LTE NFV Rollback Recovery

Muhammad Taqi Raza, *Member, IEEE*, Zhowei Tan, Ali Tufail, and Fatima Muhammad Anwar

***Abstract*—Network Function Virtualization (NFV) migrates the carrier-grade LTE Evolved Packet Core (EPC) that runs on commodity boxes to the public cloud. In the new virtualized environment, LTE EPC must offer high availability to its mobile users upon failures. Achieving high service availability is challenging because failover procedure must keep the latency-sensitive control-plane procedures intact during failures. Through our empirical study, we show that existing recovery mechanisms on the cloud and standardized LTE solutions are coarse-grained, thus unable to quickly recover from failures. They incur LTE service outage, lost network connectivity, and slow recovery. To address these issues, we describe a new design for fault-tolerant LTE EPC. It provides quick failure detection and timely recovery from failed operations. To reduce failure detection time, it leverages frequent retransmission of LTE control-plane signaling within EPC as an indication of failure. To recover from failure, it adopts a checkpointing based rollback recovery approach in the LTE context and addresses the shortcomings known in the classic checkpointing approach. Our design is LTE standard-compliant and works as a plug-and-play without modifying existing LTE implementations. Our results show that this approach can recover from the failure in 2.6 seconds and only incurs tens of milliseconds of overhead.**

***Index Terms*—NFV, availability, fault tolerance, telecommunication network reliability.**

I. Introduction

LTE NETWORK must be highly available to provide its mobile subscribers with services (voice, data, AR/VR, and others). Telecom vendors have since promised high availability of five-nines by using their vender-specific solutions over carrier-grade boxes. Each LTE “network function” is developed by a single telecom vendor addressing specific requirements of that network function. These vendors employ: (a) hardware level mechanisms to tolerate occasional internal components and modules failures; (b) software level techniques to ensure redundancy, both for error detection and error recovery; and (c) overly manager that ensures strong coupling between underlying hardware and the software [1], [2].

The Network Function Virtualization (NFV), however, moves away from propriety solutions and brings LTE implementation over public clouds. Providing high LTE service availability over general-purpose hardware and software platforms is challenging [3], [4].

In this paper, we study the fault tolerance for the highly-available LTE NFV. We first perform an empirical assessment and reveal that current LTE and cloud-based mechanisms perform poorly during failures. Existing approaches cause LTE *service disruption*, *loss of network connectivity*, and take *tens of seconds to recover* LTE network service. These issues emerge because existing NFV failure recovery solutions are too *coarse grained* for LTE. These solutions periodically take snapshots of the virtual machine and do not log LTE control-plane signaling messages; hence fail to recover LTE service during failover. Further, contemporary approaches rely on heartbeat messages – implemented over cloud platform (such as ZooKeeper [5]) and LTE protocols (such as SCTP [6]) – to detect the failure. This is too slow for LTE control-plane procedures, which timeout in a few seconds.

We next put forward our design to address slow failure detection and inefficient failure recovery mechanisms in existing LTE-NFV. Our design exploits LTE specific information to achieve high availability. To perform timely recovery, our design proposes checkpoint based rollback recovery with *receive determinism* algorithm. Because LTE control-plane procedures have causality relationship (in which one message triggers the execution of the other message and so on), blocking mode execution (a device can initiate only one procedure at a time), and receive determinism (the same message is replayed during rollback that was saved in checkpoint), our design can mitigate the issues found in classical checkpointing based rollback recovery approaches [7].

Finally, our design makes uses of available LTE path management features in its advantage to provide a quick failure detection. It uses retransmission of signaling messages within EPC as a prediction of failure and starts frequent probing towards non-responding EPC NF to confirm the failure.

We implement our design over OpenAirInterface [8], an open-source LTE implementation. Our evaluation shows that our approach: (1) detects the failure in approximately 2.5 seconds with no false positives, which is 4× improvement over existing cloud mechanism, (2) reduces recovery time by up to 6.5× compared to state-of-the-arts implementation, and (3) incurs the negligible checkpointing overhead of tens of milliseconds.

The scope of the paper is limited to NFV of LTE core network (Evolved Packet Core). The network operators can use

Manuscript received 27 September 2021; revised 29 January 2022 and 28 May 2022; accepted 1 June 2022. Date of publication 10 June 2022; date of current version 12 October 2022. This work was supported in part by NSF under Grant CNS-205162, and in part by the Arizona TRIF Grant. The associate editor coordinating the review of this article and approving it for publication was M. Tornatore. *(Corresponding author: Muhammad Taqi Raza.)*

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

Zhowei Tan is with the Computer Science Department, University of California at Los Angeles, Los Angeles, CA 90095 USA.

Ali Tufail is with the School of Digital Science, University of Brunei Darussalam, Gadong 1410, Brunei.

Fatima Muhammad Anwar is with the Department of Electrical and Computer Engineering, University of Massachusetts at Amherst, Amherst, MA 01003 USA.

Digital Object Identifier 10.1109/TNSM.2022.3182008

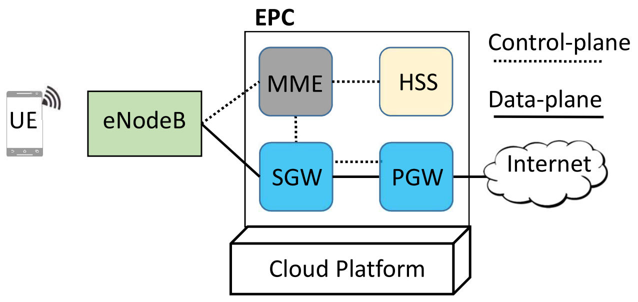


Fig. 1. LTE architecture over NFV: an overview.

existing radio infrastructure while moving their core network functionalities over the cloud.

II. Related Work

The most recent works on telecom–NFV related reliability are [3], [9], [10], [11], [12], [13], [14]. Reference [9] re-designs EPC architecture for public cloud deployment and guarantees reliable LTE operations. The main idea of [9] is to break EPC functionality into stateless and stateful components. The stateful information is stored into highly reliable storage that achieves the high availability. Contrary to [9], our solution does not re-design LTE EPC architecture and acts like plug-and-play with existing EPC architecture. It uses rollback recovery in LTE context and recovers from failure. References [10], [11] consolidate the LTE processing for fast execution of LTE procedures. References [3], [12] provide reliability to IMS core by exploiting IMS domain knowledge and refractoring IMS software modules to achieve fault tolerance. References [3], [12] cannot recover LTE failures. References [13], [14] puts forward the need of reliable LTE design in NFV. Our paper addresses these concerns by providing fault tolerance to LTE procedures.

