Uninterruptible IMS: Maintaining Users Access During Faults in Virtualized IP Multimedia Subsystem

Muhammad Taqi Raza, *Member, IEEE*, and Songwu Lu, *Fellow, IEEE*

***Abstract*— Network function virtualization (NFV) of IP Multimedia Subsystem (IMS) pose promise to service increasing multimedia traffic demand. In this paper, we show that virtualized IMS (vIMS) is unable to provide session-level resilience under faults and becomes the bottleneck to high service availability. We propose a design to provide fault-tolerance for vIMS operations. In control-plane, our system decomposes single IMS operation into different atomic actions, and partition these actions into critical and non-critical actions. Only the critical actions are then monitored in real time and the system can easily resume IMS operations after failure. In data-plane, we decompose multimedia traffic flows and partition each multimedia service as a separate Virtualized Network Function (VNF). Through data-plane partitioning, our design restricts the damage from faults to only failed VNF. Thereafter, impacted service flow is merged with other ongoing service flows. We build our system prototype of open source IMS over virtualized 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 failures in both control-plane and data-plane operations.**

***Index Terms*— Network functions virtualization, LTE, IP multimedia subsystem, fault tolerance, and software defined networking.**

I. Introduction

MOBILE network operators are planning to support a number of multimedia applications into their network. These include voice and video over LTE (VoLTE/ViLTE), evolved Multimedia Broadcast/Multicast Service (eMBMS), virtual reality, interactive gaming, and many more. These multimedia applications being real-time have stringent end-to-end latency requirements and require guaranteed bit rate service from LTE network. LTE network that can only offer best effort service deploys IP Multimedia Subsystem (IMS) as a overlay architecture for precedence treatment of multimedia traffic. Figure 1 shows IMS being an overlay to LTE network handles multimedia signaling packets through multimedia servers, and multimedia data packets through multimedia gateway. It finally connects the source device with target multimedia application (telephony application in our example).

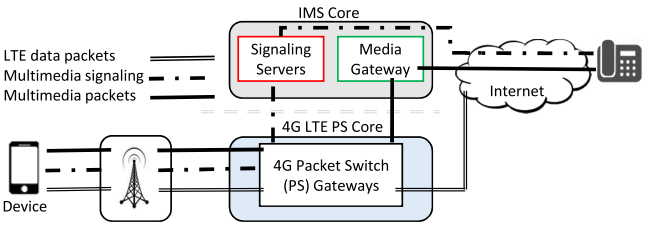

Fig. 1. IMS is an overlay to LTE network.

The ever-growing demands from numerous conventional and new generation multimedia applications have compelled network operators to look towards Network Functions Virtualization (NFV) to scale network services up and down quickly and to better align costs with network usage [1]. Service providers so far have only looked at performance [2], scalability and flexibility [3], and monetary aspects [4] of NFV, largely ignoring the impact of Network Function (NF) faults on service provisioning. IMS that provides multimedia services to LTE subscribers is the leading candidate for carrier network virtualization [1]; and requires immediate attention on its fault tolerance support in virtualized environment. To the best of our knowledge, our work is the first to disclose weak fault tolerance in existing vIMS (e.g. on-going voice and other multimedia traffic); and show how it jeopardizes device service operations. We addresses these issues through domain specific knowledge. IMS-NFV replaces dedicated IMS NFs implementation over proprietary hardwares with software running on commercial commodity servers. When IMS implementation is moved from traditional carrier-grade boxes to general-purpose boxes, vIMS needs to rely on IMS protocol-level and virtualization platform-level fault tolerance mechanisms [5], [6] to serve its subscribers during faults. However, we reveal that both IMS protocol and virtualization platforms do not fulfill fault tolerance requirements stipulated by multimedia applications. The failure recovery in 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.

Manuscript received April 15, 2019; revised February 1, 2020; accepted March 2, 2020. Date of publication June 5, 2020; date of current version June 29, 2020. *(Corresponding author: Muhammad Taqi Raza.)*

Muhammad Taqi Raza is with the Department of Information Systems, The University of Arizona, Tucson, AZ 85721 USA (e-mail: taqi@email.arizona.edu).

Songwu Lu is with the Department of Computer Science, University of California–Los Angeles, Los Angeles, CA 90095 USA (e-mail: slu@cs.ucla.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2020.2999686

A. Goals

We aim to achieve same level of fault tolerance in vIMS as provided by carrier-grade IMS platform, and that vIMS should continue serving existing and new service requests during faults. To do so, we provide session-level resilience to both control-plane and data-plane operations. Moreover, we want minimum changes in current IMS implementation that do not conflict with standardized IMS operations and recovery procedures.

B. Design

We have put forward a design to meet our goals. In the control-plane, the highlight of our design is to simplify fault tolerance procedure for device operations (e.g. voice call operation) by providing fault detection and fail-over procedures only to critical actions of IMS NFs. We argue that every IMS operation can be split into critical and non-critical actions. The fault tolerance should be provided for critical actions by letting non-critical actions (referred as provisional actions in this paper) to fail. Critical actions are those actions which play an important role in establishing multimedia operation; whereas, provisional actions are auxiliary actions whose job is to keep source and target clients informed regarding progress of multimedia operation. Provisional actions are not related to multimedia control-plane initialization, establishing and connection; and thus their failure do not impact execution of multimedia operation. To take an example, a voice call operation can be divided into *calling*, *progressing*, *proceeding*, *connected* and *terminated* actions. *Calling* and *connected* actions are critical as they deal with voice call control-plane setup and data-plane initialization; whereas, rest of the actions are non-critical as they only provide progress of the call setup to the source device and do not play any role in voice call being successful or not.

Our key design idea is to enable every IMS NF to take charge of its directly associated neighbor when it fails. Our fail-over procedure is supported on one-leg, i.e. in-service NF takes charge of out-of-service IMS NF, and resumes IMS operation by replaying the action at which the fault occurred. We have achieved this by applying atomic properties on critical actions, decoupling non-overlapping actions and partitioning likewise actions into their respective action groups. To replay the failed actions, the in-service IMS NF should have kept track of device session information located at out-of-service IMS NF before failure. Our design achieves this by letting every IMS NF to replicate its session information to all of its associated neighbors. These sessions are replicated through piggybacking over a request-response messages exchange when two neighboring NFs communicate with each other.

To address data-plane failure case, our design partitions IMS services based on their characteristics and traffic requirements. It exploits LTE specific features and creates different paths between device and IMS network, where each path carries specific IMS service. At IMS media gateway, each path terminates into separate virtualized Network Function (VNF) instances. These VNFs facilitate each others traffic during faults. When all VNFs stop serving, we efficiently transfer data-plane traffic to standby IMS media gateway server.

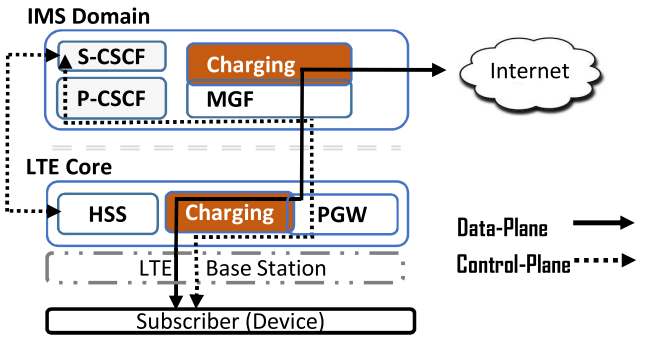

Fig. 2. IMS architecture: an overview.

C. Results

We implement our design and gather results from our OpenIMS [7] implementation over Openstack [8]. We modify OpenIMS implementation for efficient packet processing, critical actions isolation from non-critical ones, operation execution through finite state machine, and run time module reconfigurations to handle fail-over and fail-back procedures. We analyze our fault tolerance approach when failure occurs during (a) device registration, (b) multimedia service request, and (c) data-plane traffic. Our results show that our system: (1) reduces recovery time by up to 20X compared to current vIMS implementation, and (2) controls signaling storm that occurs during failure.

II. Background

IMS runs on top of LTE which provides 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 services. IMS uses Internet protocols and brings multiple media, multiple point of access and multiple modes of communication into a single network, enabling simultaneous voice and multimedia services for end users [9].

A. IMS Architecture

IMS operations are categorized into control-plane and data-plane operations, as shown in Figure 2.

B. Control-Plane

supports call sessions control through Call Session Control Function (CSCF) entities. The CSCF performs all the signaling operations, manages Session Initiation Protocol (SIP) sessions and coordinates with other NFs for session control, service control and resource allocation. It consists of two main NFs: the Proxy-CSCF (P-CSCF) and Serving-CSCF (S-CSCF). LTE device (IMS client) first camps on LTE base station and

registers with LTE core network. It then also registers with IMS network and initiates IMS signaling over IMS control-plane. The P-CSCF is an access point for IMS and acts as a SIP proxy for all the user equipments. P-CSCF simply forwards all traffic to S-CSCF. S-CSCF is the core of the IMS and it is the point of control within the network that enables operators to control the entire service delivery process and all the sessions. S-CSCF has knowledge of all the services subscribed by the users, by downloading from the Home Subscriber Server (HSS).[1]

C. 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 Packet Data Network Gateway - PGW) with IMS domain for multimedia service and converts between different transmission and coding techniques. Moreover, it employs monitoring schemes to determine policy and charging rules in real-time at both IMS and LTE.

