of HSSs deployed is certainly not the most cost efficient solution. On the other hand, advanced virtualization techniques can aid in tackling the challenges while enabling a smooth migration to 5G. This paper proposes a new architecture for a scalable and elastic HSS using virtualization. The new architecture enables easy and rapid deployment of new HSS instances at a minimal cost, while increasing efficiency of the use of resources. The paper presents the architecture, demonstrates its use via a case scenario, describes the implemented proof of concept prototype and evaluates the performance results.**

Keywords—5G Networks; HSS; Database; Virtualization;

I. INTRODUCTION

Home Subscriber Server (HSS) is the main database of the current generation's cellular communications systems [1]. It stores subscriber profiles (e.g., authentication information), service profiles (e.g., when a given service is triggered), and roaming information. The constant increase in the number of subscribers is one of the main challenges faced by current and upcoming 5G networks. Indeed, the diversity of emerging applications and the coexistence of human-centric and machine-to-machine applications are expected to tremendously increase the number of subscribers, services, and devices. The number of connected devices is predicted to reach a total of 50 billion by 2020 [2]. This brings new challenges in terms of the scalability and elasticity of customer information storage, access and management.

Advanced virtualization techniques can aid in tackling the challenges while enabling a smooth migration to 5G. Virtualization is one of the key enabling technologies for elasticity and scalability [3]. It allows the abstraction and sharing of computer and network resources, which enables efficient resource usage and can result in important cost savings when it comes to new HSS deployment. This makes virtualization a compelling case for offering HSS in the 5G cellular communications systems. Offering HSS as a virtualized service (HSSasVS) in 5G will allow new service providers to get HSS virtual instances easily, rapidly and at a lower cost. They will not need to acquire or deploy any physical nodes but will still benefit from high availability, scalability, elasticity, and reduced operation and management costs. The amount of resources that are actually assigned to each service provider may vary dynamically, depending on the current number of customers.

To ease its use and enable smooth migration to 5G, the virtualized HSS should offer the same functionalities as the non-virtualized HSS and should be accessible via the exact same interface. It should also facilitate isolation, to allow each service provider to have access to its own users' data only. Furthermore, the HSSasVS architecture should enable easy and on-the-fly creation of new HSS instances, and should allow the using entities to discover the virtualized HSS on the fly. There are some attempts to virtualize HSS in 3G networks (e.g. [4]) and others to virtualize other 4G core components (e.g. [5]), but none of these solutions meets all of the stated challenges. Attempts to virtualize databases in general are also relevant (e.g. [6], [8] and [8]), as they may bring solutions to some of the issues related to data storage and management (e.g. performance and scalability). However, they don't offer a comprehensive solution by themselves (e.g. they don't support all functionalities of non-virtualized HSS).

This paper proposes and validates a new architecture for HSSaVS. As a proof of concept prototype, a scenario where three video telephony service providers are sharing the same HSS is implemented. The paper presents the architecture, illustrates it with the video telephony service provider's scenario, and describes the proof of concept prototype we have built. The rest of the paper is organized as follows. The next section describes the proposed architecture, followed by the illustrative scenario. The prototype is described in section IV, and the last section concludes the paper.

II. THE PROPOSED VIRTUALIZATION ARCHITECTURE

A. Overall Architecture

We decompose the HSS into two components: the Diameter server and the database. Diameter is the standard protocol used to access HSS [1]. The database component is further decoupled into two parts: the database management system and storage. The virtualized HSS architecture is therefore

978-3-901882-77-7 (c) 2015 IFIP

composed of three layers, as shown in Fig. 1. The Diameter layer (i.e., the top layer) is responsible for receiving and processing the Diameter requests. The database management layer (i.e., the middle layer) provides full Atomicity, Consistency, Isolation and Durability (ACID), compliance guarantees and ensures that all database transactions are processed reliably. The storage layer (i.e., the lower layer) represents a distributed storage that provides fault-tolerant shared disks. The architecture ensures isolation among service providers by applying a multi-tenancy approach [9].