III. Background

LTE network consists of three main components: User Equipment (UE), evolved Node Base-station (eNodeB), and Evolved Packet Core (EPC), as shown in Figure 1. The eNodeB anchors as a radio interface between UE and EPC. EPC communicates with packet data networks in the outside world such as the Internet. The EPC is composed of a number of Network Functions (NFs): the Serving Gateway (SGW), the PDN Gateway (Packet Data Network Gateway or PGW), the Mobility Management Entity (MME), the Home Subscriber Server (HSS), and a few others. These LTE EPC NFs handle control-plane and data-plane traffic through separate network interfaces and protocols. As shown in Figure 1, the control-plane traffic from radio network is sent to MME, whereas data-plane traffic is forwarded to SGW. MME acts as a central management entity that authenticates and authorizes UE, handles device procedures (such as device registration, handover, location update, and service provisioning), and maintains SGW and PGW connections for data-plane traffic.

Network Function Virtualization (NFV) for 4G LTE: In LTE–NFV, LTE software implementation is deployed over cloud platforms. There exist a number of LTE software implementations that include OpenEPC [15], OpenAirInterface [8], OpenLTE [16] and others that can be deployed over cloud platforms, such as OpenStack [17], OPNFV [18], vSphere [19], etc. The cloud platform acts as a manager and provides virtualization of hardware resources to LTE NFs. Softwarization of LTE NFs accelerates the innovation by lowering operational and capital expenditures [20], [21]. The network operators, such as Verizon [22], Vodafone [23], SK Telecom [24], and others, are currently considering to deploy LTE–NFV for existing and future 5G applications.

TABLE I
Downtime Allowed for High Availability

Availability %	Downtime/year	Downtime/week	Downtime/day
99.95% ("three & half nines")	4.38 hours	5.04 minutes	43.2 seconds
99.99% ("four nines")	52.56 minutes	1.01 minutes	8.64 seconds
99.999% ("five nines")	5.26 minutes	6.05 seconds	864.3 milliseconds

IV. Availability in Virtualized LTE

The availability of LTE network is usually expressed as a percentage of uptime in a given day, week or a year. Table I shows the downtime that will be allowed for a particular percentage of availability, presuming that the system is required to operate continuously. LTE networks are expected to achieve five-nine (99.999%) high availability [25], [26]. This availability is defined under the assumption of highly available channel (i.e., the unavailability due to bad signals or no coverage is not considered) [26], [27].

Failure Recovery Mechanisms Today: LTE network operators aim to provide all-time service access to their subscribers. In order to provide end-to-end LTE services, LTE NFs must be operative even during faults. Conventionally, carrier grade LTE NFs employ sufficient mechanisms at their hardware and software to tolerate faults and provide high availability of network resources. These include Cisco ASR 5500 platform [1], Ericsson Blade System MkX platform [2], Alcatel-Lucent 5620 SAM platform [28], Huawei's ATCA platform [29], and a few others. The NFV that moves away from propriety solutions relies on 3GPP standard and cloud platforms to meet high availability requirements.

Failure Model: Similar to these efforts [30], [31] we consider the *fail-stop failure*, where under failure "the component changes to a state that permits other components to detect that a failure has occurred and then stops". We focus on the network function (NF) failures, in which the process of the LTE functions (e.g., MME, S-GW/P-GW) crashes.

Scope: LTE–NFV efforts span on virtualizing LTE EPC [10] and LTE radio access network [32]. In this paper, we limit our scope on NFV of LTE EPC.

V. Empirical Assessment of the State-of-the-Arts

In this section, we unveil the deficiencies of current cloud-based (Section V-A), and LTE's built-in failover mechanisms (Section V-B). We summarize these deficiencies in Table II. We discover that both mechanisms are insufficient for five-nine

TABLE II
SUMMARY OF LIMITATIONS OF EXISTING APPROACHES

Recovery at	**Recovery mechanisms**	**Failure resilience**	**Resource efficient**	**Key limitations**
VMware	Yes	No	No	Snapshots are expensive, VM stucks during snapshot, not scalable.
OpenStack	Yes	No	No	Work-in progress, procedures abort on failure, re-attach req. after recovery.
ETSI 3GPP standards	Yes	No	No	Slow, procedures abort on failure, devices re-attaches after recovery.
ETSI NFV documents	Yes	No	No	procedures abort, recovery happens after several timeouts.

LTE high availability in NFV. The causes are diverse, including the mismatch between cloud and NF-level failure recovery, coarse-grained recovery, long recovery time, and slow failure detections. They motivate us to devise new fault-tolerance mechanisms.

Methodology: We run experiments on open source LTE platform, OpenAirInterface [8]. We consider the shortest timer values and best possible solutions available among (1) 3GPP standard, (2) 3GPP-NFV documents, and (3) Cisco vendor platform[1] [6]. We do not assume hot-standby configuration, a naïve solution used by the telecom industry to address fault tolerance [33], [34], [35] in NFV, which is resource inefficient and doubles the cost of the network [36], [37].

A. Deficiencies in Cloud-Level Recovery

We discover that existing cloud-based failure recovery mechanisms are neither sufficient nor efficient for LTE control-plane functions in NFV.

1) Coarse-Grained Failure Recovery: We find that the existing cloud-based failure recoveries are *coarse-grained* and thus potentially waste storage resource. We analyze both the open-source platform (OpenStack), and the commercial platform (VMware vSphere).

- *OpenStack [38]:* We find that OpenStack solutions for compute-node's high availability are immature and are still in-progress [38]. Their solutions cannot recover the subscribers session information during failure and require all UEs to re-attach with the network after failure recovery.
- *VMware vSphere [39]:* It provides failure recovery mechanism through periodic snapshot. We find that vSphere's snapshot approach has several limitations and cannot be used for delay sensitive LTE NFV failure recovery. First, when it takes the snapshot, the entire state of the VM is freezed; VMware calls this as stunned operation, and VM becomes non-responsive [39]. Second, the snapshot requires complete VM memory backup, even if there are few changes in the memory since last snapshot. We show this in Figure 2 that VMware snapshot approach takes at least 43,100 MBs for backup purpose compared to the required (subscribers desired states and variables for successful rollback recovery procedure) 110 MB – which is at least 40× more expensive in terms of storage space.