III. High Service Availability in Carrier Grade IMS

End-to-End Service Provisioning: One goal of network operators is to provide all-time service access to their subscribers. In end-to-end service access, source (originating service-request device) communicates with radio network that forwards packets to LTE core network, which then redirects these packets to IMS for delivering them to destination device. Therefore, in order to provide end-to-end service, all three networks (radio, LTE core, and IMS) must function even during faults. Conventionally, these three networks employ sufficient mechanisms at their hardware and software to tolerate faults and provide high availability of network resources.

High service resilience in LTE operations is achieved through protocol level measures and vendor specific platforms. We now describe existing fault tolerance mechanisms at LTE radio network, LTE core network and IMS network.

A. Fault Tolerance in LTE Radio Network

LTE base station employs various mechanisms to tolerate faults. Radio transmission failure is addressed at LTE protocol level which provides retransmission of lost packets (i.e. HARQ procedure) [10]. Cell connectivity failure is taken care of by switching to a better cell when device's signal strength starts getting weak [10], while maintaining data flows through tunneling. To address network capacity failure, network vendors use different algorithms to switch load between cells and provision radio resources as per service type.

[1]HSS is a database that contains all subscribers' data like the services they are allowed to access, the network in which they are granted to roam and the information about the location of these subscribers. Another important function of the HSS is to provide the encryption and authentication keys of users.

B. Fault Tolerance in LTE Core and IMS Networks

Vendor specific IMS and LTE-core platforms provide two layers of defense at their hardware and software.

1) Hardware Fault Tolerance: Purpose-built hardware platforms have been developed that can tolerate faults. They continue to provide the required functioning despite occasional internal components and modules failures, either transient or permanent. Examples of hardware platforms are Ericsson's Blade Systems (EBS) [11], Alcatel-Lucent's Element Management System (EMS) [12], and Huawaei's ATCA [13]. These platforms use internal redundant hardware modules, provide NF availability during failures, and do maintenance at any time without disturbing traffic.

2) Software Fault Tolerance: IMS equipment vendors provide strong coupling between their software and hardware. Software fault tolerance is achieved by software design that ensures redundancy, both for error detection and error recovery. As the system operates, functional checks are made on the acceptability of the results generated by each piece of software component. Software platforms include Ericsson's ERLANG [11], Alcatel-Lucent's NVP [14], and Huawaei's Fusion [15] that use various software techniques for scalable real-time systems with requirements on concurrency, distribution and fault tolerance.

C. Fault Tolerance in IMS Protocol

Our study reveals that IMS protocol does not provide fault tolerance mechanisms, instead it performs failure recovery by rebooting or switching to redundant NF [16]; and takes tens of seconds to get back into service.

1) On S-CSCF Failure: When device registers with IMS network at first time, its SIP proxies (including P-CSCF address), contact information, authentication information, and others are backed-up at HSS. If any of the above data is modified, S-CSCF updates the record at HSS. On S-CSCF failure, the 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, HSS re-assigns another S-CSCF and performs failure recovery procedure by restoring already registered device information and reconfiguring the connection with P-CSCF.

2) On P-CSCF Failure: P-CSCF failure is detected by PGW that informs user equipment by sending an address of new P-CSCF. When the device receives new P-CSCF address, it declares previous P-CSCF as unavailable and sends IMS registration request towards new P-CSCF.

3) On MGF Failure: 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 devices. The path between MGF and PGW is declared out of service. This failure is also propagated to S-CSCF which in turn prevents registered devices to use media service. Thereafter, PGW reselects a new MGF with the help of media gateway control function. Once new MGF becomes operational, all media-traffic is forwarded to new MGF.

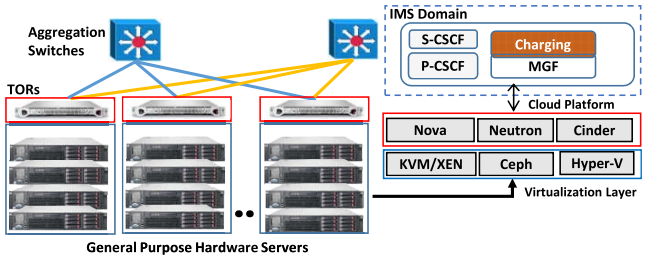

Fig. 3. NFV implementation of IMS.

In short, fault recovery procedures provided by IMS protocol are lengthy and are not triggered by carrier grade IMS platforms. These platform apply dedicated mechanisms at their dedicated NF boxes to provide session-level resilience.

IV. State of the Art NFV Implementation

vIMS has already been embraced by a number of operators, such as SK telecom in Korea [17] and Telefonica in Europe [18]. In other countries, vIMS is currently being rolled out such as, Sprint telecom in the USA [19], and Spark telecom in New Zealand [20]. Yet others are considering to deploy vIMS in near future, such as Telstra from Australia [21] and Telecom Argentina [22]. vIMS has potential benefits: (1) NFV based IMS can achieve high scalability and high flexibility to quickly scale services up and down [23], and (2) reduce network expenses (both capital expenditures (CAPEX) and operational expenditures (OPEX)) [24].

Figure 3 shows NFV implementation of IMS in virtualized Data Center Network (DC) [25]. The cloud platform (e.g. OpenStack [5]) acts as a manager which provides virtualization (e.g. KVM/XEN [26]) of hardware resources to IMS NFs. OpenStack is running a number of service such as Nova, Neutron and Cinder [27] to provide compute, networking and storage, respectively. Neutron service provides Bare-metal provisioning [28] to IMS NFs that decreases call-setup time and meets multimedia quality of service (QoS) requirements during traffic load. Cinder service allows multiple IMS NFs to access common subscribers profile information and allows Nova to do computation without requiring any knowledge of where users storage is actually deployed. In short, vIMS meets subscribers service requirements by dynamically allocating common hardware resources.

A. Empirical Study

We conduct empirical study to compare IMS service provisioning between operational IMS network and our state of the art vIMS network implementation.

1) Fault-Tolerance in Operational IMS Network: In order to artificially create faults in operational IMS network, we drop the incoming IMS network packets at source device by injecting malicious packets into Voice over LTE (VoLTE) downlink (DL) voice bearer. We observe that SIP client at device makes several retries at an interval of 500 milliseconds to establish the connection with IMS network. This confirms the behaviour of VoLTE device if such faults would actually have occurred in operational IMS network. After 5 seconds (10 number of retries), device aborts the IMS operation and escalates the call failure to LTE telephony module. Thereafter, device switches to 3G network and re-initiates the call operation over 3G network.

2) Fault-Tolerance in State of the Art vIMS Network: We setup the state of the art vIMS network on OpenStack [8], a widely-used cloud computing platform, and experiment by causing failure in each vIMS NF. Implementation details are provided in Section VI. State of the art vIMS implementation employs two separate fault tolerance mechanisms, provided by IMS protocol [16] and OpenStack [5].

a) Failure recovery in IMS Protocol: Although IMS protocol provides a series of fault recovery procedures for IMS control-plane, we find that these procedure takes tens of seconds to recover from control-plane failure, which is very high for operational IMS network. To investigate how IMS fault recovery procedures work, we perform experiments on state of the art vIMS by turning off the fault-tolerance mechanism provided by OpenStack. We explain this through S-CSCF failure example, as shown in Figure 4a. When S-CSCF stops responding, P-CSCF takes 30 seconds to detect the failure. It then informs HSS about S-CSCF failure that prepares a new S-CSCF NF, and provides S-CSCF address to P-CSCF. In total it takes around 31 seconds for IMS protocol to get back into function. Moreover, we find that IMS protocol hides failure propagation to device by setting default client timeout value to 32 seconds (as shown Figure in 4c).

In short, lack of fault-tolerance support at IMS protocol level requires fault-tolerance at cloud platform that should be exploited for high availability.

b) Failure recovery in cloud-platform: We repeat above experiment by enabling cloud platform recovery procedures. The default cluster-detecting system provided by OpenStack monitors the availability of a service and makes the service available again if fail-stop failure occurs. As shown in Figure 4b, the default heartbeat interval for failure detection is 2 seconds, and the maximum number of ticks in a round are 5. In addition, the time to prepare a backup VM for restoring and recovering the service is about 8 seconds. It takes about 18 seconds for OpenStack to bring back S-CSCF into service. In short, current cloud systems, termed as Infrastructure as a Service (IaaS), do not provide instance-level fault tolerance for session-resilience during faults.

Fault tolerance is not adjusting timer values: It can be argued that choosing smaller timeout values at IMS protocol and OpenStack should solve the problem. However, adjusting smaller timeout timer values cause ping-pong between fail-over and fail-back procedures with greater number of false positives on fault detection [29].

From this empirical study, we can conclude that vIMS cannot deliver same level of fault tolerance as compared to carrier grade IMS platform. Our conclusion is also strengthened by the concerns shown by telecom giants, i.e. Verizon [30] and Vodafone [31], towards NFV based LTE service provisioning.

B. Impacts

We discuss how weak fault tolerance support can impact multimedia service provisioning.

