Modular Redundancy for Cloud based IMS Robustness

Muhammad Taqi Raza*
Computer Science Department
University of California, Los Angeles
taqi@cs.ucla.edu

Hsiao-yun Tseng
Computer Science Department
University of California, Los Angeles
tsenghy@cs.ucla.edu

Changlong Li
Computer Science Department
University of Science and Technology of China
liclong@mail.ustc.edu.cn

Songwu Lu
Computer Science Department
University of California, Los Angeles
slu@cs.ucla.edu

ABSTRACT

IMS (IP Multimedia Subsystem) is an emerging architectural framework that delivers a number of multimedia services – ranging from voice/video over LTE, interactive gaming and many more – in operational LTE network. Network operators are embracing cloud-based IMS to meet increasing multimedia traffic demand. They can easily and cost-efficiently implement multimedia applications while ensuring superior end-user experiences through always-on services and real-time engagement. In this paper, we reveal that cloud-based IMS cannot provide session-level resilience under faults and becomes the bottleneck to high service availability. The root cause lies upon the weak failure recovery mechanisms at both IMS protocol and cloud platform that terminate the on-going IMS control-plane procedure. To address this, we propose a design that provides fault-tolerance to IMS control-plane operations. Our design provides modular redundancy to perform real time failure recovery. As the system operates, the control-plane operations are logged at redundant IMS NFs modules. These logs are replayed from the failed operation to resume IMS working after failure. We build our system prototype of open source IMS over cloud platform. Our results show that we can achieve session-level resilience by performing fail-over procedure within tens of milliseconds under different combinations of IMS control-plane operations failures.

1 INTRODUCTION

IP Multimedia Subsystem (IMS) is a core network solution that delivers real-time multimedia services over LTE network. The use of an all IP core network for both signaling and transmitting media packets makes the IMS a prospective candidate for the cloud system. There are a number of advantages where cloud based IMS: (1) can use elasticity and utility style pricing of the cloud model; (2) reduces Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) that require to purchase IMS network functions along with the hardware; (3) brings agility in network function placement (4) quickly scales to support a variety of new multimedia services. In cloud based IMS implementation, IMS elements (also referred as network functions in this paper) are virtualized over data center network. These virtualized instances reside over Commercial Off-the-Shelf (COTS) distributed platform for the delivery of end-to-end multimedia services. The virtualized IMS (vIMS) implementation cannot enjoy traditional carrier-grade solutions implemented at hardware, software and service platforms to achieve ultra reliable communication. Instead, it needs to rely on IMS protocol-level and cloud platform-level fault tolerance mechanisms to serve its subscribers during faults. We discover that these mechanisms do not provide any fault tolerance support, rather they employ fault recovery procedures once user session states are lost. The failure recovery in cloud based IMS (i.e. vIMS) takes up to tens of seconds, which not only terminates on-going user service requests but also de-registers the device from IMS network.

In this paper, we address weak fault tolerance that exists in cloud based IMS. Our goal is to provide high service availability in vIMS where we aim to serve existing and new multimedia requests during faults. To do so, we provide modular redundancy to mask the real-time failures. Furthermore, we want minimum changes in current IMS implementation that do not conflict with standardized IMS operations and recovery procedures.

We propose a design that records Session Initiation Protocol (SIP) signalling messages exchange over IMS NFs redundant modules. These messages are replayed to recover the failed SIP operation. Our design implements a number redundant IMS NF modules which are assembled into their respective NF. It creates a number of new interfaces to connect original NFs with redundant NFs (that hold redundant modules). For every multimedia request (e.g. voice call request), vIMS NFs carry out a number of SIP operations (e.g. call invite, call progressing, call connecting and more). They execute these operations by exchanging a number of SIP messages between different IMS NFs. These signalling messages are also forwarded to respective redundant modules where they are logged. The failure recovery procedure involves quick failure detecting and failure tolerance. The failure detection is done through finite state machine (FSM) where different FSM states keep track of different stages of SIP procedure. When a particular IMS NF stops responding, it is actively probed to confirm the failure. On detecting the failure, neighboring IMS NF reconfigures its interfaces towards redundant modules of failed NF. It also replays the failed SIP operation (the SIP

*Corresponding author

MobiWac'17, November 21–25, 2017, Miami, FL, USA

ACM ISBN 978-1-4503-5163-8/17/11…$15.00.
https://doi.org/10.1145/3132062.3132064

stage at which the failure has occured). Our design performs smooth transition of IMS operations in fail-back procedure and avoids sudden traffic flow towards recovered NF.

We evaluate our design and gather results from our OpenIMS[1] implementation over Openstack[2]. We analyze our fault tolerance approach when failure occurs during device registration procedure. Our results show that our system reduces recovery time upto 20X compared to current vIMS implementation.

2 BACKGROUND

The IP Multimedia Subsystem (IMS) is a network architecture that delivers services based upon the Internet protocols to mobile subscribers. The IMS architecture brings multiple media, multiple point of access and multiple modes of communication into a single network, enabling end-user experiencing simultaneous voice, data, and multimedia sessions.

LTE being packet switched domain provides a best effort service to the users, with no guarantee about the amount of bandwidth a user gets for a connection and the delay experienced by the packets. Therefore, IMS is the preferred choice of mobile operators to support real-time multimedia service.

IMS Architecture: IMS operations are categorized into control-plane and data-plane operations, as shown in Figure 1.