Impact The OpenStack detects the failure through heartbeat messages (which are in the order of a couple of seconds [25]) and failure is recovered by assigning an alternative VM instance. As a result, there is a long service disruption and loss of network connectivity. Similar to OpenStack, the VMware solution has two major impacts. First, VMware's stunned operation [39] results into temporary LTE service disruption which can take up to 60 seconds [40]. Second, there is a tradeoff between frequency of snapshots and fine-grained recovery of LTE failed procedures. LTE procedures require snapshots to be taken at the granularity of milliseconds (as it takes few seconds for one LTE procedure to conclude) which is not possible with VMware approach.

[1]The reason we chose Cisco as our vendor choice for comparison is that its configuration and implementation specifications are available on the Internet.

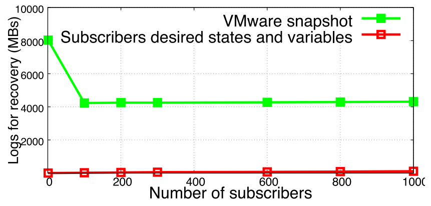


Fig. 2. For failure recovery: VMware approach takes several thousands MBs of snapshot, even though it is sufficient to backup only subscriber related messages, states and variables of few MBs.

TABLE III
DOWNTIME PER DAY FOR LTE CONTROL-PLANE PROCEDURES. ONE FAILURE PER DAY FOR EPC NF IS CONSIDERED

Proc	MME failure			SGW/PGW failure			HSS failure		
	Cloud 5 nines	MME	UE	Cloud 5 nines	S/P GW	UE	Cloud 5 nines	HSS	UE
Attach	864.3ms	2s	10s	864.3ms	1s	10s	864.3ms	2.5s	10s
TAU	864.3ms	2s	10s	864.3ms	1s	10s	864.3ms	2.5s	10s
SR	864.3ms	2s	5s	864.3ms	1s	5s	864.3ms	2.5s	5s

Root Cause Current cloud systems do not provide fault tolerance for *session-level resilience*. Even though, VMware snapshot approach is available for cloud based service robustness, it is designed for VM backup purpose rather than VM applications' high service availability.

Issue 1: State-of-the-art cloud based approaches are coarse-grained and thus consume more storage resources.

2) Cloud Availability ≠ LTE NF Availability: We find that the five-nine clouds today [25] can only ensure *three and half nine* availability for LTE control-plane in NFV. Table III shows the downtime of the cloud nodes and corresponding LTE network functions (using OpenAirInterface [8]). Table III shows the downtime per day for MME, SGW/PGW and HSS under Attach Request, Tracking Area Update (TAU), and Service Request (SR) LTE procedures. LTE failure recovery time includes the time taken by cloud platform, LTE NF bootstrap and connection setup time, and the time UE waits for timeout to re-initiate the failed procedure. We can see from Table III that the total *user-perceived*[2] downtime for Attach/TAU and SR is 10 seconds and 5 seconds, respectively. Although the cloud's downtime meets five-nine criteria, the LTE control functions do not (as like the case of other applications [41]): The bootstrap and connection setup time for MME, SGW/PGW and HSS are 2 seconds, 1 second, and 2.5 seconds, respectively. Further, these NFs need to wait either 5 seconds or 10 seconds for SR or TAU/Attach request procedure to be re-initiated from the device.

[2]The time after which the user device will re-initiate the failed procedure, which is typically equivalent to LTE procedure timeout value.

As a result, the high availability for LTE procedures are *'three and a half nines'*,[3] even if the cloud platforms could achieve five nines of infrastructure availability.

Root Cause: The cloud's failure recovery only ensures high availability at *VM level, not NF level*.

Issue 2: Cloud-level failure recovery is insufficient for five-nine LTE control-plane, because of the network-side NF interplays and device-side slow failure detections.

B. Deficiencies Rooted in LTE Design

We discover that LTE standardized mechanisms also bring LTE service disruption during failure and loss of network connectivity after failure recovery.

1) Loss of Network Connectivity After Failure Recovery: LTE-NFV solutions from 3GPP (refer to 3GPP specification reports that discuss scalable LTE-NFV architecture for reliability [42], and end-to-end LTE-NFV reliability models [43]) are coarse grained. When the failure occurs, VNF failure recovery procedure selects an alternative VNF (that VNF does not have UE contexts) [42], [43].

To quantify this, we consider SR procedure and bring the fail-stop failure. Once the LTE service is back in about 32 seconds, an alternative MME takes charge of the failed MME. The UE sends SR message, which is returned with an error cause *#9 (UE identity cannot be derived by the network)* (refer to 3GPP NAS specification section 5.6.1.5: Service request procedure not accepted by the network [44]). This has been shown in device side logs snapshot in Figure 4.

On receiving the service reject message, the UE enters into the state EMM-DEREGISTERED and automatically initiates the registration procedure. Once the device is registered with LTE network, it can get the LTE service.

Impact There are several issues in LTE NFs restoration procedure after failure. First, available LTE recovery mechanisms result into loss of network connectivity. Second, EPC NF failure is propagated to device which is against the philosophy of fault tolerance [45]. Third, EPC relies on device to recover from failure by performing re-registration procedure with LTE network. Fourth, LTE control-plane failure not only terminates any control-plane session, but data-plane traffic is also aborted, where device initiates re-registration procedure with LTE network.

Root Cause The standardized mechanisms are designed by considering highly reliable EPC NFs (five-nine availability) where failure is only anticipated for vendor-specific platforms that have built-in redundancy for fault tolerance [1], [2], [28], [29]. The 3GPP standards provide only load balancing, load re-balancing [46] and service restoration [47] mechanisms to distribute the workloads. It does not standardize the LTE control-plane procedure level recovery mechanisms.

Issue 3: Existing failure recovery brings loss of network connectivity and results in termination of on-going data-plane traffic.

2) Slow Failure Detection: We find that LTE recovery mechanisms are also slow to detect the failure. As shown in

[3]Assuming good radio conditions with no radio failures.