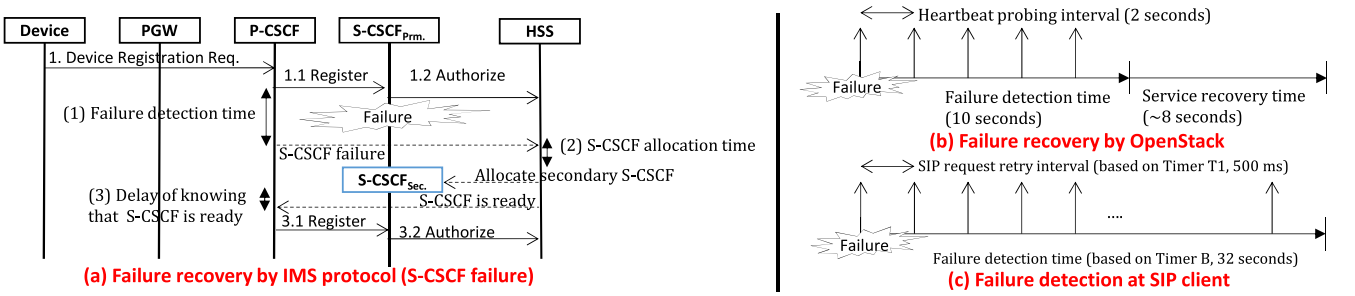

Fig. 4. Failure recovery procedure at IMS protocol, OpenStack, and IMS SIP client.

1) Impact From S-CSCF Failure: There are several issues with S-CSCF restoration procedure after failure. First, S-CSCF failure is propagated to device which is against the philosophy of fault tolerance: system failures should always be hidden from end devices [32]. Second, on S-CSCF failure, IMS service is temporarily suspended. Such failure recovery procedure is not sufficient for crucial telephony service that requires high reliability. Third, IMS heavily relies on LTE core network element, i.e. HSS, for S-CSCF recovery procedure. This is because IMS is an overly network on LTE service which is provided by third party service providers. Therefore, LTE network operators provide access of private user information to third party service provider that can risk user data breach [33]. Fourth, instead of storing backup data on redundant S-CSCF, it is stored at HSS. As a result, HSS can become a single point of failure for the stored backup of all devices.

2) Impact From P-CSCF Failure: 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, IMS service remains unavailable to users during P-CSCF failure. 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 that P-CSCF failure has domino effect to data-plane traffic, even though P-CSCF does not play any role in data-plane communication. Fourth, failure recovery from all users happen at almost the same time (where users send re-connection request to IMS), which brings signaling storm and can potentially knock alternative IMS NF out as well.

3) Impact From MGF Failure: 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 served by IMS. Further more, lack of data-plane fault tolerance also results in incorrect billing information, and loss of policy information at charging function.

V. Design

Overview: Figure 5 gives an overview of our design. To provide control-plane fault tolerance, we propose that whenever a particular IMS NF fails, its neighboring IMS NF should take charge of that failed NF and continue serving the users. In other words, our design performs one-leg IMS operation during faults when in-service NF executes out-of-service NF's tasks. We employ Finite State Machine (FSM) to quickly detect fail-stop failure. However, we face two challenges to resume one-leg IMS operation after failure: (1) both NFs playing different roles cannot access device session information stored on each other; and (2) failure can occur without raising any alarm that makes it impossible for IMS operation to resume if complete up-to-date session information is not stored somewhere else. To address these two challenges, we propose that both NFs must piggyback each others' on-going device session information while communicating in request-response loop before faults. To reduce piggyback message overhead, we partition IMS operation into critical and provisional actions. We provide fault tolerance to critical actions, while letting provisional actions to fail.

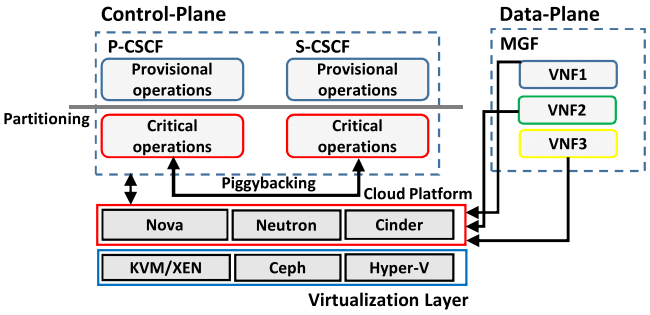

Fig. 5. Design overview.

For data-plane, we partition IMS services based on their characteristics. These decomposed services are handled by different MGF instances (VNFs) that are relocated to other VNF when their respective VNF instance fails.

Failure Model: In this paper, we consider fail-stop failure [34]. In IMS protocol design, IMS operations work properly until NF crashes or the link between IMS NFs are broken [35]. However, current IMS implementation does not distinguish between omission [36] and fail-stop failures. As a result, the NF failure is marked as fail-stop even when it does not respond to incoming/outgoing messages. We argue that omission failures are harbinger to fail-stop failure. Therefore, we use omission failure to build fail-stop failure model. In our proposed IMS design, whenever we detect send/receive omission failure, we (a) relocate ongoing session to running NF processes, and (b) start recovery procedure of failed server.

We consider four different failure scenarios. We believe that failure can occur (1) during device registration procedure

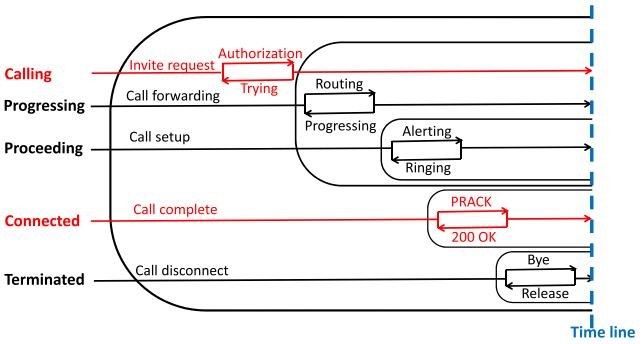

Fig. 6. Atomic actions for voice call.

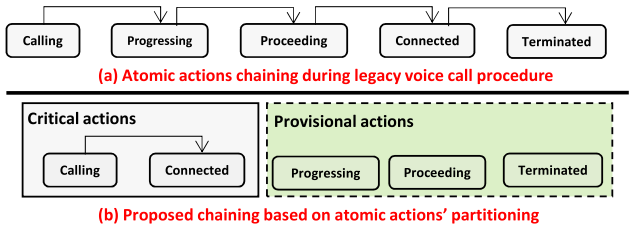

Fig. 7. Partition actions in IMS operations into critical and provisional action groups.

with IMS network, (2) when device initiates service request to send/receive multimedia service, and (3) during multimedia data-plane traffic flow to/from device. Lastly, we consider that (4) IMS failure creates a ripple effect for other transitory failures in LTE Network. For example, delays in device handover procedure can make IMS NFs wrongly assume device failure.

A. Control-Plane Fault Tolerance

To ensure continued operations during faults, IMS control-plane operations must withstand failures. IMS NFs keep check on each others' operations and protect critical operations from failures. In case of complete IMS black-out, our IMS standby server efficiently coordinate with HSS to retrieve users session records.

1) Design Preliminaries: We first present design preliminaries for our control-plane fault tolerance.

a) Decomposing IMS operations into atomic actions: We decouple control-plane operations using the concept of atomic actions [37]. To execute one operation, IMS system performs a number of actions that either generates further action(s) or execute a number of steps to conclude that action. For example, during voice call operation (as shown by Figure 6), *calling action* sends *invite request*, authorizes the device and acknowledges the *invite request* by sending *trying* message. Calling action requires *progressing*, *proceeding*, *connected* and *terminated* actions to forward, set-up, complete, and finally disconnect the call, respectively. These all actions are necessary to successfully execute one call operation. We name these nested actions as "atomic actions" because they are carried out by different modules of IMS system [35]. In an example of voice call operation, the atomic action of IMS security module is to *authorize* the device from HSS, the atomic action for call forwarding module is to *forward* session information to the destination IMS system and set-up the call, whereas the atomic action for call connection module is to *connect* to media function for data-plane traffic. An atomic action could involve several steps where (1) one atomic action could rely on one or more actions to assist it; and (2) two or more actions could cooperate directly to execute a number of steps in a shared atomic action space. We show this procedure in Figure 6, in which rectangular arrows indicate request and response steps within an atomic action, whereas incomplete rounded rectangle represents atomic actions that are still in progress. Figure 6 illustrates that atomicity has to be regarded as relative rather than absolute, and atomic actions, by their very nature, cannot overlap.

b) Partitioning of atomic actions: Once we identify atomic actions for an IMS operation, we partition them into *critical* and *provisional* action groups. Such partitioning is logical where (1) dynamic recovery is performed on *critical* actions, but *provisional* actions are only protected from being timed-out, and (2) only critical actions' session-level information is sent to other IMS NF for session resilience that also reduces fault recovery overhead. We take an example of voice call operation in which multiple atomic actions are chained in a fixed order, as shown in Figure 7a. When voice call is initiated, each atomic action performs its task and transfers call control to next atomic action. For example *calling* action triggers *progressing* action that then forwards the call to target IMS system. After successful execution of *progressing* action, *proceeding* action alerts the user through ring-tone. However, *progressing* actions that rely on target IMS system to provide timely response may delay *proceeding* action due to queuing delays [38] at target IMS system. Such delay not only causes expiry of timer for *progressing* action, but also has a domino effect to other chained actions; resulting in call failure. We argue that delay in *progressing* and *proceeding* actions should not be the reason to abort the call, if such a delay is within the tolerance range of call operation. We call such actions *provisional* that are a part of an operation, can potentially fail an operation, yet they are not critical enough for the execution of the operation. Therefore, we partition the provisional actions from *critical* actions, as shown in Figure 7, tolerate the failure of *provisional* actions and block the affect of their failure to *critical* actions.