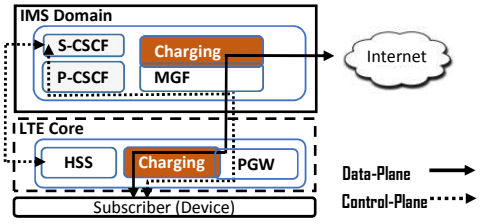

Figure 1: IMS architecture: an overview

Control-Plane contains all the Call Session Control Function (CSCF) entities to support the call session control. The CSCF performs all the signaling operations, manages Session Initiation Protocol (SIP) sessions and coordinates with other network entities for session control, service control and resource allocation. It consists of two main entities: the Proxy-CSCF (P-CSCF) and Serving-CSCF (S-CSCF). LTE subscriber first registers with LTE core network and then initiates IMS signalling over IMS control-plane. These signalling messages are relayed by Packet Data Network Gateway (PGW) of LTE towards P-CSCF. The P-CSCF being an access point to IMS acts as a SIP proxy server for all the user equipments. P-CSCF is only a point of access to IMS and does not authenticate within the IMS. The S-CSCF is responsible for confronting devices that try to establish session without being registered in the network. The S-CSCF is the core of the IMS, providing the point of control within the network that enables operators to control all service delivery and all sessions. It implements a SIP server having in charge of handling all the aspects of the services for a subscriber, maintaining the status of the sessions the user has initiated, and controlling and delivering of the multimedia contents. The S-CSCF has knowledge of all the services subscribed by the users, by downloading from the Home Subscriber Server (HSS). The HSS is a database that contains all subscribers' data like the services that are allowed to access, the network in which each subscriber is granted to roam, and the information about the location of every subscriber. An important function of the HSS is to provide the encryption and authentication keys of the mobile devices.

Data-Plane includes media-gateway NF which processes, stores data and generates services for the subscribers. Once user session has been established, the user data-plane traffic is sent to Media Gateway Function (MGF). The MGF connects LTE core domain (via PDN gateway - PGW) with IMS domain for multimedia service and converts between different transmission and coding techniques. Moreover, it employs monitoring schemes to determine policy rules in real-time.

3 EXISTING FAULT TOLERANCE SUPPORT

Existing fault tolerance schemes can be divided into a) IMS protocol level, b) vendor specific platform level, and c) cloud platforms level. Table 1 provides an overview of these mechanisms, their functions, pros and cons as well comparison with proposed design.

3.1 IMS Protocol Fault Tolerance

IMS protocol level fault tolerance can be further categorized into I) control-plane and II) data-plane fault tolerance approaches.

3.1.1 IMS protocol control-plane fault tolerance. IMS protocol does not provide any fault-tolerance mechanism, rather it relies on fault recovery procedures.

On S-CSCF Failure: When S-CSCF failure occurs then device registration procedure is aborted if in-progress, or device is de-registered from the IMS network if it has already been registered. After failure, S-CSCF recovery is performed by HSS through IMS serving proxy. HSS performs two separate actions depending whether this failure occurs during device initial registration or when the device has already been registered with IMS. In the former case, device has not yet established connection with IMS, so it is safe to switch to new S-CSCF. Therefore, HSS re-assigns another S-CSCF and all requests coming from P-CSCF are forwarded to new S-CSCF. We show this procedure in Figure 2a. In the later case, HSS restores device session information to S-CSCF. The standard S-CSCF recovery procedure mandates S-CSCF to be rebooted. The S-CSCF loses all subscribers data when it restarts after a failure or it is unable to trust any data after it resumes operation which is due to the fact that it may have lost profile updates from the HSS during the service interruption period. Therefore, failure recovery from HSS is mandatory so that device policy information is restored without requesting information from LTE core network elements such as PGW. When device registers with IMS network at first time, its SIP proxies (including P-CSCF address), contact information, authentication information etc are stored at HSS. If any of above data is changed, S-CSCF updates the record at HSS. Moreover, S-CSCF stores caller-ID, From-To record, destination device information at HSS before media traffic flow starts.

There are several issues with such restoration procedure. First, S-CSCF failure is propagated to device which is against the philosophy of fault tolerance that requires system failures should be hidden from end devices. Second, on S-CSCF failure, IMS service is temporarily suspended. Such failure recovery procedure is not sufficient for crucial telephony service that require high reliability (*99.999% also*

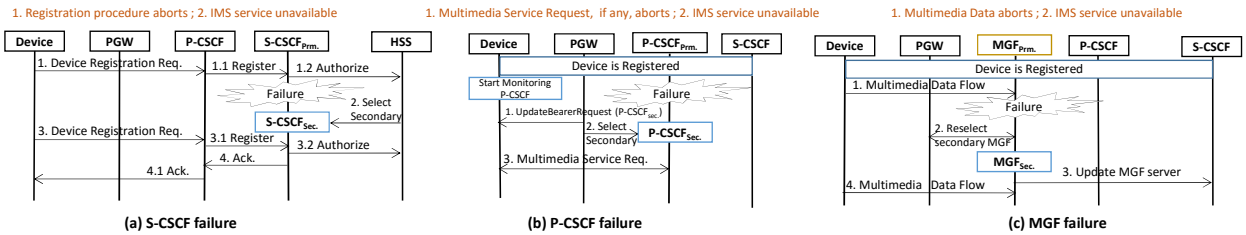

Figure 2: Recovery from S-CSCF, P-CSCF and MGF failures

known as five 9s reliability). Third, although, IMS being an overly network on LTE service that can be provided by third party service provider, heavily relies on LTE core network element, i.e. HSS, for S-CSCF recovery procedure. Therefore, LTE network operators need to provide access to subscribers private information to third party service provider. Fourth, instead of storing backup data on redundant S-CSCF, it is stored in HSS. As a result HSS can become single point of failure for the stored backup for all devices in IMS.