B. *Funtional entities*

The Diameter layer consists of three entities: the virtual Diameter entity, the virtual Diameter routing agent, and the Diameter layer management entity. The virtual Diameter entity is the core entity, which receives and processes the Diameter requests. To ensure scalability, many instances of this entity may exist and others may be created and terminated on the fly to account for request load variations. The virtual Diameter routing agent is the first point of contact between the virtualized HSS and the external entities and it is responsible for forwarding the incoming requests to the appropriate virtual Diameter entity while ensuring load-balancing among entities. The Diameter layer management entity keeps track of the Diameter layer load and manages the layer's entities and their numbers. It receives load notifications from each of the active entities in the layer and decides whether to increase or decrease the number of the virtual Diameter entities and routing agents on the layer. The layer includes a redundant management entity

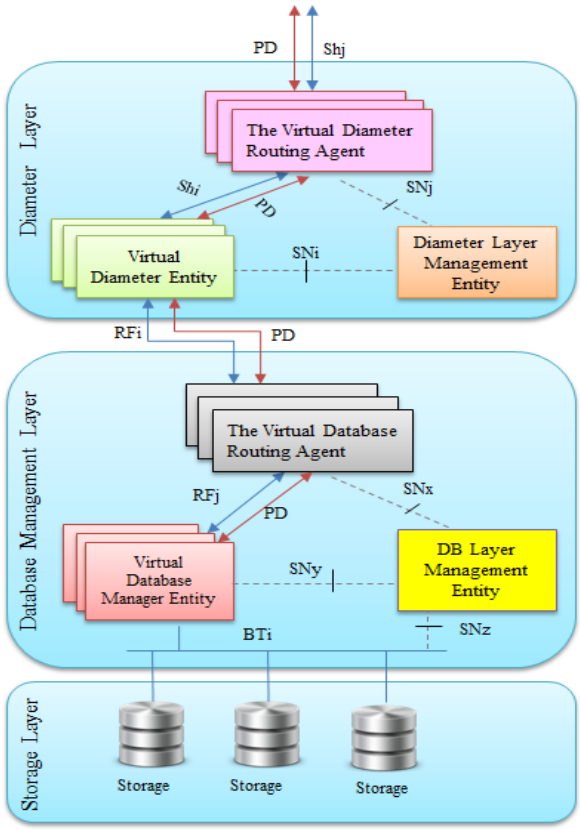

Fig. 1. The Virtualized HSS Architecture

to prevent a single point of failure.

The database management layer entities are the virtual database manager entity, the virtual database routing agent, and the database layer management entity. The virtual database manager entity is responsible for processing the database management requests (e.g., storing new or updating existing information). Many instances of this entity may run simultaneously and may offer access to various storage entities. The routing agent and the management entity have functionalities similar to the diameter layer entities with similar names. The database layer management entity also has a redundant entity to avoid a single point of failure.

The storage layer is made up of multiple storage entities, which host the actual data. This layer serves as distributed shared storage, with data being broken into several blocks and distributed among storages. The storages are mirrored to allow for recovery from failure. The virtual database manager entities have access to the same shared data (while respecting the multi-tenancy), so that if one entity fails, the requests are routed to one of the remaining entities.

C. *Communication Interfaces*

The interfaces between the virtual database routing agent and the virtual Diameter entity (i.e., RFi) and the virtual database manager entity (i.e., RFj) are REST-based. We selected REST because it is standard-based, lightweight and can support multiple data formats [10].

The publication and discovery interfaces (i.e., PD) between the virtual Diameter routing agent, the virtual Diameter entity and the external entities are based on the Service Location Protocol (SLP) [11]. The same applies to the PD interfaces between the virtual database routing agent, the virtual Diameter entity and the virtual database manager entity. SLP provides a mechanism for advertising and discovering network services. The discovery can be demand-driven (i.e., the service information is actively queried) or notification-driven (i.e., the entities are notified about the appearance or disappearance of the services in which they are interested).

The interfaces between the Diameter layer management entity, the virtual Diameter routing agent (i.e., SNj) and the virtual Diameter entity (i.e., SNi) are based on the Simple Network Management Protocol (SNMP). The interfaces between the database layer management entity, the virtual database routing agent (i.e., SNx), the virtual database manager entity (i.e., SNy) and the Storage (i.e., SNz) are also SNMP-based. SNMP is a widely used network management protocol that allows for the monitoring of network entities by a management host [12]. The communication between the virtual database manager entity and the storage (i.e., BTi) is based on the B-tree indexing technology. The B-tree technology stores data in tree form and provides data retrieval mechanisms accordingly [13]. As described in the 3GPP 4G architecture, the *Sh* communication interfaces are *Sh* Diameter interfaces.