2) Replicating Ongoing Users Session Information Before Failure: We use design preliminaries to provide fault tolerance in vIMS. During normal vIMS operation, our design allows every NF to replicate its users session information of critical on-going actions at its directly associated neighboring IMS NF. To provide session-level resilience during failure, both neighboring IMS NFs should have real time access of device session information as the IMS operation proceeds. However, each NF keeps different session information that the other NF cannot access. To address this, we exploit the fact that both P-CSCF and S-CSCF communicating in a feedback loop of request and response can piggyback one NF device session to the other. The piggyback information is sent using optional IMS header fields. Note that we only replicate *critical* actions'

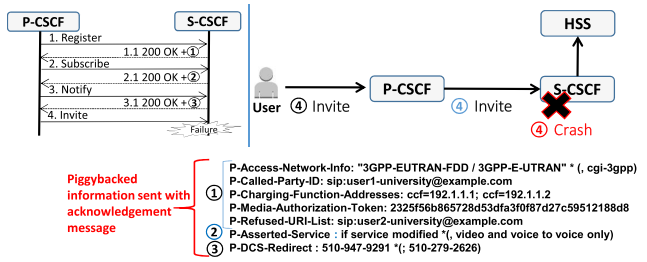

Fig. 8. On-going user session is piggybacked over acknowledgement messages.

session information that are vital in resuming device operation after failure. We describe this procedure by using device registration and voice call operations. As shown in Figure 8, device initiates registration action by sending *register* message to P-CSCF which then forwards it to S-CSCF. On receiving *register* message, S-CSCF creates device session that includes user identities, charging function address, and device authentication information. Contrary to conventional reply, where S-CSCF sends back an acknowledgement message *200 OK* to device, we propose piggybacking newly created device session information with this acknowledgement message. We encode each session value inside SIP Private Header (P-Header) fields. SIP P-Headers are specific to 3GPP technology (i.e. LTE, GSM and WCDMA) context and are used to correlate access network information across multiple IMS NFs [39]. Encoding and decoding of sessions into P-headers are achieved within 10 lines of C-code.

In short, we can replicate live session information from one IMS NF to other without introducing any changes to the IMS protocol.

3) Use Omission Failures to Detect Fail-Stop Failure: We argue that because P-CSCF and S-CSCF are directly connected, the control-plane message retry timeout 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 5xRTT with maximum 5 number of retries (which is our implementation choice). In case of no reply within 25xRTT, non-responding server'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 fail-over in 30xRTT.

a) Detecting IMS NF failure through finite state machine: We create FSM of *critical* actions to efficiently detect the fail-stop failure. We introduce few temporary states that keep track of different steps in an atomic action. We explain our proposed state transition diagram by using voice call operation which is running at P-CSCF. Figure 9b 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[2] fail-over 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 have progressed to *complete-pending* state.

Timer G[3] 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, it discards duplicate *invite* requests [40]. This proactive probing helps quickly recover from transitive faults when previous invite request(s) is(are) failed to be delivered to target device.

Tolerating transitory device failures: We also consider the case when response from originating device is delayed because of transitory radio failures (device may temporarily gets stuck in a loop, or device handover procedure may take long time). From our field tests during empirical study (Section IV-A), we find radio delays can go up to 3 seconds, as shown in Figure 10. Such delays mostly occur during device mobility when hard handover is performed [41]. Therefore, we tolerate delays up to 3 seconds from originating device (as shown in Figure9a).

4) Fail-Over Procedure: After detecting failure, we perform fail-over procedure when one IMS NF declares other NF out-of-service and takes charge of non-responding NF. Such recovery is only provided to critical atomic actions. We explain this procedure through S-CSCF NF failure. When S-CSCF does not respond within *30xRTT*, we deactivate the link between P-CSCF and S-CSCF (Link 1 shown in Figure 11), and activate the link between P-CSCF and P-CSCF's internal S-CSCF service.[4] Now P-CSCF forwards all the traffic to its internal S-CSCF components. New S-CSCF resumes the operation from the step at which failure occurred. Figure 8 describes a case, when S-CSCF fails on *Notify* message where P-CSCF responds with both *notify-request* and *notify-response*(through its internal S-CSCF service logic) messages.

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 faces arbitrary failure – not impacting the function of S-CSCF. We still define such case as a failure that impact user Quality of Service (QoS). However, we hand over to the actual S-CSCF function in fail-back procedure. Note, in order to avoid ping-pong effect between fail-over and fail-back procedures, fail-back procedure does not occur within certain time (30 minutes in our implementation) to fail-over.

We should mention that during control-plane fault tolerance procedure, we do not disturb any other IMS 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).

5) Fail-Back Procedure: Fail-back procedure starts when (1) minimum time (i.e 30 minutes in our implementation) has elapsed after fail-over, and (2) failed NF has recovered from

[2]We propose local RTT, measured between P-CSCF and S-CSCF NFs.

[3]We propose timer G, calculated based on global RTT – measured between source IMS and destination IMS systems.

[4]S-CSCF execution logic implemented within P-CSCF.

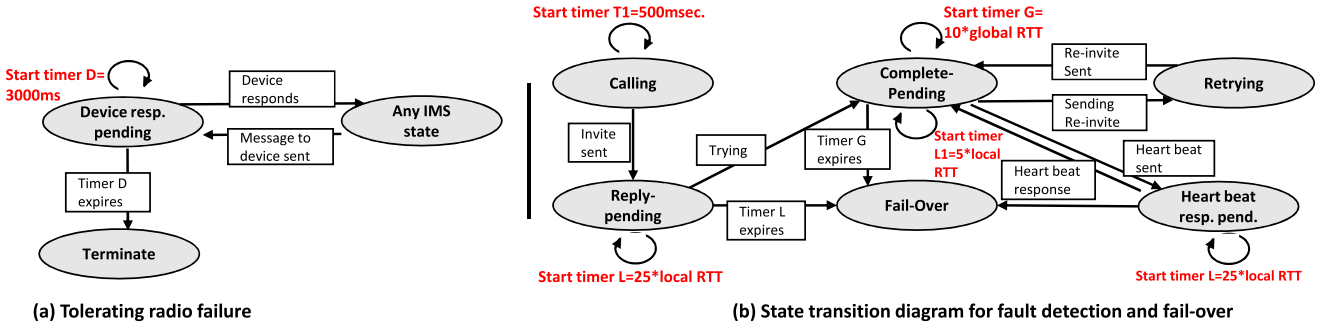

Fig. 9. State transition diagram tolerating transitory device delays and detecting failure in IMS system.

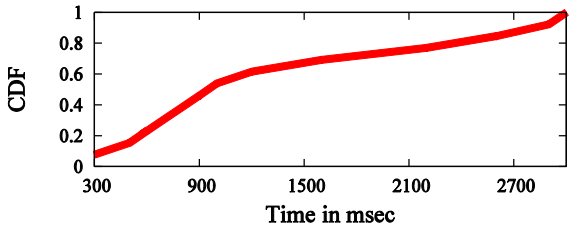

Fig. 10. Transitory LTE delays.

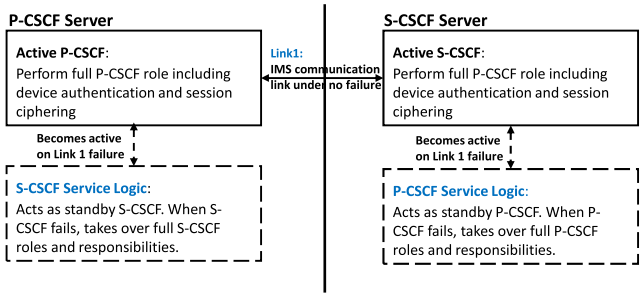

Fig. 11. Recovering from control-plane failure.

failure either through IMS protocol or cloud platform failure recovery procedure.

Smooth transition: During fail-back procedure, in-service NF, executing both P-CSCF and S-CSCF functions, 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 failure. In this migration procedure, we only send device identities by using P-Called-Party-ID of P-Header field. 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 [42]. 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 signaling load of active users.