On P-CSCF Failure: Two mechanisms, one at device and the other at IMS side, are implemented to identify P-CSCF failure. When the device registers with IMS service, it starts monitoring P-CSCF health by periodically sending keep-alive messages. If P-CSCF does not respond to device keep-alive messages, device declares P-CSCF as *failed* and contacts PGW to acquire new P-CSCF address. It then re-initiates the registration procedure with IMS. However, it is possible that P-CSCF is active but the link between P-CSCF and S-CSCF is broken. In this case, P-CSCF failure is detected by PGW that informs UE by sending new P-CSCF address. When the device receives new P-CSCF address, it declares previous P-CSCF as unavailable and sends IMS registration request towards newly assigned P-CSCF. We have shown P-CSCF recovery procedure in Figure 2b.

Although, P-CSCF failure recovery procedure tries to mitigate IMS unreachability by assigning new P-CSCF, it introduces a number of problems. First, IMS relies on device to recover from failure by performing re-registration procedure with IMS. Second, during P-CSCF failure as well as the time device connects to new P-CSCF, IMS service remains unavailable to subscribers. This potentially compromises promised 99.999% voice service availability by network operator. Third, P-CSCF failure not only terminates any control-plane session, but data-plane traffic is also aborted, where device initiates re-registration procedure with IMS network. Therefore, we can say P-CSCF has ripple effect to data-plane traffic, even though P-CSCF does not play any role over data-plane communication.

3.1.2 IMS protocol data-plane fault tolerance. Like control-plane, IMS protocol does not provide any data-plane fault-tolerance mechanism. MGF is a critical IMS component that connects IMS with the outside world. When MGF failure occurs, the media traffic terminates between source and destination. The path between MGF and PGW is declared out of service. This failure is also propagated to S-CSCF that prevents device to use media service. Thereafter, PGW reselects a new MGF, as shown in Figure 2c, with the help of media gateway control function. Once new MGF becomes operational, all media-traffic is forwarded to new MGF.

Note that the device remains unreachable by the time its MGF recovers from failure. There is no mechanism that informs device once new MGF starts serving. Therefore, the user needs to keep trying until its requests are being served by IMS. Lack of data-plane fault tolerance not only renders IMS service black-out, but also IMS loses important device information. For example, such failures result into incorrect billing information, and loss of policy information at charging function.

3.2 Vendor Specific IMS Platform Fault Tolerance

Vendor specific IMS platforms provide three lines of defense at a) hardware, b) software, and c) overly manager. Figure 3 highlights hardware and software fault tolerance mechanisms for a number of fault models.

Hardware Fault Tolerance: Purpose-built hardware platforms have been developed which can "tolerate" faults. They continue to provide the required functioning despite occasional internal components and modules failures, either transient or permanent. Fault tolerance is achieved by providing redundant hardware modules. Whenever a fault is encountered, the redundant modules takeover the functions of failed hardware module. Such redundancy not only caters hardware faults but also be exploited to perform system functions under zero fault conditions. Therefore, load sharing is achieved without installing redundant hardware system. For example, Ericsson's Blade Systems (EBS)[3], Alcatel-Lucent's Element Management System (EMS)[4], and Huawaei's ATCA[5] provide network function availability even during failures and operational maintenance without disturbing multimedia traffic flow in the network.

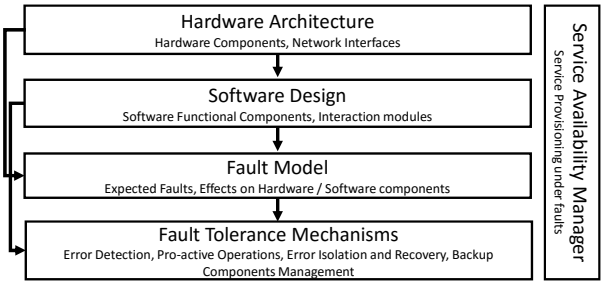

Figure 3: Vendor-specific fault tolerance approaches

Software Fault Tolerance: IMS equipment vendors provide strong coupling between their software and hardware. Software fault tolerance is achieved by software design and through programming languages.

Table 1: Summary of IMS fault tolerance schemes

Fault Tolerance Support	Function 1	Function 2	Function 3	Prose	Cons
IMS Protocol	Ignores device request and re-selects NF	Reboots the server and obtains information from HSS	Drops the call and let device re-dial	Simple logic, redo the process	call drops, service unsalability for longer durations
Telecom vendor-specific platforms	Coupled software and hardware	overload protection function	geo-redundancy pool	IMS signalling resilience, no call drop	Expensive dedicated platform, not scalable, voice jitters
Cloud Platform	restarts current process and escalates faults to higher layers	exploits infrastructure level redundancy	Runs software vendor specific approaches	Simple, works with all type of cloud applications, provides infrastructure level availability guarantees	Call drop, system unavailable for longer time, no instance level fault tolerance, single-point-of-failure controller
Proposed Design	NFs modules act each other proxy	support of multiple IMS NFs failure	exploits modular redundancy for failed operations recovery	IMS signalling resilience, no call drop, simple logic, exploits already available modules, scalable	Minor changes required at IMS NF implementation, voice jitter during failure recovery

Software design ensures redundancy, both for error detection and error recovery. Such redundancy is not mere replication of programs rather redundancy of design. As the system operates, functional checks are made on the acceptability of the results generated by each piece of software component. When an abnormal result is generated a spare software component is switched-in to replace the previous faulty software component block. This new component itself is an independent design that takes external factors into account when iterating through a set of inputs. Therefore, it can cope with the circumstances that has caused the main software block to fail. These circumstances mainly arise because of faulty process synchronization, race conditions, the timing constraints and software block's interactions with other processes.