D. *Procedures*

This section describes two of the main architectural procedures: the Diameter request processing and elasticity

procedures. The elasticity procedures are executed when the system discovers an unexpectedly high or low load on one of the layers and needs to instantiate new entities or deactivate some of the running ones.

Diameter request processing: When an external entity needs to communicate with the virtualized-HSS, it first discovers the list of available virtual diameter routing agents using the SLP. The agents reply to the discovery request with their current load sizes. The external entity identifies the agent with the lightest load and forwards the Diameter request to that agent. The external entity saves the address of the selected agent for future use. If the saved entity does not reply, it means that the entity has been deactivated, is currently handling the maximum allowed load, or is experiencing difficulties, in which cases the discovery process is re-executed.

The agent receiving the Diameter request is responsible for finding the appropriate virtual diameter entity, also using LSP, and transmitting the request to it. The same procedure is executed at the database management layer in order for the Diameter request to reach the virtual database management entity with the lightest load. The latter uses the B-tree technology to communicate with the storage layer and fulfill the request (i.e., store or retrieve data).

Elasticity Procedure: The management entity in each layer receives load notifications from the other entities in the same layer. It then aggregates the information for each type of entity (e.g., the virtual diameter entities) across the layer and compares it to the appropriate threshold. The thresholds can either be pre-configured or determined on the fly, depending on the capabilities of the instantiated entities. The entities' individual thresholds are monitored and managed as the incoming requests are processed (i.e., no entity is assigned more requests than it can handle). The entities already handling the maximum number of requests ignore the incoming discovery requests.

If the layer's load exceeds the maximum allowed threshold, the layer management entity recognizes that the layer needs more entities and instantiates new ones. If the load is less than the minimum threshold, the layer management entity selects the entities with no load and terminates them. If all of the entities are currently busy, the management entity instructs the entities with the lowest loads to stop accepting new incoming requests, in order to accelerate the freeing-up of and termination of them. We avoid transferring the load between entities because it is expensive and the majority of requests (e.g., to add new information to or retrieve information from the database) are not time consuming.

To optimize communication with the layer management entity, each other entity sends its notification only when the local load exceeds a maximum or falls below a minimum local threshold.

III. APPLICATION SCENARIO

A. Overall scenario

To illustrate how the virtualized HSS can be used, we consider the case wherein three next generation service providers share the same Evolved Packet Core (EPC) network to offer video telephony services to their customers. EPC is the IP-based core network of the fourth generation of mobile networks [14]. We use a 4G networking environment because it is the closest to 5G thus far. We assume that none of the three providers own an HSS; instead, they use a virtualized HSS offered by the EPC network provider. We also assume that the end-users can subscribe to different classes of service (i.e. bronze, silver and gold) and therefore get different priorities for calls establishment. For instance, if not enough resources are available to establish a gold call, the application may need to terminate an ongoing bronze call to accommodate the new one.

To allow the network provider to offer virtualized HSS instances to video telephony service providers on the fly, the EPC network includes a virtualized HSS (VHSS) enabler. To request a new instance, the service provider directs a request to the VHSS enabler, which assigns a virtual diameter routing agent to it. This may be an existing agent or a new agent, depending on the security policies and the current load on the diameter layer. The same applies to the other entities in the architecture of Fig. 1. The interface between the video telephony application and the VHSS enabler is also REST-based and is a management interface used to instantiate and manage new virtualized HSSs on the fly.

B. Procedures

Diameter retrieval request processing: To establish a video session with the appropriate class of service, the telephony application should first retrieve the user's profile from the virtualized HSS. The application sends a multicast service request to identify the list of available virtual diameter routing agents. It then forwards the retrieval request to the appropriate agent among those that replied. The same procedure is undertaken by the chosen agent to find the adequate virtual diameter entity and forward the request it. The virtual Diameter entity then discovers the first entry point to the lower layer (i.e., the virtual database routing agent), which also chooses the most suitable virtual database manager entity. This is performed using the same procedures executed by the application and the virtual Diameter routing agent. To retrieve the requested data from the storage layer, the virtual database manager uses an index-based search to locate data and then send it back to the application.