6) Session Resilience When Complete IMS System Fails: We also anticipate rare scenario when both S-CSCF and P-CSCF fails at the same time. In fact, this is reasonable assumption where failure in one rack [43] or in particular portion of DC can knock whole IMS out. In such a scenario, we propose using standby IMS server (with an implementation of both S-CSCF and P-CSCF NFs).

a) Failure detection procedure: The failure is detected by a router that diverts all the traffic towards standby IMS server. The router monitors both S-CSCF and P-CSCF by sending probe packets. It declares IMS system failure when both NFs do not reply to 5 consecutive probing packets sent with an interval of 500 milliseconds.

b) Fail-over using semaphore: When all IMS NFs stop responding, it is not possible to replicate on-going users session information to standby IMS unit, therefore standby server needs to fetch users records from HSS. However, sequential read/write operations pose a bottleneck when 1000s of users records are to be fetched from HSS. Read/write operations can go up to tens of seconds to retrieve session information for all users [44]. To address this issue, we propose coordination between HSS and standby IMS server using semaphore [45]. When vIMS system fails, standby IMS server signals HSS about the failure. HSS populates all users – registered with failed IMS system – record to shared database where standby IMS server retrieves users session information and replays the device operation (e.g. voice call).

In our implementation, we find that recovery from IMS blackout (when all IMS NFs stop responding) can take up to 10 seconds in worst case, which is twice the tolerant range used in operational IMS network. The higher recovery time is caused when IMS failure occurs and control-plane operations are about to conclude. After fetching the records from HSS, the standby IMS server needs to contact target IMS to reconnect the broken service link. To address this issue, standby IMS sends *re-invite* message to source device requesting it to "hold-on[5]" meanwhile it is re-establishing the device connection with target IMS.

c) Fail-back procedure: Once vIMS system is back in service, it retrieves passive, but registered, user's information from standby database server. Moreover, new registration requests are also forwarded to vIMS system. In short,

[5]In operational IMS working, the *re-invite* message is sent when one of the calling party puts the call on-hold. We use re-invite message in our advantage and avoid call drop even in extreme cases.

active SIP sessions are still handled by standby IMS server, whereas future SIP requests are forwarded to recovered IMS system.

B. Data Plane Fault-Tolerance

Providing session resilience to data-plane traffic is challenging because: (1) the real-time traffic does not employ any reliable mechanisms (e.g. re-transmission of lost packets). This causes complete service interruption or even failure (e.g. call drop). (2) Detecting MGF failure at PGW is slow that renders data-plane unavailable even to new multimedia service requests.

To address these challenges, we propose multimedia service partitioning at MGF and employ two mechanisms to recover from (1) particular service failure and (2) complete MGF failure.

1) Data Plane Service Partitioning: Different multimedia services have different characteristics and requirements. Real time services, such as voice (e.g. VoLTE) and video conferencing call (e.g. Skype video call) require guaranteed packets delays (100ms and 150ms, respectively [46]) to serve users. Non-real time applications, such as playback video (e.g. youtube) do not require guaranteed bit rate and can relax packet delay up to 300ms (which is 3 times higher than voice service). However, current IMS implementation in operational LTE network treats all types of IMS services same. Our IMS experiments with US network operators reveal that at the time of device registration, LTE core network creates two bearers and assign two different IPv6 addresses (i.e. one for normal data traffic, and the other for IMS applications). Irrespective of IMS service type, device uses same IMS bearer IPv6 address to forward IMS packets to PGW that eventually forwards to MGF.

In this paper, we propose partitioning of data-plane services based on traffic characteristics. Such partitioning is done between device and MGF. At first step, PGW assigns 3 dedicated bearers each for voice, video conferencing and playback video services of IMS. These bearers remain active until device is registered with LTE network. At second step, we partition one MGF into three virtualized network functions (VNFs), each for voice, video conferencing, and playback video services. This enables us to separate IMS data-traffic between device and MGF, as shown in Figure 12. The advantage of such partitioning is three fold, (1) by using same hardware resources, different applications can stipulate different requirements, (2) partitioning helps restricting failures propagation across different IMS services, and (3) PGW can use different mechanisms to promptly detect VNF failures and adopt appropriate measures to restore service.

2) Failure Handling: In order to avoid unnecessary probing, PGW uses downlink (DL) user traffic as MGF's VNFs monitoring. In case, the PGW does not receive expected DL packet within service tolerant time (i.e. 100 ms for voice, and 300 ms for video), it declares that VNF(s) as failure-prone. During failure-prone status, PGW uses frequent probing to discover whether packet delays were because of VNF failure or due to destination network delay/failure.

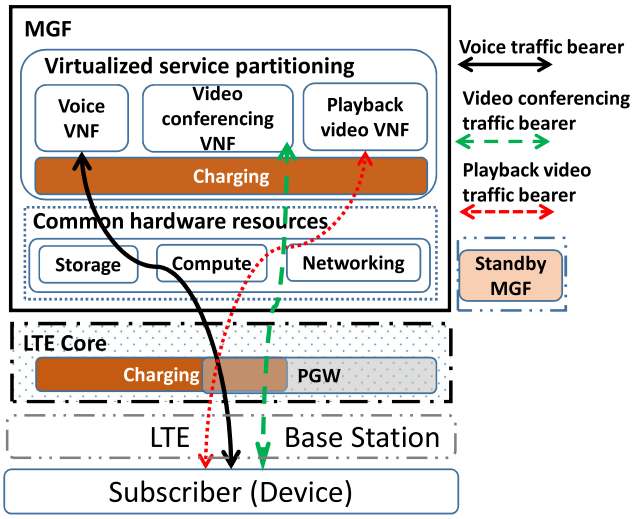

Fig. 12. Data-plane partitioning into virtualized services based on their characteristics.

We distinguish between few VNFs (one or more but not all) failure case, and all VNFs failure case (when MGF server fails).

At least one VNF is active: PGW monitors failure-prone MGF's VNF(s) through heart-beat probing. If VNF does not reply to heart-beat within a configurable amount of time (5xRTT in our implementation), PGW declares that VNF out-of-service. The data traffic from out-of-service VNF is then transferred to one of the other two in-service VNFs. Because target VNF has its own coding scheme as per its traffic requirements, the transferred VNF traffic also uses the target VNF's coding technique. However, it is possible that transferred VNF traffic (say video conferencing call) was using higher coding technique compared the one available at target VNF (handling playback video). Such change of coding technique (from higher to lower) only affects the quality of service (QoS) of the transferred traffic. We believe, this is an acceptable tradeoff for achieving service-resilience at the cost of QoS during faults.

When all VNFs fail – MGF server failure case: When all VNFs do not reply to PGW heart-beat message within a configurable amount of time (5xRTT in our implementation), we declare that MGF out-of service. The PGW select standby MGF to restore user data traffic. We quickly detect MGF failure because all VNFs stop responding to PGW heart-beat probing packets at the same time. This reduces total fail-over time that only brings user service jitter, compared to traditional MGF recovery that terminates data service [47].

Meanwhile, PGW keeps probing out-of-service VNF to detect its recovery. When out-of-service VNF starts responding to PGW heart-beat packets, its status is changed to in-service and PGW steers traffic back to recovered VNF.

VI. Implementation

In our implementation, we use open source IMS platform (OpenIMS [7]) and open source cloud operating system (OpenStack [8]) to implement the functionalities of IMS

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

A. State of the Art NFV Implementation of IMS

OpenIMS has coupled all IMS NFs by implementing them over single virtual machine (e.g. VMware [48]) that provides optimal performance when hundreds of users are accessing IMS network at the same time. For NFV 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 state of the art vIMS implementation. We also provide 1:1 redundant copy of IMS NFs to achieve minimum industry requirement for NFV [49]. We use default timers as specified by IMS and OpenStack documents [50], [51]. We consider this implementation as base-line vIMS with which we compare our design.

B. Implementation of Proposed vIMS

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

1) Call Session Control Functions (CSCFs): Our design supports one-leg operation where both S-CSCF and P-CSCF are implemented as one NF. To achieve functionalities of both NFs within single NF, we modify SIP Express Router (SER) [52] of OpenIMS. SER handles all SIP registration, SIP service requests, and directs their signaling to P-CSCF and S-CSCF functional modules (which are located at same NF). SER allows us to optimize performance by not performing redundant tasks (such as manipulating SIP and P-Header fields) if one of IMS NFs has already processed it.

2) Piggybacking: To reduce piggybacking processing delay, we first convert single threaded packet processing module into multi-threaded module. We leverage multi-threaded processing to construct the actual packet header and supplementary piggybacked information using P-Header at the same time. Because we are dealing with real-time session, all active users information has already been pulled by OpenIMS into memory – which helps us quickly fill-in P-Header fields.

3) Atomic Actions: In order to implement atomic actions, we first break the dependency cycle of different actions and then partition them. OpenIMS provides basic IMS implementation, but it does not used modular approach in implementing different actions for IMS operations. We modify OpenIMS source code to convert these actions into modular C functions. We achieve transition of atomic actions when one functional module calls other functional module in a chain of IMS actions. We skip provisional actions and do not exchange their states using P-Header fields.

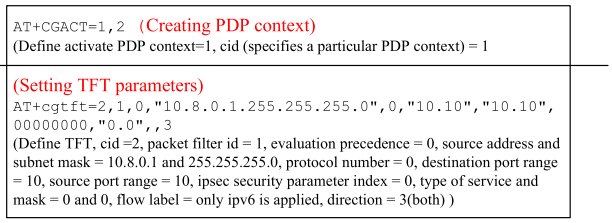

Fig. 13. Creating dedicated bearers from LTE device.