Programming languages provide high availability through declarative programming techniques. These techniques do not guard against errors, rather they try to minimize the impact of errors and recover from them as quickly as possible. As a matter of fact, during faults, data may get corrupted and represent false information. Such incorrect data may propagate further within the system and may get written to database; bringing incorrectness to whole system. To address this issue, as soon as some fault occurs, crashed code, through functional programming, is isolated from the rest. Then only that problematic functional block is restarted and later re-integrated within the system. Examples of these programming langues include Ericsson's ERLANG[3], Alcatel-Lucent's NVP[6], and Huawaei's Fusion[7]. They develop scalable real-time systems with requirements on concurrency, distribution and fault tolerance.

Service Availability Manager: Service availability manager acts as an overlay on top of purpose-build IMS platform that enables end-to-end network and service management across all IMS NFs. Service manager's goal is to maximize operational efficiencies through service provisioning and troubleshooting. It also provides proactive assurance and flexibility that eases integration into the IMS network. The examples include Ericsson's Network Manager (ENM)[8], Alcatel-Lucent's Service Aware System (SAM)[9], and Huawaei's Managed Services Unified Platform (MSUP)[10] that provide in-band and out-of-band management and isolate the failure from IMS managed network. Moreover they also provide provisioning of service mirrors to monitor service traffic for troubleshooting or official surveillance purposes.

3.3 Cloud Platform Fault Tolerance

Network operators are keen to deploy their IMS platform over general-purpose servers over cloud platform. Lack of fault-tolerance support at IMS protocol level requires fault-tolerance at cloud platform should be exploited for high availability. We find that current cloud systems being Infrastructure as a Service (IaaS) do not provide instance-level fault tolerance. We detail cloud based fault tolerance schemes as:

Hardware redundancy: High availability is implemented with redundant hardware running redundant instances of each service. Example of such cloud platform is OpenStack[11]. If one piece of hardware running one instance of a service fails, the system can then fail-over to use another instance of a service that is running on hardware that did not fail[11]. However, IMS service remains unavailable during fault-recovery period.

Fault Injection: OPNFV is tailored open source cloud solution for network function virtualization (NFV). Although, OPNFV runs network services that have very high resilience and availability requirements, it fails to achieve these goals. Current OPNFV fault-tolerance schemes are same as provided by OpenStack. OPNFV community has started project named "Vaccine" to extend fault tolerance on the three layers, i.e. hardware, Hypervisor and VM, using methods like fault injection[12].

VM restart: CloudStack recovers from faults either by restarting VM, or migrating to secondary VM, if primary VM failure persists. This implementation not only loses vital subscribers records but also becomes a bottleneck if several attempts are made to recover faulty VM[13]. Moreover, CloudStack does not provide any monitoring feature for secondary VMs. This may result in selecting already unresponsive secondary VM for fault-recovery.

Context switch to replica: In case of active node failure, OpenNebula yields control to passive node, which is acting as redundant node. After fail-over, secondary node will see the resources in the exact same way as the one in the server that crashed[14]. However, there will be a set of virtual machines which will be stuck in transient states while copying the disks to the target host server. Such recovery procedure adds significant latencies at system to become operational again – which is not acceptable to high availability of IMS.

4 MODULAR REDUNDANCY DESIGN

Our Approach Our approach is motivated from the success of fault tolerance approaches in traditional carrier grade boxes. Network vendors can tolerate faults mainly through redundancy (at software and hardware). We argue that same style of redundancy is required to achieve high availability for failure susceptible network functions – implemented as software. Contrary to redundancy provided by software and cloud platforms, we aim to detect and recover from

failures in real time. Our goal is to keep control and media planes in-tact during failures. We first discuss our failure model and later describe our design.

Failure Model IMS employs a number of procedures prior and during serving users. First, P-CSCF performs DNS queries to retrieve S-CSCF address. Then it establishes the connection with S-CSCF for performing control-plane operations. S-CSCF is connected with HSS to authenticate, authorize and retrieve users records. Through PCRF (Policy and Charging Rules Function), S-CSCF obtains users QoS profile and applies it to every subscriber. Failure can occur (1) in retrieving DNS records when P-CSCF is relocating subscribers to new S-CSCF for load balancing, (2) populating subscribers record from HSS and PCRF at the time when S-CSCF is registering the subscribers, and (3) when on-going multimedia sessions drop during P-CSCF and S-CSCF communication. These failures can occur because of hardware failure, software failure, or network failure between IMS NFs and helper functions (such as PCRF, DNS or HSS). In any of such failures, IMS instance cannot obtain desire information, as a result IMS control-plane/data-plane traffic is aborted. For simplicity, we regard all such failures as "fail-stop" failure when either one or more IMS NFs fail to respond. We immediately start failure recovery procedure to maintain session-level resilience.