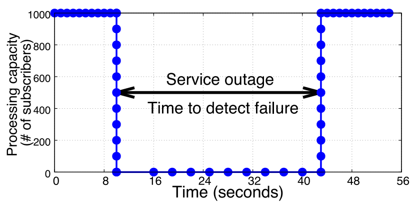
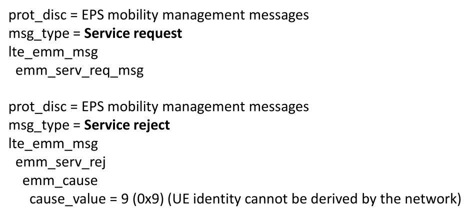


Fig. 3. Slow failure detection results into LTE service disruption.



```
prot_disc = EPS mobility management messages
msg_type = Service request
lte_emm_msg
  emm_serv_req_msg

prot_disc = EPS mobility management messages
msg_type = Service reject
lte_emm_msg
  emm_serv_rej
    emm_cause
      cause_value = 9 (0x9) (UE identity cannot be derived by the network)
```

Fig. 4. Device side logs show that the service request was rejected on failure recovery.

Figure 3, before any failure our MME has processing capacity of handling 1,000 UEs (considering both their control and data plane traffic). In our experiment, we launch LTE control-plane procedures with LTE and stopped MME instance (imitating the fail-stop failure of MME). As soon as the failure occurs (at 10th second), the MME stops responding LTE control-plane signalling messages. The subscriber devices timeout and retry their LTE procedures. They keep retrying until a response is received to their requests. It takes them 32 seconds to receive any response to their requests (when the VM has recovered), after which the alternative MME serves the subscriber requests.

Impact Slow failure detection has two major impacts. First, the subscriber devices will go through *service outage during failure*, which takes around 32 seconds. This value is not acceptable for delay-sensitive LTE control-plane procedures. The retry interval for Attach/TAU and SR procedures are 10 seconds (timer T3411 [44]) and 5 seconds (timer T3417 [44]), respectively.

Another major issue is that *eNodeB receives higher signaling load*. All the devices retry for failed service (e.g., SR is retried every 5 seconds) and bring signaling spikes at eNodeB.

Root Cause We find that 3GPP standard does not provide explicit failure detection mechanism. Rather, it relies on communication interfaces' heartbeat to record the failure. When MME goes down, the failure is detected by both eNodeB and SGW. The eNodeB detects the MME failure through heartbeat mechanism provided by SCTP protocol interface whose value is 30 seconds (according to Cisco implementation [6]).

The SGW also detects the MME failure through *path management procedure* [10], [48]. The SGW takes 25 seconds (timer T3-Response value choice by Cisco [49]) to detect the MME failure and mark it unreachable. However, the SGW cannot select an alternative MME and it relies on eNodeB to assign a different MME from its MME pool [50]. This new MME contacts SGW and starts serving the subscribers. We should add that, although the cloud platforms provide mechanisms (like ZooKeeper [5] or Pacemaker [51]) faster than eNodeB failure detection through SCTP interface, these solutions do not work in LTE, because no mechanism exists in which an alternative MME can ask eNodeB to associate with it.

Issue 4: Fail-stop failure detection is slow that causes service disruption during failure.

VI. Design Rationale and Overview

A. Design Insights

We leverage the way LTE procedures get executed in designing our solution. Before we describe our solution, we want to highlight the key LTE specific design insights.

Messages Causality (I_1): To execute one LTE procedure, EPC NFs exchange messages with each other. All these messages have *causality relationship* or *happen-before relationship*. That is, a message m_1 may causally affect another message m_2, if and only, if message m_2 can only execute once message m_1 has been executed. This means, m_1 happens before m_2, and forms a causality relationship as $\mathrm{m}_1 \rightarrow \mathrm{m}_2$. For example, in device attach procedure, the device is first authenticated (authentication request, followed by authentication response messages) at MME; thereafter, SGW creates device sessions, when it receives create session request message from MME, processes it and sends the create session request message to PGW; the PGW processes the request (creates PDP context, assigns bearers and QoS, and enables charging operation) and replies create session response message to SGW; the SGW replies create session response message to MME. All these messages have happen-before relationship.

Device Procedures Execute in Blocking Mode (I_2): We refer to LTE NAS standard and find that LTE NAS procedures from a device execute in blocking mode. That is, when one procedure is initiated by a device, it cannot initiate any other procedure until the previous one results into success/failure. Refer to Figure 5.1.3.2.2.7.1: EMM main states in the UE in the NAS spec [44] that explains our insight through transitions between procedures' *init* states and device's *registered* state.

Replay Messages are Receive-Deterministic (I_3): LTE states are non-deterministic where the non-determinism comes from the decision logic and actions due to timers. However, it is not the case for LTE signalling messages that need to be replayed (when previously executed control-plane messages are resent for failure recovery). The receiver will get the same message that it has executed before, and it produces the same result sent earlier. Therefore, we say that the replayed messages are deterministic at receiver (that generates the same response message which was sent earlier).

Split and Join Relationship of Concurrent Execution of Messages (I_4): Certain messages are allowed to execute concurrently by LTE standard (they are not causally related according to LTE standard), such as Authentication Request and Security Mode command messages, TAU related and Identity Request messages, and a few others (refer to message collision at [44]). When the concurrency happens, the main procedure execution splits into two parts; both parts execute concurrently. Later, when both parts finish their tasks, they join together and create one execution path again; afterwards the rest of the messages are executed following happen-before relationship. Because we already know from the standard which messages can be executed concurrently, we can employ specific mechanism for those few messages.

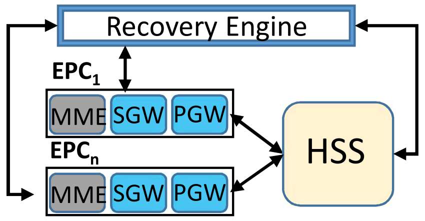


Fig. 5. Our design architecture.

LTE Path-Management Mechanism Retransmits Messages on Timeout (I_5): LTE EPC leverages its path-management mechanism to retransmits GTP-C signalling messages in case the receiver EPC NF does not respond in time. This retry interval is calculated based on *echo-request* and *echo-response* values between two EPC NFs [48]. We can leverage this insight to quickly detect the failure.