4) Finite State Machine: In FSM implementation an operation must start from an initial state and transit to another accepted state. To achieve this, we create FSM transition table in each NF that transits from a given state to a new state when either an atomic action has progressed or its guard timer has expired. By doing so, the proposed FSM only executes on necessary functional module.

5) Fail-Over and Fail-Back Procedures: To successfully execute fail-over and fail-back procedures, we are required to immediately resume IMS operation from the atomic action at which 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. We implement two separate modules for fail-over and fail-back procedures. When failure occurs, fail-over module intercepts the failure and retrieves last stored atomic action and related session information from hash table. Then fail-over modules updates the network configurations at incoming and outgoing interfaces and contacts destination IMS/source device using same user identities. It also applies filters to distinguish whether service requests and registration requests are coming from existing subscribers or new ones. Our fail-back module is relatively simple that only gets activated when out-of-service NF is back to service. After that, it intercepts all the traffic going to fail-over module and decides whether traffic should be processed locally or forwarded to recovered NF.

6) Data-Plane Service Partitioning: We create three VNFs of MGF. Because our implementation does not include LTE core network, therefore, our client cannot create three separate data-plane bearers at device. To address this we distinguish between service flows at virtualization layer of OpenStack before sending messages to appropriate VNF.

To show our proposal is technically feasible in operational LTE network, we create three dedicated bearers (using AT commands [53]) from operational LTE device to LTE core network. Figure 13 shows the snapshots of this procedure.

VII. Evaluation

A. Session Resilience During Faults

We consider session resilience when NFs stop responding during device (1) registration procedure, (2) multimedia service request, and (3) multimedia data-plane traffic flow. On control-plane we consider fail-stop failure at: (a) P-CSCF, (b) S-CSCF, (c) both P-CSCF and S-CSCF at same time, and (d) MGF.

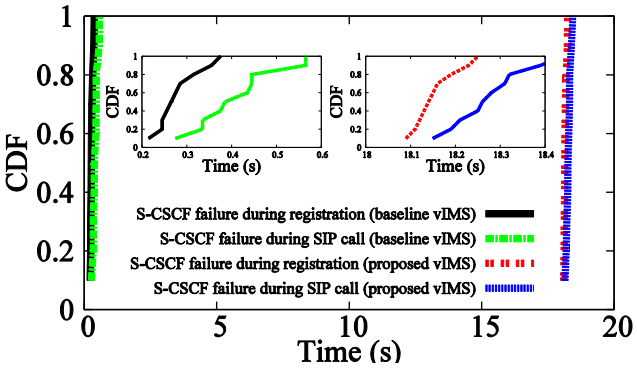

Fig. 14. Service recovery time after S-CSCF fail-stop failure: comparing proposed-vIMS with baseline-vIMS.

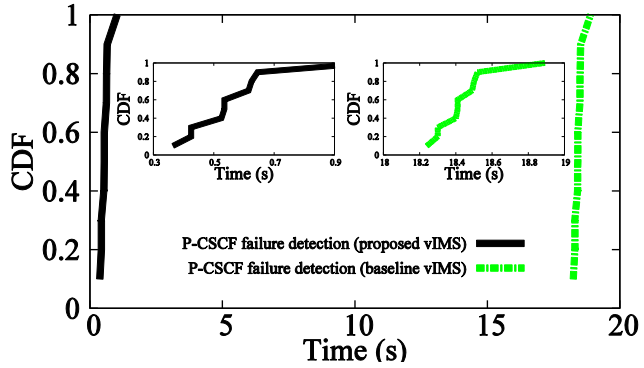

Fig. 15. Service recovery time after P-CSCF fail-stop failure: comparing proposed-vIMS with baseline-vIMS.

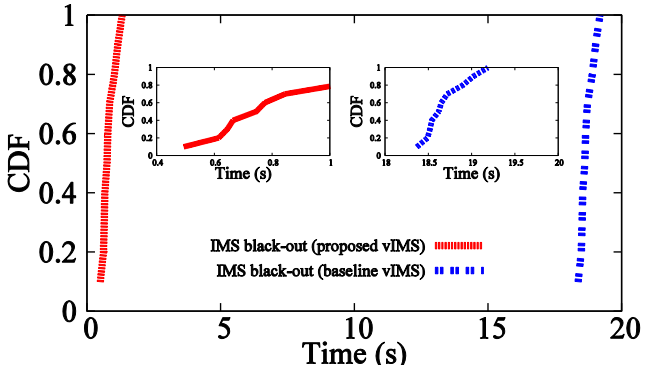

Fig. 16. Service recovery time when both P-CSCF and S-CSCF stop responding: comparing proposed-vIMS with baseline-vIMS.

B. How Proposed vIMS Achieves Control-Plane Resilience?

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 14, Figure 15 and Figure 16 show experimental results along with enlargements of critical regions. Irrespective of P-CSCF, S-CSCF or complete IMS crash, the control-plane operation aborts in 5 seconds (in accordance to operational IMS network, we set timeout value of 5 seconds at device). 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, proposed vIMS takes about 500ms in case of S-CSCF, 1000ms for both P-CSCF and IMS blackout case to recover from failure. We observe two different recovery phenomena. First, when S-CSCF crashes, the recovery is made at P-CSCF that successfully resumes the failed S-CSCF operation. But when P-CSCF takes more than 500ms to perform recovery, the first timeout happens at device. On time-out, device retries the operation and P-CSCF successfully executes the operation on one-leg. Second, when P-CSCF crashes or IMS blackout happens, we always observe device experiencing time-out (although 40% of the time S-CSCF starts one-leg operation within 500ms, as shown in figure 15). This is because S-CSCF needs to re-establish the IPsec tunnel with 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 S-CSCF.

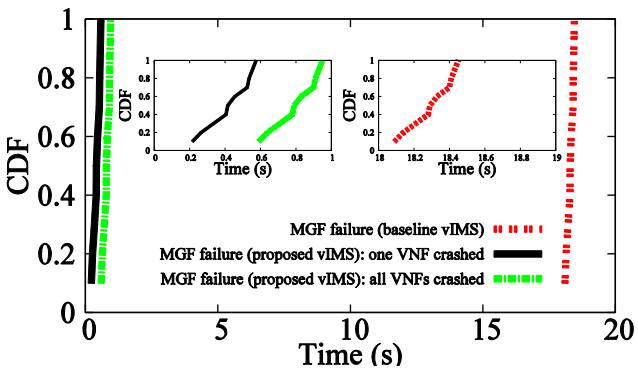

Fig. 17. Service recovery time after MGF fail-stop failure: comparing proposed-vIMS with baseline-vIMS.

C. How Proposed vIMS Provide Data Traffic Continuity?

When MGF crashes in base-line vIMS, the data-service aborts and user remains out of service for 18 seconds, as shown in Figure 17. In contrast, proposed vIMS transfers ongoing data-plane traffic from serving VNF to its neighboring VNF, when serving VNF stops responding. The fail-over happens within 500ms and device observes voice jitters, but keeping its data-plane connection intact during faults. Because we are monitoring all VNFs at same time, we detect MGF server failure as quickly as we detect single VNF failure. But our implementation takes upto 500ms more to tunnel traffic to standby MGF. During failure recovery period, we observe voice-mute upto 2 seconds because all of incoming/outgoing packets are lost. We believe unavailability of service for a couple of seconds in extreme MGF failure scenario is acceptable.

D. Controlling Domino Effects of Failure

We discover vIMS failure can potentially gives birth to signaling storm and inconsistent data charging at IMS network.

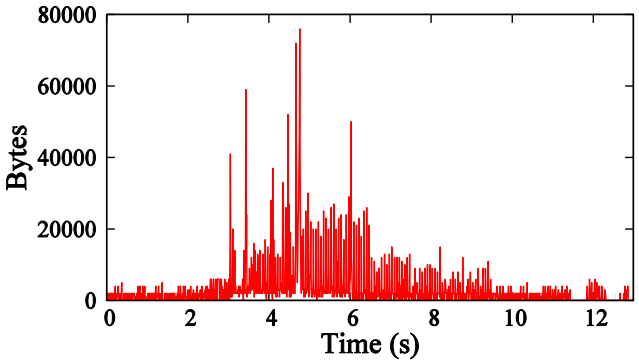

Fig. 18. Base-line vIMS causes signaling storm on P-CSCF failure (when failure is detected by PGW).

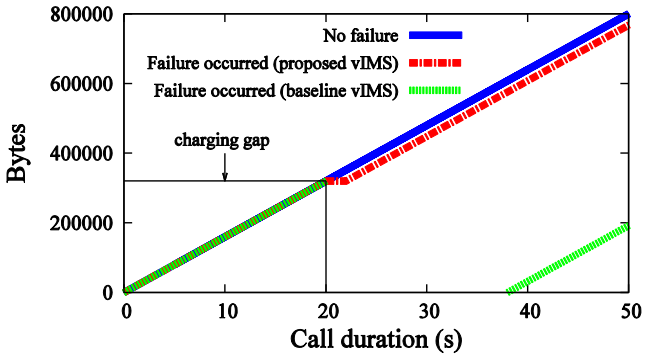

Fig. 19. Subscriber charging information gets lost on data-plane failure in base-line vIMS – causes charging-gap phenomena occurs.