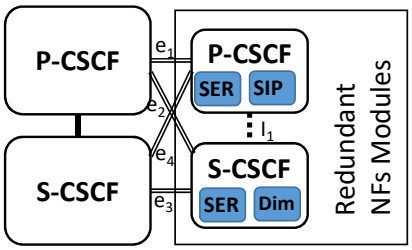

Figure 4: Modules redundancy: an overview

Overview We explain our design through Figure 4. We show two IMS NFs (P-CSCF and S-CSCF) in Figure 4 to elaborate our design. However, our design is generic and provides fault tolerance to all IMS NFs (including media NFs). We propose that each IMS NF modules must be replicated to recover from failure. The number of modules to be replicated has a tradeoff with the type of failure. To support complete NF failure resilience (fail-stop failure type), we are required to duplicate *all* modules of that NF. P-CSCF implements a number of modules (i.e. SIP Express Router and SIP client and more). SIP Express Router (SER) anchors as a routing block of all IMS NFs. It supports 4 major features. (1) The initial routing block (route[0]) acts as the point of entry to SER for all requests regardless of method, e.g. process registration requests. (2) SER parses requests and transfers program control to a destination specific route block. (3) It supports various transformations (e.g., canonicalizing request-URIs). (4) It directly forwards the requests to registered clients. P-CSCF implements SIP proxy module (not to be confused with SIP client) that implements user agent scenarios and establishes and releases multiple calls with the INVITE and BYE methods. This module directly communicates with SIP client implemented at device(s) and handles real time requests.
Similarly, S-CSCF also implements SER to support SIP signalling routing. Moreover, it implements Diameter protocol (Java Diameter Peer library) for user authentication on registration, and communicates with HSS. An incoming (from P-CSCF) Diameter message received by a communicator (Diameter peer) is pushed in a TaskQueue at first. This TaskQueue is a FIFO (first-in first-out) blocking queue. As soon as the message is available in the queue, a DiameterWorker takes it out of the queue and delivers it to a set of event listeners defined by the type of incoming request.

Our design implements four new external interfaces ($e_1 - e_4$) where original P-CSCF and S-CSCF NFs are connected with replicated P-CSCF and S-CSCF NFs (holding respective redundant modules), as shown by double straight lines in Figure 4. These replicated P-CSCF and S-CSCF NFs also connect with each other through a separate interface (i.e. I_1 shown by dotted line). This fifth interface is added to support more than one NF failure case. We explain our design through three steps procedure, (1) before failure, (2) during failure, and (3) after failure. The pseudo procedure is shown below.

```
Before Failure:
    if ( message is received from device / S-CSCF )
        forward the message to S-CSCF / device
        forward the message to redundant modules over e1 and e2
    if ( message is received from HSS / P-CSCF )
        forward the message to P-CSCF / HSS
        forward the message to redundant modules over e3 and e4
During Failure:
    detect the failure using fast probing
    reconfigure the interfaces towards respective redundant modules
    replay the failed SIP operation (re-execute failed SIP stage)
After Failure:
    if (the device is inactive)
        reconfigure the interfaces back to recovered NF
    if (new registration request is received)
        forward it to recovered NF
```

Pseudo code of failure recovery procedure

4.1 Before Failure

During regular vIMS operation, our design creates SIP signalling message level redundancy. These redundant messages are replayed in case original NF stops responding. The P-CSCF extends two new interfaces, one (e_1) connects to redundant P-CSCF modules, and the other (e_2) extends to redundant S-CSCF modules. The device is oblivious of any redundancy at IMS and communicates with P-CSCF. P-CSCF that intercepts incoming/outgoing device messages forwards them to redundant P-CSCF NF modules. In this way, the redundant P-CSCF modules are aware of all signalling messages coming from/sent to device. Whenever P-CSCF communicates with S-CSCF, it also sends these messages to redundant S-CSCF NF modules over e_2 interface. Similarly, S-CSCF implements two separate interfaces, one (e_3) connects to redundant S-CSCF modules, and the other (e_4) links with redundant P-CSCF modules. All NFs beyond S-CSCF (such as HSS) are oblivious P-CSCF/S-CSCF modules redundancy. Whenever S-CSCF receives incoming/outgoing SIP signalling messages, it also forwards them to redundant S-CSCF NF modules over e_3 interface. Moreover, all communication between S-CSCF and P-CSCF are recorded at redundant P-CSCF NF modules over e_4 interface.
Our procedure logs all signalling messages between P-CSCF and

S-CSCF at real time. This helps in providing session-level resilience during failures.

4.2 During Failure

Our design is required to detect and tolerate the failure. The failure detection can only be done if the failed NF remains unresponsive for a certain amount of time. We argue that because P-CSCF and S-CSCF are directly connected, the control-plane message retry interval between pair of these NFs should be multiple of their message Round-Trip-Times (RTTs). We propose that during an IMS operation, if one NF does not receive response from other NF, it retries the message every 5*RTT with maximum 5 number of retries. In case of no reply within 25*RTT, non-responding NF's status is changed from in-service to failure-prone. Afterwards, in-service NF starts probing failure-prone NF on every RTT, with 5 number of retries. If non-responding NF still does not reply, we change its status from failure-prone to out-of-service.
This is how we detect and confirm failure in 5xRTT and 25xRTT, respectively; and perform failure recovery in 30xRTT.