B. Solution Overview

Solution Components: Our solution has two main components: (a) retransmission-induced quick failure detection, and (b) receive-deterministic rollback recovery through checkpointing on failure.

Our design architecture is shown in Figure 5. During normal working, every NF independently takes periodic checkpoints. Every checkpoint saves subscribers session states, control-plane procedures' messages and their execution, and NF configurations, since preceding checkpoint into highly reliable HSS database [52], [53], [54], [55]. Our solution leverages LTE control-plane messages retransmission mechanism between two EPC NFs to predict the failure, in case one NF stops responding the control-plane signalling messages. The predicted failure is reported to recovery engine. The recovery engine confirms the failure by frequently probing the non-responding NF. If failure is confirmed, it replaces the failed NF with an alternative one. The alternative NF retrieves the last saved checkpoint and rollbacks to that checkpoint. Then it replays the control-plane messages to reach the state at which the failure has occurred. Once all the recorded messages are replayed, the halted control-plane procedures resume without breaking the connectivity. Our solution addresses the shortcomings of existing failure recovery mechanisms (as discussed in Section V). Our approach takes periodic checkpoints to provide failure resilience. These checkpoints only record LTE specific information instead of complete VM snapshot, addressing **issue 1**. Our solution makes use of LTE path management mechanism to quickly detect the failure and solves **issue 4**. After failure detection, the failed LTE control-plane procedures are quickly restored by rolling back to the last saved checkpoint and replaying the checkpoint messages, solving **issues 2 and 3**.

VII. Design

We put forward a design to achieve efficient LTE control-plane rollback recovery, and quick failure detection.

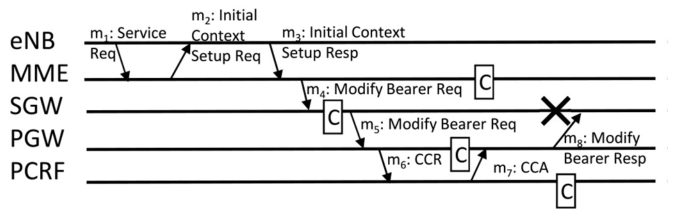


Fig. 6. Service request as an example. Every EPC NF will take its checkpoint independent from other NFs. Before/after sending/receiving messages every NF may have some internal execution of messages (not shown in the Figure). In this Figure, we only show the SR as message exchange.

Algorithm 1: Checkpointing for LTE Control-Plane Procedures

```
$UESessions$ = Session states of all devices;
$NFConfigs$ = NF$_j$ configurations;
$UEProcMsgs$ = Messages of all procedures of devices;
Let $CkPt$ = $UESessions$ + $NFConfigs$ + $UEProcMsgs$;
if the concurrent messages execution is starting for procedure$_k$ then
    Take checkpoint ($CkPt_i$) at NF$_j$;
    SEND ($i$, NF$_j$, $CkPt_i$) to HSS;
if the concurrent messages execution has finished for procedure$_k$ then
    Take checkpoint ($CkPt_i$) at NF$_j$;
    SEND ($i$, NF$_j$, $CkPt_i$) to HSS;
if certain time has passed at NF$_j$ then
    Take checkpoint ($CkPt_i$) at NF$_j$;
    SEND ($i$, NF$_j$, $CkPt_i$) to HSS;
end
```

A. Enabling LTE Control-Plane Rollback Recovery

We first brief the limitation of existing available rollback recovery mechanism in LTE context. Thereafter, we discuss our approaches of communication-induced checkpointing and receive-deterministic rollback recovery.

1) Limitation of Existing Rollback Recovery Mechanism: Conceptually, the simplest solution is that different LTE NFs take their checkpoints more or less independently. When a failure occurs, the LTE NF is rolled back to the latest checkpoint and resumes the LTE failed control-plane procedure. However, this simple approach can cause the *domino effect* during rollback recovery [56]. The domino effect appears when a subset of LTE NFs, which have to be resumed after a failure, rollback unboundedly while determining a set of mutually consistent checkpoints among all NFs. In the worst case, all the NFs have to roll all the way back to the beginning (the first most checkpoint when these NFs started) [56]. We find that in LTE context the *domino effect* can be even worse, where it can take the rollback recovery to the first most checkpoint, when the device was initially registered – days or even weeks before the failure.

Service Request Procedure as an Example: We understand the *domino effect* during rollback in LTE by using LTE SR control-plane procedure as an example. As shown in Figure 6, assume the failure at SGW occurs after sending message m_5 and just before receiving message m_8. The SGW rolls back to its last checkpoint (which is in between messages m_4 and m_5). From SGW point of view, the message m_5 was never sent (as the checkpoint was taken before message m_5 was sent); whereas from PGW point of view, m_5 was sent by SGW and it has progressed towards message m_8. These inconsistencies between SGW and PGW bring the phenomena of *orphan messages* – the messages which are not mutually claimed by any two NFs, and cannot establish send/receive relationship. Furthermore, SGW will not re-send the message m_5 because it requires message m_4 to be received before sending m_5. To deal with this problem, protocols based on checkpointing force the *orphan messages* NFs (i.e., PCRF and PGW) to rollback to the checkpoint preceding the *orphan messages* reception. This rollback may trigger more rollbacks at other NFs (such as MME) to find the consistent global state. In the worst case, all NFs end up rolled-back to their initial checkpoints. This initial state might be the start of LTE EPC NFs that means no control-plane procedure was *ever* executed.

Furthermore, this approach can lead to storage overhead in which all the checkpoints need to be saved because the recovery procedure could rollback to the first most checkpoint (i.e., due to *domino effect*). In this paper, we use LTE specific insights and design our checkpointing and rollback recovery algorithms that solve the *domino effect* without requiring different NFs to coordinate to do checkpointing (solutions that are considered to address *domino effect* problem [7], [56], [57]).