E. How Proposed vIMS Avoids Signaling Storms?

We discover that vIMS failure can cause signaling storm in which all registered users start sending re-registration requests towards IMS. Such signaling storm can potentially knock already recovered server out as well.

LTE core network considering IMS as its overlay network monitors its availability. From LTE network point of view, P-CSCF being an entry point to IMS network is crucial to multimedia service availability. Therefore, in case of P-CSCF failure, PGW informs all devices connected to non-responding P-CSCF to perform IMS registration procedure with new P-CSCF. This potentially leads thousands of IMS subscribers to send re-connect request, within short interval of time, towards IMS network – causing a signaling storm. Both IMS NFs process the requests coming from thousands of subscribers and exchange further control-plane signaling messages with these devices.

To access damage caused by signaling storm, we send registration requests from 1000 of devices which are virtually connected with our OpenIMS. Figure 18 shows high signaling messages exchange that can last for few seconds before system operations return to normal. Note that, our proposed vIMS does not cause any signaling storm where our quick failure recovery procedure does not let PGW to take action on P-CSCF failure.

F. How Proposed vIMS Avoids Charging Gap?

The charging function is part of MGF that controls billing of multimedia usage by subscribers. When MGF fails during device data-plane operation, all the charging information is lost and subscriber is not billed for services it has used before failure. As shown in Figure 19, MGF stops responding at 20sec into voice call. The recovery takes in 18 seconds in case of baseline vIMS and less than 2 seconds in case of proposed vIMS. After recovery, baseline vIMS starts charging from zero. However, proposed vIMS efficiently coordinate with PGW charging function (we implement it as a replica of IMS charging function) and retrieves subscriber charging profile. The only missing amount of charging in proposed vIMS is the time to fail-over (which is less than 2 second). We do not observe any charging gap in case of VNF failure.

VIII. Related Work

Our work is in contrast with recent efforts on vIMS, LTE–NFV and other NFV applications' fault tolerance space.

Research on vIMS has recently received significant attention from both academia and industry. [54] discovers that different modules in vIMS create a feedback loop that causes failures and introduces latencies. [55] explains that NFV of IMS cannot meet industry requirements on high availability because of control plane failures. It introduces the concept of software modules that are fully connected and transition among each other to handle failures. In contrast, our work does not introduce any redundancy and optimizes the control plane failure recovery by partitioning into critical and provisional actions. Moreover, we also discuss data-plane fault tolerance. [56], [57] discuss that fault tolerance and security are two major challenges that IMS face in public cloud. They do not provide concrete solution to address failure recovery for IMS. Other works [58], [59] discuss IMS performance over NFV. [60] provides dynamic resource allocation algorithm for vIMS. [56] enhances vIMS features for M2M. But all these efforts do not discuss fault tolerance aspects of NFV-IMS.

LTE—NFV reliability related works include [61], [62] [63], [64], and [65]. [61] re-designs LTE core architecture for public cloud deployment and guarantees reliable LTE operations. The main idea of [61] is to break LTE core functionality into stateless and stateful components. The stateful information is stored into highly reliable storage that achieves the high availability. Different to [61], in this work, we provide the notion of critical and non-critical actions. Our work does not require highly reliable storage unit, rather, we use available IMS mechanisms to piggyback key information required to replaye the failed procedures. [62], [63] consolidate the LTE processing for fast execution of LTE procedures. The main focus of these work is to delegate the processing of critical procedures to a different network function instance. [62], [63] are mainly concerned regarding the performance and low latency of LTE procedures rather than providing highly reliable LTE packet core. [64] puts forward the need of reliable LTE design in NFV. But it does not provide any design solution to achieve it. [65] shows high availability in cellular networks is hard to achieve and provides recommendations to improve it. In our work, we take a focused approach to

address the reliability of IMS, which is a middleware to LTE for providing voice service and other multimedia services.

Fault tolerance has also been discussed in other NFV applications. [66] and [67] propose logging NF states during normal operations and reconstructing them after a failure. Their approaches cannot address real-time and transitory NF sessions recovery. [68], [69] and [70] discuss fault tolerance in non-IMS (SIP based) voice over IP applications. All these works are mainly concerned about the reliability of voice service, and provide suggestion to improve SIP protocol. In contrast, this paper provides the reliability of IMS and achieves the reliability of all multimedia applications (the voice is one of the multimedia applications). [71] discusses general load balancing strategies in vIMS and does not discuss vIMS working during faults.

In short, contrary to all above mentioned efforts, this paper exploits IMS domain specific knowledge to achieve reliability. It makes the critical actions execution reliable at the cost of non-critical actions failure.

IX. Conclusion

In this paper, we make the first effort in providing vIMS fault-tolerance in both control-plane and data-plane. We propose a design that can meet session-level resilience during faults. Our design brings innovation by partitioning IMS operations into critical and provisional action groups. The critical actions are recovered from failure in real time by letting provisional actions to fail.

Future Work: In this effort, we gain deep system insights and identify other research issues – to be discussed in future work. Few of them include vIMS performance when network services are scaled up and down, partitioning vIMS into control and data plane functionalities for mobile edge compute application, and maintaining vIMS security assumptions.

Acknowledgment

The authors would like to thank H.-Y. Tseng and C. Li who have produced the experimental results.

References

[1] *ETSI GS NFV 001: Network Function Virtualization (NFV) Use Cases*, 3GPP, ETSI, Sophia Antipolis, France, 2013.

[2] *Overture and Brocade and Intel and Spirent and Integra*, NFV Perform. Benchmarking VCPE, Intel, Spirent Integra, Santa Clara, CA, USA, 2015.

[3] *Virtual Solutions for Your NFV Environment*, F Netw., Seattle, WA, USA, 2018.

[4] *NFV: Beyond Virtualization*, F Netw., Seattle, WA, USA, 2014.

[5] *High Availability in OpenStack*. Accessed: May 1, 2019. [Online]. Available: http://docs.openstack.org/ha-guide/

[6] *OpenNebula:Host and VM High Availability*. Accessed: May 3, 2019. [Online]. Available: https://docs.opennebula.org/5.2/advanced_components/ha/frontend_ha_setup.html#overview

[7] *OpenIMS*. Accessed: Apr. 3, 2019. [Online]. Available: http://www.openimscore.org/

[8] *OpenStack*. Accessed: Jan. 20, 2019. [Online]. Available: https://www.openstack.org/software/

[9] (Nov. 2014). *IMS Release 10 Tutorial*. [Online]. Available: http://disi.unitn.it/locigno/didattica/AdNet/10-11/IMS_Tutorial_Scalisi.pdf

[10] *Radio Resource Control (RRC)*, document TS36.331, 3GPP, 2014.

[11] *Ericsson Blade System (EBS) for IMS*. Accessed: Feb. 13, 2019. [Online]. Available: http://archive.ericsson.net/service/internet/picov/get?DocNo=266/03819-FAP130506&Lang=AE&HighestFree=Y

[12] *Alcatel-Lucent End-to-End IMS Solution*. Accessed: Feb. 13, 2019. [Online]. Available: https://www.alcatel-lucent.com/solutions/communications-collaboration

[13] *High Reliability Using ATCA Platform for IMS*. Accessed: Mar. 8, 2019. [Online]. Available: http://www1.huawei.com/en/products/core-network/singlecore/ims-core/index.htm

[14] A. Avizienis, "The N-version approach to fault-tolerant software," *IEEE Trans. Softw. Eng.*, vols. SE–11, no. 12, pp. 1491–1501, Dec. 1985.

[15] *Fusion Programming*. Accessed: Apr. 11, 2020. [Online]. Available: www.huawei.com/ilink/en/download/HW_327323

[16] *IMS Restoration Procedures*, document TS23.380, 3GPP, 2014.

[17] *SK Telecom's vIMS Wins IMS Industry Awards 2016*. Accessed: Apr. 21, 2020. [Online]. Available: https://www.netmanias.com/en/post/korea_ict_news/10002/sk-telecom/sk-telecom-s-vims-wins-ims-industry-awards-2016.

[18] *Telefonica Deploys ZTE's vIMS Tech*. Accessed: Dec. 23, 2018. [Online]. Available: http://www.lightreading.com/nfv/vnfs-(virtual-network-functions)/telefonica–deploys-ztes-vims-tech/d/d-id/729264.

[19] *The Sprint NFV Journey: Accelerating Mobile Network Innovation With NFV OpenStack Cloud*. Accessed: Feb. 13, 2019. [Online]. Available:http://newsroom.sprint.com/the-sprint-nfv-journey.htm

[20] *Spark New Zealand Signs Deal with Ericsson to Deploy Virtualized IMS*. Accessed: Jan. 19, 2019. [Online]. Available: https://www.thefastmode.com/technology-solutions/10016-spark-new-zealand-signs-deal-with-ericsson-to-deploy-virtualized-ims

[21] *Telstra Advances Cloud Strategy Virtualizing Network Functions and Media Workloads*. Accessed: Jan. 21, 2019. [Online]. Available: https://www.ericsson.com/en/news/2017/2/telstra-advances-cloud-strategy-virtualizing-network-functions-and-media-workloads

[22] *Telecom Argentina: Transforming Into 2020 With cloud*. Accessed: May 3, 2019.[Online]. Available: http://www.huawei.com/en/publications/communicate/82/transforming-into-2020-with-cloud

[23] *GS-NFV-SWA-001: NFV Virtual Network Functions Architecture*, ETSI, Sophia Antipolis, France, 2015.