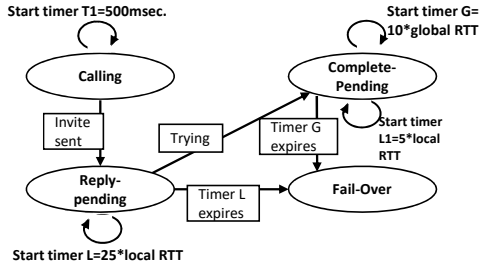

Figure 5: State transition diagram detecting failure in IMS system

Detecting Failure: Our failure detection is done through finite state machine (FSM). We introduce few temporary states that keep track of different stages of SIP procedure. We explain our proposed state transition diagram by using voice call operation which is running at P-CSCF. Figure 5 shows when device sends *invite* request, it transitions from *calling* to *reply-pending* state. If S-CSCF replies to *invite* request, P-CSCF moves to *complete-pending* state, otherwise after expiration of local timer (timer L) of 30xlocal RTT[1] failure recovery procedure kicks in. In *complete-pending* state, S-CSCF keeps probing P-CSCF every 5xlocal RTT. Note that, this probing message is sent only once every 5xRTT even though more than one device has progressed to *complete-pending* state.
Timer G[2] keeps track of response from target IMS system. When there is no response from target IMS system, a *re-invite* request is sent. By sending *re-invite* message, target IMS system may receive more than one *invite* request messages where it discards duplicate *invite* requests. This proactive probing helps in quick recovery from transitive faults when previous invite request(s) is(are) failed to be delivered to target device.

Failure Recovery: After detecting failure, we perform failure recovery procedure when one IMS NF declares other NF out-of-service and reconfigures its interface towards redundant modules of failed NF. We explain this procedure through S-CSCF NF failure scenario. When S-CSCF does not respond within *30xRTT*, we deactivate the link between P-CSCF and S-CSCF (solid line in Figure 4), and activate the link between P-CSCF and redundant S-CSCF modules (via interface e_2). Now P-CSCF forwards all the traffic to redundant S-CSCF components through e_2 interface. The redundant S-CSCF modules resume the operation from the step at which the failure has occurred.
When a timeout is not a failure: It is possible that proposed, but configurable, timeout value does not represent actual NF failure. Such rare case occurs when the link between S-CSCF and P-CSCF is severely congested or S-CSCF goes through random failure – not impacting the function of S-CSCF. We still define such case as a failure that impacts user Quality of Service (QoS). However, we handover to the actual S-CSCF function through fail-back procedure (i.e. after-failure case). Note that, in order to avoid ping-pong effect between failure-recovery and fail-back procedures, the fail-back procedure does not occur until certain time (30 minutes in our implementation) has passed since the start of proposed failure-recovery procedure.

We should mention that during control-plane fault tolerance procedure, we do not disturb any other IMS default failure recovery and health-monitoring procedures, and let IMS protocol/cloud platform recover from failure (either through reboot or switching to alternate S-CSCF NF).

4.3 After Failure

Fail-back procedure starts when (1) minimum time (i.e. 30 minutes in our implementation) has elapsed since proposed failure-recovery procedure, and (2) failed NF has recovered from failure either through IMS protocol or cloud platform failure recovery procedure.
Smooth transition: In this procedure, in-service NF (comprise of redundant modules), starts redirecting traffic to recovered NF. In our approach, we do not migrate users' on-going SIP session information, rather we migrate registration information of a device in its idle mode (when there is no ongoing SIP session). We explain this with an example where S-CSCF has recovered from the failure. In migration procedure, we only send device identities to recovered S-CSCF. We let S-CSCF retrieve the rest of the information, i.e. network info, charging function address, and preferred service information from HSS. We also require S-CSCF to retrieve authentication vector from HSS and re-authenticate the device[15]. Moreover, all new registration requests are diverted to recovered S-CSCF.
In short, our scheme performs smooth transition towards recovered NF by not exposing it to signalling load of active subscribers.

5 PROTOTYPING

In our implementation, we use open source IMS platform (OpenIMS[1]) and open source cloud operating system (OpenStack[2]) to implement the functionalities of IMS protocol and its virtualization, respectively. The OpenIMS provides basic implementation of IMS NFs (both S-CSCF and P-CSCF) and HSS that can be deployed over Unix-based platforms like Linux, BSD or Solaris. The OpenStack provides full flexibility on how IMS NFs are managed on cloud platform. It provides abstraction of common hardware resources through virtualization and meets compute, networking and storage

[1]We propose local RTT, measured between P-CSCF and S-CSCF NFs.
[2]We propose timer G, calculated based on global RTT – measured between source IMS and destination IMS systems.