2) Our Checkpointing and Rollback Recovery Mechanisms: **Independent checkpointing** In our approach, every NF takes its own checkpoint whenever it wants. Our checkpointing approach does not put any restriction on NF regarding when it should take a checkpoint and how many checkpoints should be taken in a given period of time; except the case of *concurrent messages* execution. To make sure that after failure the concurrent messages execution are replayed exactly in the same order as they were executed before the failure, our approach ensures either all of the concurrent messages will be re-executed or none during rollback recovery. To achieve this, our approach puts a forced checkpoint as soon as the concurrent messages execution starts, and takes another forced checkpoint when the concurrent execution stops. It means we isolate the execution block of concurrent messages execution. Our approach can do so because of split and join phenomena to achieve concurrency (Insight: I_4). To take an example, when concurrency happens in LTE control-plane procedure, the main thread responsible of executing LTE procedure creates a new thread() (i.e., child thread) and delegates mutually exclusive part of procedure's execution to the child thread. This is the point where the procedure execution *splits*. Both parent and child threads execute the part of procedure independently. When the child thread has finished its task, it *joins()* with the parent thread and afterwards the parent thread executes the rest of the procedure. Because we can easily know when the procedure is being split() and join(), we can enforce the checkpoint (Insight: I_4).

We propose that all checkpoints should be stored in HSS database. We believe that HSS database is critical component of LTE that hosts all subscribers related records and their location information, and is available at all time. The cloud operators have extra mechanisms to provide high availability of those database records. This assumption is consistent with both industry and academic research communities [52], [53], [54], [55].

Checkpointing algorithm We explain our checkpointing procedure through Algorithm 1. The checkpoint is mainly

TABLE IV
EXAMPLE OF INFORMATION TO BE SAVED DURING CHECKPOINTING

Type	Required Information
UE Sessions	Auth vector, ciphering keys, IP, APN, QoS param, Bearer & session IDs, GUTI, IMSI and GUMI, charging ID, user location, and others.
NF Configurations	MME-FQ-CSID, SGW-FQ-CSID, S1-U eNB F-TEID, S5/S8-U SGW FTEID, local IP address, local port #, protocol configs, and others.
Procedures	LTE procedures, their messages & values. E.g. auth. req/resp, create/modify session req/resp, etc.; Msgs timers, input/output values.

taken when the procedure is being *split* or *join* or certain time has passed. Every checkpoint mainly saves the following three key records (instead of taking complete VM snapshot – *discussed in issue 1*): (1) device-specific session states, (2) NF configurations, and (3) device control-plane procedures and their messages. We summarize desired checkpointing information in Table IV.

States consistencies after rollback recovery Our checkpointing rollback recovery solution can simply replay the messages one by one (Insights: I_1 and I_2). It avoids the domino effect by ensuring that all states remain consistent after failure. We have three key ideas to achieve this. First, the LTE messages to be replayed during rollback are "receive deterministic" (Insight: I_3). That is, the receiver of the replayed messages knows that these messages were executed before. Therefore, instead of actually re-executing those messages, the receiver NF simply produces the same response that was generated earlier. The second idea is that we let the alternative NF re-send the message on timeout (i.e., the orphan message will eventually be re-sent on timeout). This ensures that all of the checkpoints messages will eventually be replayed and bring EPC NFs to consistent state. The third key idea is about handling of the message that is perceived to be not sent by alternative NF. This is the case when alternative NF recovers quickly and receives the response of the message that was sent before the failure. Because the alternative NF rolls back to previous checkpoint (that does not capture that sent message), this response message brings states inconsistencies. Our solution is that the alternative NF should discard such message – the future message in the past state. This message will eventually be recovered during rollback recovery.

Checkpoint replay algorithm We explain our checkpointing procedure as shown in Algorithm 2. The *while()* loop ensures that all checkpoint messages are replayed. On each iteration (the first *for()* loop), we check whether the message was sent earlier or not. If it was not sent during rollback then this message is immediately resent, otherwise the message is sent on timeout. The second *for()* loop checks all messages that are received from the neighboring NF. If we receive the response of the message that has not been sent (according to checkpoint), then we discard the received response message. Once all the messages are replayed, the system reaches to the state at which the failure has occurred and the control-plane procedure progress has halted. This is a consistent state among all EPC NFs; and the halted procedures can start progressing again (*solving issues 2 and 3*).

Algorithm 2: Receive-Deterministic Rollback Recovery Through Checkpointing

```
An alternative NF assigned; Checkpoints are read; and UE sessions restored;
Let $CkPt$ = $UEProcMsgs$;
while all messages are not replayed from $CkPt$ do
    for each neighboring NF, $NF_j$ do
        if $CkPt_i$ is a message to be resent then
            compute SENT$_{i \to j}$ ($CkPt_i$);
            send a Rollback ($i$, SENT$_{i \to j}$ ($CkPt_i$)) message to NF$_j$;
        else if $CkPt_i$ is a message that was not sent then
            Let message timeout happen;
            compute SEND_ON_TIMEOUT$_{i \to j}$ ($CkPt_i$);
            send a Rollback ($i$, SEND_ON_TIMEOUT$_{i \to j}$ ($CkPt_i$))
            message to NF$_j$;
    end
    for every Rollback($j$, $c$) message received from a neighbor NF$_j$ do
        if RCVD$_{i \leftarrow j}$ ($CkPt_i$) > $c$ then
            Discard received message;
            compute SENT$_{i \to j}$ ($CkPt_i$);
            send a Rollback ($i$, SENT$_{i \to j}$ ($CkPt_i$)) message to NF$_j$;
    end
end
```

B. Quick LTE Failure Detection

Limitation of existing failure detection mechanism In order to provide high LTE availability, it is important that the failure is quickly detected. We refer to Section V and recall that both cloud platform and LTE standardized mechanisms are slow and can take tens of seconds to detect the failure (*issues 2 and 4*). To address this limitation, we provide our retransmission-induced quick failure detection approach.

Retransmission-induced quick failure detection Our failure detection solution exploits the LTE signalling messages for failure prediction and quick failure detection. LTE uses GTP-C signalling messages for control-plane path management, tunnel management and for mobility management [48]. We argue that these messages can also be used for quick failure detection (Insight: I_5). To take an example, when MME sends create session request to SGW, and it does not receive a response message (create session response) within a certain amount of time; it will re-send that signalling message (create session request). The retry interval is in the order of a few milliseconds [10]. MME will keep sending create session request message until the SGW failure is detected by GTP path management heartbeat timer expiration or the device times-out and re-initiates the control-plane procedure. We propose to leverage no-response of the signalling messages as a prediction of failure. The NF predicted to be experiencing a failure will be further probed to confirm the prediction; and the failover procedure can start, thereafter, if the failure has been confirmed.