Diameter storage update request processing: To illustrate this procedure, we use a case scenario in which an end-user is willing to change his class of service after a first session was successfully established. We assume that all of the entity addresses stored from the discovery processes (performed during the session establishment) are still valid. The application then sends the request to the virtual Diameter routing agent assigned to it during the previous procedure, and the request follows the same path as the retrieval request until it reaches the virtual database manager entity, which issues a B-tree index request to update the data storage.

Elasticity procedure: To illustrate this procedure, we describe the functioning of the management entity in the database management layer. Consider the case wherein a video telephony service provider experiences an unexpected increase

in session establishment requests (e.g., during a natural disaster). This creates an excessive load on the virtualized HSS, and the database management layer is flooded by database requests (e.g., to retrieve user profiles). Due to the notifications received from the layer's entities, the database management entity discovers the sudden excessive load. It then instantiates new routing agents and/or virtual database management entities to enable the layer to cope with the load increase.

IV. Prototype and Performance Evaluation

The video telephony scenario in the previous section was implemented as a proof of concept prototype. We used Fraunhofer Fokus OpenEPC Release 2 [15], a prototype implementation of the 3GPP EPC, as the 4G EPC. ScaleDB is used to implement the virtualized database. ScaleDB consists of four virtual machines: ScaleDB node, cluster manager, primary storage and mirror storage. Fig. 2 depicts the architecture. The end-users are connected to the EPC via a Wi-Fi network. The REST interfaces are implemented using the Restlet framework.

Testing Scenario: The testing scenario includes four end-users: Alice, Bob, Emma and Charlie. Alice's and Bob's accounts are configured to use the silver class of service, while Emma's and Charlie's accounts are configured with the gold class. We assume that the four end-users have the same service provider. We first initiate a call session between Alice and Bob. During the session, Emma calls Charlie. We assume that we do not have enough resources to establish the new call and that the application needs to terminate the call between Alice and Bob in order to free-up resources. The scenario ends when the second call (i.e., between Emma and Charlie) is established.

Environment settings: Seven machines are used for the EPC network, application, and end-users, each running VMware workstation. The first machine runs the video telephony service entities (i.e., video telephony application, application policy rules entity, and session information repository). Each of these entities runs in a separate virtual machine. The EPC entities (e.g. ePDG and P-GW) and the four end-users each run on a separate machine. For the virtualized HSS, we used six virtual machines running on a Xen Server. These represent the virtual diameter routing agent, the virtual diameter entity, the ScaleDB node (including both the virtual database routing agent and the virtual database manager entity), the cluster manager (i.e., the database layer management entity), and mirror and primary cluster accelerator server-CAS (i.e., storage).

Performance metrics: The prototype performance is assessed in terms of session establishment delay, the end-to-end time delay needed to establish a new call with a specific class of service. We compare the session establishment delays among the cases where the resources are available (i.e. *case-1*) and when the resource release is needed (i.e. *case-2*). We also compare the session establishment delays using a virtualized and a non-virtualized HSS.

Performance measurements and analysis: Each of the delays is calculated as an average of 10 measurements. In the case of a non-virtualized HSS, the delays for *case-1* and *case-2* have an average of 306 ms and 514 ms, respectively, and they

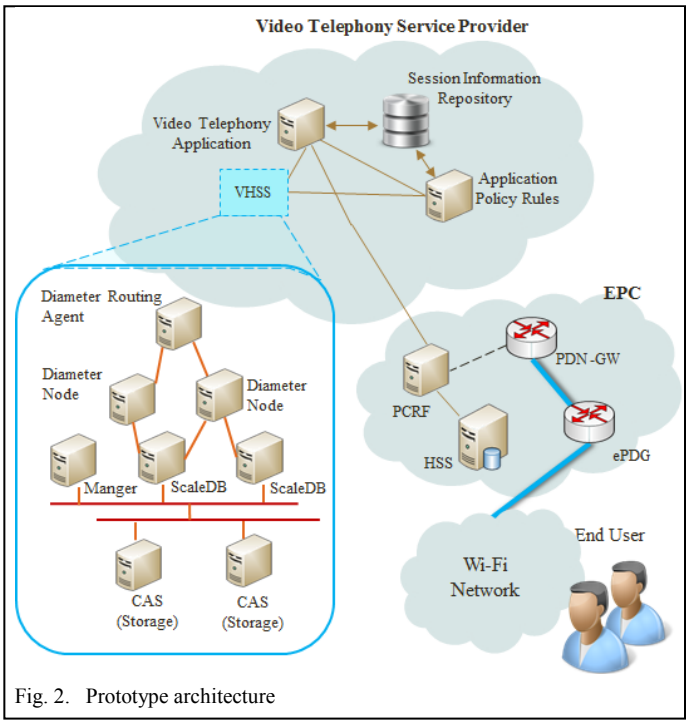

Fig. 2. Prototype architecture

are barely observable by the end-users in both cases. The increase in the delays in *case-2* is due to the extra delays for session modification (i.e., session termination in the context of the prototype).