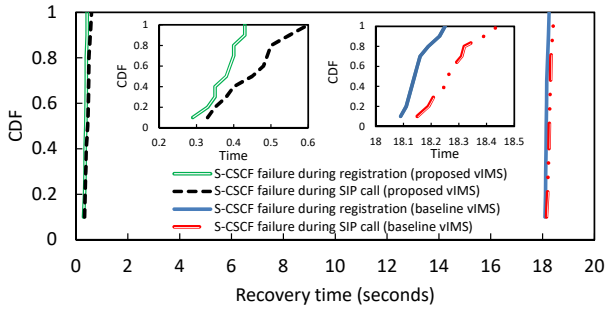

Figure 6: Service recovery time after S-CSCF fail-stop failure: comparing proposed-vIMS with baseline-vIMS

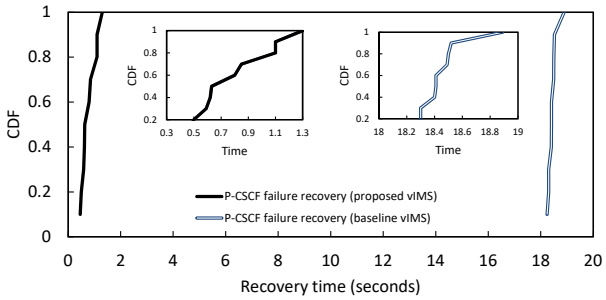

Figure 7: Service recovery time after P-CSCF fail-stop failure: comparing proposed-vIMS with baseline-vIMS

demands of different IMS applications. We spent significant efforts to modify source code in both platforms to suite our needs.

5.1 Prototyping Baseline vIMS

OpenIMS has coupled all IMS NFs by implementing them over single virtual machine (e.g. VMware) that provides optimal performance when hundreds of users are accessing IMS network at the same time. For our cloud IMS deployment, we first decouple IMS NFs into separate VM. Then these VMs are bridged through virtual network interface. These stand-alone VMs are deployed over OpenStack to achieve baseline vIMS implementation. We also provide 1:1 redundant copy of IMS NFs to achieve minimum industry requirement for cloud applications[16]. We use default timers as specified by IMS and OpenStack documents[17] and consider this implementation as base-line vIMS with which our design is compared.

5.2 Prototyping Proposed vIMS

We exploit OpenIMS modular structure and adopt its implementation to our needs. We describe our efforts as below:

Call Session Control Functions (CSCFs): Our design switches between failed CSCF and redundant CSCF in real time. To do so, we modify SIP Express Router (SER) of OpenIMS to implement *sendto()* and *receivefrom()* for more than one NF module. SER handles all SIP registration, SIP service requests, and directs their signalling to P-CSCF and S-CSCF functional modules.

Finite state machine: In FSM implementation an operation must start from an initial state and transitions to another accepted state. To achieve this, we create FSM transition table in each NF that transitions from a given state to a new state when either SIP procedure has progressed or its guard timer has expired. By doing so, the proposed FSM only executes using necessary functional modules.

Failure recovery: To successfully execute failure recovery procedure, we are required to immediately resume IMS operation from SIP procedure stage at which the fault has occurred. To achieve this goal, we keep track of on-going device session before fault using a hash table to store/retrieve user's session information. When failure occurs, the neighboring IMS NF detects it and activates the redundant NF module(s). These modules starts the failure recovery procedure by retrieving last stored SIP procedure stage and related session information from hash table (sent by the NF that has detected the failure). The network configurations at incoming and outgoing interfaces are also modified. The redundant NF modules applies filters to distinguish whether service requests and registration requests are coming from existing subscribers or new ones to optimize their working.

6 EVALUATION

We evaluate the fault-tolerance mechanisms of our proposed vIMS. The baseline vIMS implementation, described in Section 5.1, serves as the baseline of our experiments. We run our tests on a local network of servers with Intel Xeon(R) ES-2420 V2 processor at 2.20GHZ x 12, 16M Cache size, and 16GB memory. For each VM, we use Ubuntu Server 14.04.3 LTS with the OpenIMS Core.

We consider session resilience when NFs stop responding during device registration procedure. We consider fail-stop failure at: (a) P-CSCF and (b) S-CSCF.

Failure recovery? The device initiates control-plane operation (either registration or SIP call) with IMS network. While control-plane operation is on-going, we let one of the IMS NF to crash. Figure 6 and Figure 7 show experimental results along with enlargements of critical regions. Irrespective of P-CSCF or S-CSCF crash, the control-plane operation aborts in 5 seconds (in accordance to operational IMS network, we have set device timeout value as 5 seconds). OpenStack takes 10 seconds to detect the failure and takes another 8 seconds to prepare backup NF and restores the service. In about 18 seconds, the baseline vIMS comes back to service, but the client does not make a new registration attempt, as it has timed-out 13 seconds prior to recovery.

In contrast, at worst, proposed vIMS takes about 600ms in case of S-CSCF, and about 1300ms for P-CSCF case to recover from failure. We observe two different recovery phenomena. First, when S-CSCF crashes, the recovery is made at redundant S-CSCF modules that successfully resume the failed S-CSCF operation. But when S-CSCF takes more than 500ms to perform recovery, the timeout happens at device. On time-out, device retries the SIP operation and redundant S-CSCF modules successfully execute the failed operation. Second, when P-CSCF crashes, the device often experiences time-out (only 20% of the time P-CSCF redundant modules take charge within 500ms, as shown in Figure 7). This is because a redundant P-CSCF module needs to re-establish the IPsec tunnel with the device that aborts on-going control-plane operation. Once new IPsec tunnel has been established, device re-attempts its unsuccessful operation and is served by redundant P-CSCF modules.

7 RELATED WORK

Our work is in contrast with other recent efforts on NFV, vIMS and middle boxes' fault tolerance space.

NFV: [18] provides general purpose NFV platform, [19] and [20] make use of software and hardware choices to meet specific service demands. [21] discusses NFV integration in mobile network. [22] challenges the way NFV is done for LTE and proposes an alternate way (based on NFs logic) of doing LTE-NFV.