Failure detection algorithm We explain our failure detection solution using Algorithm 3. Our approach is to launch a *monitoring thread*, $Tmonitor$, in every NF that monitors whether the other peer has provided the response to GTP-C signalling message or not.[4] Note that, it is sufficient to monitor the signalling messages transmission of any single device. This is because peer NF fail-stop failure for one device is the failure for all devices that are communicating with the failed

[4]To detect the failure between MME and HSS, $Tmonitor$ monitors the pending message queue that significantly increase during failure, see [58, Sec. 5.5.4].

Algorithm 3: Quick LTE Failure Detection

```
$Tmonitor$ = Monitoring thread;
$Pthread$ = Probing thread;
$Tmonitor$ finds the re-transmission of signalling message(s) from NF$_i$ to NF$_j$;
if $Tmonitor$ finds signalling re-transmission more than threshold then
    Mark NF$_j$ as failure prone;
    Report NF$_j$ to Recovery Engine;
if Recovery Engine receives a marker from $Tmonitor$ then
    Create a new thread $Pthread$;
    while $NF_j$ is not probed for certain times do
        $Pthread$ keeps probing NF$_j$;
    end
    if $Pthread$ does not receive a response from $NF_j$ for certain number of probes then
        Start Rollback recovery (Algorithm 2);
    end
```

peer NF. When monitoring thread does not find the signalling message response for certain number of retries (5 retry attempts in our implementation), it marks the non-responsive peer NF as failure-prone. *Tmonitor* then notifies the failure-prone NF to recovery engine. On receiving the failure prediction message, the recovery engine creates NF specific thread, *Pthread*, and frequently probes the failure-prone NF. If *Pthread* does not receive the response for certain number of retries then the probed NF will be declared as *failed*. Thereafter, the failure recovery will start using our rollback recovery mechanism (Algorithm 2).

VIII. Implementation

In our implementation, we use OpenAirInterface (OAI) [8], an open source LTE platform to implement EPC functionality according to 3GPP standard. The OAI provides basic implementation of EPC NFs (such as MME, SGW/PGW, and HSS) that can be deployed over Unix-based platforms like Linux, BSD or Solaris.

We modified EPC NFs *network config*, and *network interfaces* files to ensure MME should be able to communicate with more than one SGW NF to perform fail-over. We modify the SCTP implementation to enable SCTP maximum number of retransmissions, timeouts according to Cisco implementation [6]. We enhance the path management implementation of OAI by ensuring the retransmission of packets are done according to GTP-C tunneling protocol [48], and use the GTP-C timer values as defined by Cisco implementation [49].

Implementation of Proposed Design: Our implementation has a recovery engine that consists of (a) *checkpoint manager*: prepares an alternative NF and asks that NF to retrieve the desired checkpoint during failover, and (b) *failure detector*: runs *Pthread* and probes the NF which is predicted to be failed. We implement a separate middleware (we call it stub) along side EPC NFs. The stub has three main components (i) *checkpoint agent*: responsible for taking checkpoints, (ii) *local monitor*: runs the failure prediction thread, *Tmonitor*, and (iii) *virtual replayer*: generates the messages from the cache when its peer NF is performing rollback recovery.

IX. Evaluation

We evaluate the fault-tolerance mechanisms of our proposed approach. We run our tests and collect experiment results on Ubuntu Server 14.04.3 LTS as virtualized instance running on Intel Core i7 3517U Processor x 2 with maximum turbo frequency of 3.00 GHz, 4 MB SmartCache and 6 GB DDR3 RAM. We discuss our results regarding (1) failure detection, (2) failure recovery, (3) overhead. Finally, we briefly discuss overall failure recovery and high availability of our design.

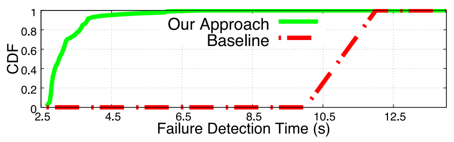


Fig. 7. Failure detection time comparison.

TABLE V
Relaxing the Failure Prediction as Retransmission Time in Path Management Leads to No False Positive

Criteria	False Positive
Taking path management retransmissions values as it as	12.4%
Relaxing path management retransmissions values	3.8%
Using path management retransmissions values with backoff	0%

A. Failure Detection

How quickly we detect the failure? To find out how quickly our approach can detect the failure, we initiate the control-plane procedures and let one of the EPC NF crash. The monitoring thread (*Tmonitor*) counts five number of retransmissions and reports the failure prediction to recovery engine. The probing thread (*Pthread*) at recovery engine probes the failure-prone NF at fixed interval of 500 milliseconds for 5 number of retries. Thereafter, the failure is confirmed. Figure 7 shows that on average our approach can detect the failure in about 2.7 seconds; and in 3.5 seconds for 80% of the cases when compared to the baseline approach (implementing 3GPP standardized timers and operations). At worst case, it takes about 4.9 seconds to detect the failure. This is mainly due to the varying retransmission in LTE path management due to network congestion (i.e., echo request and response take longer if one of the peer NF is severely congested).

False positives? Our failure prediction relies on the number of re-transmissions between two EPC NFs (we consider 5 retransmissions to be treated as failure prediction). This retransmission value is calculated from LTE path management technique.[5] If we consider the number of retransmissions as it is (calculated based on path management value), our approach can bring up to 12.4% of false positives,[6] as shown in Table V. However, when we relax our failure prediction by doubling the time retransmission is being happened then we can significantly reduce the false positives to 3.8%. We can reduce false positives to zero if we exponentially back-off the time retransmission is being happened in order to predict the failure. In our result, we have achieved zero false positive on 2$^{\text{nd}}$ time back-off.

[5]echo-req/echo-resp value plus some alpha (we set it to be 20% of echo-req/echo-resp value).

[6]We observe these false positives when our EPC NF is overloaded and some of the response messages take unusually long time.

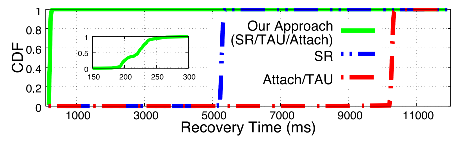


Fig. 8. Failure recovery time comparison.