The average delays when a virtualized HSS is used are 486 ms for *case-1* and 714 ms for *case-2*. These delays are higher than the ones with a non-virtualized HSS, but the difference remains barely observable by the end-users. This difference is induced by the three-layer architecture and about 50% of the increase is due to the REST communication among the different layers and to the communication between the diameter routing agent and the diameter node. These remain acceptable delays if we factor in the extra advantages (e.g., flexibly and scalability) gained by the virtualized architecture.

V. Conclusion

This paper proposes an architecture for a scalable and elastic virtualized HSS for 5G networks, which offers the same functionalities as the non-virtualized one and via the same interface. The architecture separates the HSS functionalities into three groups (i.e., HSS access, database management, and storage) and offers each group in a separate layer, allowing therefore each layer to grow and shrink independently as needed. The communication among the different layers is based on standard interfaces. The architecture uses redundancy to avoid single points of failure and it supports isolation among different HSS instances by applying a multi-tenancy approach.

Acknowledgments

This work is partially supported by the Canadian National Science and Engineering Research Council (NSERC) through the Canada Research Chair in End User Service Engineering for Communications Networks and through a Collaborative Research Grant (CRD) with Ericsson.

REFERENCES

[1] 3GPP TS 29.336," Home Subscriber Server (HSS) diameter interfaces for interworking with packet data networks and applications" version 12.1.0, 2013

[2] A. Osseiran et al; "Scenarios for 5G mobile and wireless communications: the vision of the METIS project," IEEE Communications Magazine, vol.52, no.5, pp.26,35, May 2014

[3] A. Khan et al., "Network Vitrtualization: A Hypervisor for the Internet?", IEEE Communications Magazine, vol.50, no.1, pp.136,143, January 2012

[4] T. Yang et al; "A new architecture of HSS based on cloud computing," 13th IEEE International Conference on Communication Technology (ICCT), pp.526,530, 25-28, Sept. 2011

[5] X. AN , F. Pianese, I. Widjaja , "DMME: Virtualizing LTE Mobility Managment", 36th Annual IEEE Conference on Local Computer Networks, 2011

[6] A. A. Soror , A. Aboulnaga , K. Salem, "Database Virtualization: A New Frontier for Database Tuning and Physical Design " , 23rd IEEE Data Engineering Workshop ,2007

[7] T. Kiefer, W. Lehner ,"Private Table Database Virtualization for DBaaS" , 4th IEEE International Conference on Utility and Cloud Computing, 2011

[8] S. Das, D. Agrawal, A. El Abbadi ," Elastras: An elastic , Scalable, and Self Managing Transactional Database in the Cloud", UCSB Computer Science Technical Report 2010-04

[9] H. AlJahdali, A. Albatli, P. Garraghan, P. Townend, L. Lau, and J. Xu, "Multi-tenancy in Cloud Computing," in 2014 IEEE 8th International Symposium on Service Oriented System Engineering (SOSE), 2014, pp. 344–351.

[10] F. Belqasmi, R. Glitho, C. Fu, "RESTful web services for service provisioning in next-generation networks: a survey", Communications Magazine, IEEE, vol. 49, no. 12, pp. 66–73, 2011.

[11] RFC 3082, "Notification and Subscription for SLP" , March 2001

[12] RFC 1157 ," Simple Network Management Protocol (SNMP)" May 1990

[13] D. Comer, "Ubiquitous B-Tree", Journal ACM Computing Surveys (CSUR), Volume 11 Issue 2, June 1979

[14] M. Olsson et al. , SAE and Evolved Packet Core: Driving the Mobile Broadband Revolution, Elsevier, Second Edition 2012

[15] Fraunhofer Fokus OpenEPC; available on web at: http://www.openepc.net/index.html, August 2014.