Cloud based IMS: [23] provides dynamic resource allocation algorithm for vIMS, [23] discusses merits of deployment strategies of vIMS. [24] enhances vIMS features for M2M. But these efforts do not discuss fault tolerance aspects in vIMS. Recent work [25] reveals that modular vIMS design can introduce latencies and cause failures. In contrast, our work does not discuss working of individual module, rather it aims to provide fault tolerance to a system as a whole.

Fault Tolerance: [26] and [27] propose logging NF states during normal operations and reconstructing them after a failure. Their approaches cannot address real-time and transitory NF sessions recovery. [28] and [18] discuss fault tolerance in non-IMS (SIP based) voice over IP applications.

Contrary to above mentioned works, our design provides session-level resilience during faults.

8 CONCLUSION

Our study shows that cloud based IMS cannot achieve high service-resilience solely relying on cloud-computing platforms. Our design is inspired from the way failures are masked in carrier grade IMS. We propose providing software modular redundancy to recover from failures in real-time. Our design does not bring any changes to standardized IMS implementation, other than creating and configuring few network interfaces for logging signalling messages and recovering from failures.

Future work: In future we seek to propose a solution that achieves vIMS session resilience without introducing any redundancy. Moreover, we aim to provide IMS fault tolerance solution at data-plane, addressing security issues due to failures, and mitigating failures due to NFs handoffs when network elements are scaled up and down.

Acknowledgement

We thank anonymous reviewers for providing excellent feedback. This work is also supported in part by NSF grants (CNS-1422835 and 1528122).

REFERENCES

[1] OpenIMS – The Open Source IMS Core Project. http://www.openimscore.org/.
[2] OpenStack Open Source Cloud Computing Software. https://www.openstack.org/software/.
[3] Ericsson Blade System (EBS) for IMS. http://archive.ericsson.net/service/internet/picov/get?DocNo=266/03819-FAP130506&Lang=AE&HighestFree=Y.
[4] Alcatel-Lucent End-to-End IMS Solution. https://www.alcatel-lucent.com/solutions/communications-collaboration.
[5] High reliability using ATCA platform for IMS. http://www1.huawei.com/en/products/core-network/singlecore/ims-core/index.htm.
[6] N-Version Programming. http://www.inf.pucrs.br/~zorzo/cs/n-versionprogramming.pdf.
[7] Fusion Programming. www.huawei.com/ilink/en/download/HW_327323.
[8] Ericsson redefines network control and analytics. http://e.huawei.com/en/products/enterprise-networking/wlan/access-controllers/ac6605/.
[9] Alcatel-Lucent 5620 – SERVICE AWARE MANAGER . https://infoproducts.alcatel-lucent.com/cgi-bin/dbaccessfilename.cgi/3HE03415AAAA01_V1_Alcatel-Lucent/.
[10] MSUP – Standardization and Converged Operation. http://www1.huawei.com/en/video/hw-263534.htm/.
[11] High availability concepts in OpenStack platform. http://docs.openstack.org/ha-guide/.
[12] Fault tolerance and recover-ability for OPNFV platform. https://wiki.opnfv.org/project_proposals/vaccine/.
[13] System Reliability and High Availability in CloudStack. http://docs.cloudstack.apache.org/projects/cloudstack-administration/en/4.6/reliability.html/.
[14] OpenNebula:Host and VM High Availability. https://docs.opennebula.org/5.2/advanced_components/ha/frontend_ha_setup.html#overview.
[15] 3GPP. TS33.203: Access security for IP-based services, Sep. 2012.
[16] A. Bernstein. Availability For Network Function Virtualization. *Cisco: Technical Report*, 2015.
[17] 3GPP. TS186.008–2: IMS Network Testing: IMS Configurations and Benchmarks.
[18] S. Palkar and et al. E2: a framework for NFV applications. In *ACM SOSP*, 2015.
[19] R. Mahindra and et al. A practical traffic management system for integrated LTE-WiFi networks. *ACM Mobicom*, 2014.
[20] L. Osmani and et al. Building Blocks for an Elastic Mobile Core. *ACM CoNEXT on Student Workshop*, 2014.
[21] I. F. Akyildiz and et al. Wireless software-defined networks (W-SDNs) and network function virtualization (NFV) for 5G cellular systems: an overview and qualitative evaluation. *Elsevier Computer Networks*, 2015.
[22] M. T. Raza and et al. Rethinking LTE Network Function Virtualization. In *IEEE ICNP*, 2017.
[23] G. Carella and et al. Cloudified IP Multimedia Subsystem (IMS) for Network Function Virtualization (NFV)-based architectures. In *IEEE Symposium on Computers and Communication (ISCC)*, 2014.
[24] M. Abu-Lebdeh and et al. Cloudifying the 3GPP IP multimedia subsystem for 4G and beyond: A survey. *IEEE Communications Magazine*, 54(1):91–97, 2016.
[25] M. T. Raza and et al. Reducing Latencies and Improving Fault Tolerance in NFV of 3GPP Standardized IMS. In *IEEE CNSM*, 2017.
[26] J. Sherry and et al. Rollback-Recovery for Middleboxes. In *ACM SIGCOMM*, 2015.
[27] S. Rajagopalan and et al. Pico Replication: A high availability framework for middleboxes. In *ACM Cloud Computing*, 2013.
[28] M. Bozinovski and et al. Fault-tolerant SIP-based call control system. *IET Electronics Letters*, 39(2):254–256, 2003.