[24] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latré, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 98–105, Jan. 2016.

[25] J. Mudigonda and *et al.*, "NetLord: A scalable multi-tenant network architecture for virtualized datacenters," *ACM SIGCOMM Comput. Rev.*, vol. 41, no. 4, pp. 62–73, 2011.

[26] *Hypervisor Support in OpenStack*. Accessed: Feb. 11, 2019. [Online]. Available: https://wiki.openstack.org/wiki/HypervisorSupportMatrix

[27] *OpenStack—A Modular Collection of Cloud Services*. Accessed: Mar. 11, 2019. [Online]. Available: https://netapp.github.io/openstack-deploy-ops-guide/mitaka/content/section_modular-collection.html

[28] *Beyond Virtual Machines: Overview of Bare Metal Provisioning*. Accessed: Mar. 13, 2019. [Online]. Available: https://www.mirantis.com/blog/bare-metal-provisioning-with-openstack-cloud/

[29] R. Stewart, "Reliable massively parallel symbolic computing: Fault tolerance for a distributed Haskell," Ph.D. dissertation, School Math. Comput. Sci., Heriot Watt Univ., London, U.K., 2013.

[30] *Verizon Worried About SDN, NFV Impacts*. Accessed: Dec. 23, 2018. [Online]. Available: http://www.lightreading.com/nfv/nfv-strategies/verizon-worried-about-sdn-nfv-impacts/d/d-id/709392.

[31] *Vodafone Calls for End to Five Nines*. Accessed: Mar. 3, 2019. [Online]. Available: http://www.lightreading.com/carrier-sdn/nfv-(network-functions-virtualization)/vodafone-calls-for-end-to-five-nines/d/d-id/719137.

[32] C. Guidi, I. Lanese, F. Montesi, and G. Zavattaro, "Dynamic error handling in service oriented applications," *Fundamenta Informaticae*, vol. 95, no. 1, p. 73, 2009.

[33] *Data Breach Hits Roughly 15M T-Mobile Customers, Applicants*. Accessed: Apr. 13, 2019. [Online]. Available: http://www.cnet.com/news/data-breach-snags-data-from-15m-t-mobile-customers/

[34] P. J. Denning, "Fault tolerant operating systems," *ACM Comput. Surv.*, vol. 8, no, 4, pp. 359–389, 1976.

[35] *IP Multimedia Subsystem (IMS);Stage 2*, document TS23.228, 3GPP, 2012.

[36] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*. London, U.K.: Pearson, 2005.

[37] T. Anderson, P. A. Lee, and S. K. Shrivastava, "A model of recoverability in multilevel systems," *IEEE Trans. Softw. Eng.*, vol. SE-4, no. 6, pp. 486–494, Nov. 1977.
[38] *SIP 182 Queued Message*. Accessed: Apr. 13, 2019. [Online]. Available: http://www.dialogic.com/webhelp/CSP1010/8.4.1_IPN3/sip_software_chap_-_sip_182_queued_message.htm
[39] *Support of SIP P-Headers for 3GPP*. Accessed: Mar. 11, 2019. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/voice_ip_comm/pgw/9/feature/module/9-8_1_/pheaders.html/
[40] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, *SIP: Session Initiation Protocol*, document RFC2543, 1999.
[41] *Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3*, document TS24.301, 3GPP, Jun. 2013.
[42] *Access Security for IP-Based Services*, doucment TS33.203, 3GPP, Sep. 2014.
[43] P. Gill and *et al.*, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. ACM SIGCOMM*, 2011, pp. 350–361.
[44] B. Zhu and *et al.*, "Avoiding the disk bottleneck in the data domain deduplication file system," in *Proc. USENIX Conf. File Storage Technol.*, 2008, pp. 1–4.
[45] *Optimizing Large Data Handling in SAP ASE for Peformance*, S. T. Report, 2012.
[46] *Policy and Charging Control Architecture*, document TS 23.203, 3GPP, 2013.
[47] *Speech and multimedia Transmission Quality (STQ); IMS/PES/VoLTE exchange performance requirements*, document TS101.563, 3GPP, 2014.
[48] *VMware Virtualization for Desktops & Servers*. Accessed: Mar. 11, 2019. [Online]. Available: http://www.vmware.com/
[49] W. River, "Virtualization: Ensuring carrier grade availability," SAP, Weinheim, Germany, Tech. Rep. SP03, 2014.
[50] *IMS Network Testing: IMS Configurations and Benchmarks*, document TS186.008–2, 3GPP, 2013.
[51] *Default openstack timers*. Accessed: Apr. 11, 2020. [Online]. Available: http://docs.openstack.org/kilo/config-reference/content/cinder-conf-changes-kilo.html
[52] *SIP Router Project*. Accessed: Dec. 21, 2018. [Online]. Available: http://sip-router.org/
[53] *AT Commands List*. Accessed: Dec. 23, 2018. [Online]. Available: http://www.lte.com.tr/uploads/pdfe/1.pdf
[54] M. T. Raza and S. Lu, "Reducing latencies and improving fault tolerance in NFV of 3GPP standardized IMS," in *Proc. ACM/IEEE CNSM*, 2017, pp. 1–9.
[55] M. T. Raza and *et al.*, "Modular redundancy for cloud based IMS robustness," in *ACM MobiWac*, 2017, pp. 75–82.
[56] M. Abu-Lebdeh, J. Sahoo, R. Glitho, and C. W. Tchouati, "Cloudifying the 3GPP IP multimedia subsystem for 4G and beyond: A survey," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 91–97, Jan. 2016.
[57] R. Glitho, "Cloudifying the 3GPP IP multimedia subsystem: Why and how?" in *Proc. NTMS*, 2014, pp. 1–5.
[58] A. Sheoran and *et al.*, "Contain-ed: An NFV micro-service system for containing E2E latency," in *Proc. ACM HotConNET Workshop*, 2017, pp. 54–60.
[59] A. Sheoran and *et al.*, "An empirical case for container-driven fine-grained VNF resource flexing," in *Proc. IEEE NFV-SDN Conf.*, Nov. 2016, pp. 121–127.
[60] G. Carella and *et al.*, "Cloudified IP multimedia subsystem (IMS) for network function virtualization (NFV)-based architectures," in *IEEE ISCC*, Jun. 2014, pp. 1–6.
[61] B. Nguyen *et al. Technical Report*. Albuquerque, NM, USA: Microsoft Research, 2018.
[62] M. T. Raza, D. Kim, K.-H. Kim, S. Lu, and M. Gerla, "Rethinking LTE network functions virtualization," in *Proc. IEEE 25th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2017, pp. 1–10.
[63] Z. A. Qazi, "A high performance packet core for next generation cellular networks," in *Proc. ACM SIGCOMM*, 2017, pp. 348–361.
[64] A. Gonzalez, "Service availability in the NFV virtualized evolved packet core," in *Proc. IEEE Globecomm*, Dec. 2015, pp. 1–6.
[65] A. Elmokashfi, "Adding the next nine: An investigation of mobile broadband networks availability," in *Proc. ACM Mobicom*, 2017, pp. 88–100.
[66] J. Sherry and *et al.*, "Rollback-recovery for middleboxes," in *Proc. ACM SIGCOMM*, 2015, pp. 227–240.
[67] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico replication: A high availability framework for middleboxes," in *Proc. 4th Annu. Symp. Cloud Comput. (SOCC)*, 2013, pp. 1–15.
[68] M. Bozinovski, L. Gavrilovska, and R. Prasad, "Fault-tolerant SIP-based call control system," *Electron. Lett.*, vol. 39, no. 2, pp. 254–256, Jan. 2003.
[69] H. Pant, A. R. McGee, U. Chandrashekhar, and S. H. Richman, "Optimal availability and security for IMS-based VoIP networks," *Bell Labs Tech. J.*, vol. 11, no. 3, pp. 211–223, Nov. 2006.
[70] S. Palkar *et al.*, "E2: A framework for NFV applications," in *Proc. 25th Symp. Operating Syst. Princ. (SOSP)*, 2015, pp. 121–136.
[71] F. Lu, H. Pan, X. Lei, X. Liao, and H. Jin, "A virtualization-based cloud infrastructure for IMS core network," in *Proc. IEEE 5th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2013, pp. 25–32.

Muhammad Taqi Raza (Member, IEEE) is currently an Assistant Professor with the Information Systems Department, The University of Arizona, Tucson, AZ, USA. His research interests include network security, network function virtualization, and networked systems and their applications with an emphasis on cellular security.

Songwu Lu (Fellow, IEEE) is currently a Professor with Computer Science Department, University of California–Los Angeles, Los Angeles, CA, USA. His research interests include wireless networking, mobile systems, sensor networks, and data center networking.

Prof. Lu was on the editorial board of the IEEE/ACM TRANSACTIONS ON NETWORKING and the boards of the IEEE TRANSACTIONS ON MOBILE COMPUTING, *ACM Wireless Networks*, and IEEE WIRELESS COMMUNICATIONS MAGAZINE.