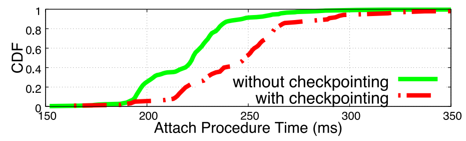


Fig. 9. Time overhead of keeping checkpoints.

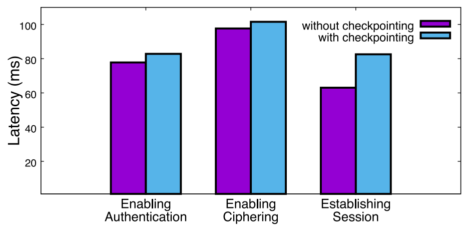


Fig. 10. Breakdown of latency overhead for keeping checkpoints.

B. Failure Recovery

How quickly we recover from failure? Figure 8 shows the comparison of failure recovery between our approach and the baseline when SR and Attach/TAU control-plane procedures are used. In baseline, the device's control-plane procedure always times-out – costing 5 seconds for SR and 10 seconds for Attach/TAU case – before recovering the failure (through re-attach). In contrast, our approach can take a few hundred milliseconds to recover from the failure by using our rollback recovery mechanism. In our approach, the recovery engine instructs the alternative VM (in cold-standby) to read the logs from the stable storage, and then the alternative NF simply replays the logs. Because these logs are being replayed using virtual replayer, it does not incur radio transmission delays during failure recovery.

1) Negligible Overhead: **Checkpointing operation overhead during normal working** Our design is required to take periodic checkpointing so that it could rollback and replay the checkpoint logs during failure recovery process. We show that our checkpointing approach does not have much overhead on LTE control-plane procedures before failure. Figure 9 shows the overhead for Attach Request control-plane procedure. The reason we consider Attach Request is that attach procedure requires more number of signalling messages exchange compared to SR and TAU. As we can see from Figure 9, our checkpointing approach only costs up to 50 milliseconds of overhead for LTE control-plane procedure. The overhead of 50 milliseconds is small when compared to overall timeout value of 5,000 milliseconds for SR, and 10,000 milliseconds for Attach/TAU procedures.

Latency breakdown during checkpointing Figure 10 shows the latency break down as micro benchmark for checkpointing before failure. We consider the authentication process (that includes the rounds of communication with HSS to retrieve authentication vector), ciphering process, and session creation process (that includes assigning IP address and setting-up QoS, etc.). We can see from Figure 10 that on average our checkpointing approach just adds tens of milliseconds (i.e., 5% to 30%) more processing while executing LTE control-plane procedure.

X. Conclusion

This paper studies the high availability of LTE control-plane procedures in NFV. Our paper addresses three critical challenges in providing high availability in LTE: fast failure detection, recovering the lost states and messages, and incurring less overhead. By leveraging LTE specific design insights we have solved these challenges. The failure is detected through LTE packets retransmission and probing, only when it is required. The failed control-plane procedures are restored through checkpointing and rollback recovery mechanisms.

References

[1] "Cisco ASR 5500: Elastic Packet Core Network From Cisco." [Online]. Available: https://www.cisco.com/c/en/us/products/wireless/asr-5500/index.html (Accessed: May 28, 2022).
[2] "Ericsson Blade System (EBS) for EPC." [Online]. Available: https://www.ericsson.com/ourportfolio/digital-services-solution-areas/sgsn-mme?nav=fgb_101_09%7Cfgb_101_256 (Accessed: May 28, 2022).
[3] M. T. Raza and S. Lu, "Enabling low latency and high reliability for IMS-NFV," in *Proc. IEEE CNSM*, 2017, pp. 1–9.
[4] A. Carbonari and I. Beschasnikh, "Tolerating faults in disaggregated datacenters," in *Proc. ACM HotNets*, 2017, pp. 164–170.
[5] "Apache ZooKeeper." [Online]. Available: https://zookeeper.apache.org (Accessed: May 17, 2022).
[6] "CISCO—SCTP Parameter Template Configuration Mode Commands." [Online]. Available: https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/20/CLI/books/R-Z/20_R-Z_CLI-Reference/20_R-Z_CLI-Reference_chapter_010110.pdf (Accessed: May 28, 2022).
[7] G. Cao and M. Singhal, "On coordinated checkpointing in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 12, pp. 1213–1225, Dec. 1998.
[8] "Open Source LTE Platform." [Online]. Available: http://www.openairinterface.org (Accessed: May 28, 2022).
[9] H. Jiang, N. Choi, M. Thottan, and J. Van der Merwe, "FestNet: A flexible and efficient sliced transport network," in *Proc. IEEE 7th Int. Conf. Netw. Softwarization (NetSoft)*, 2021, pp. 97–105.
[10] M. T. Raza, D. Kim, K.-H. Kim, S. Lu, and M. Gerla, "Rethinking LTE network functions virtualization," in *Proc. IEEE ICNP*, 2017, pp. 1–10.
[11] Z. A. Qazi, M. Walls, A. Panda, V. Sekar, S. Ratnasamy, and S. Shenker, "A high performance packet core for next generation cellular networks," in *Proc. ACM SIGCOMM*, 2017, pp. 348–361.
[12] M. T. Raza, H.-Y. Tseng, C. Li, and S. Lu, "Modular redundancy for cloud based IMS robustness," in *Proc. ACM MobiWac*, 2017, pp. 75–82.
[13] M. T. Raza, F. M. Anwar, D. Kim, and K.-H. Kim, "Highly available service access through proactive events execution in LTE NFV," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 2531–2544, Sep. 2021.
[14] G. Ilievski and P. Latkoski, "Experimental evaluation of network packet latency within a distributed NFV infrastructure," in *Proc. 29th Telecommun. Forum (TELFOR)*, 2021, pp. 1–4.
[15] "Open EPC—Software LTE Implementation." [Online]. Available: http://www.openepc.net/ (Accessed: May 28, 2022).
[16] "OpenLTE: Open Source LTE Implementation." [Online]. Available: http://openlte.sourceforge.net (Accessed: May 28, 2022).
[17] "OpenStack Open Source Cloud Computing Software." [Online]. Available: https://www.openstack.org/software/ (Accessed: May 17, 2022).
[18] "Open Platform for NFV (OPNFV)." [Online]. Available: https://www.opnfv.org/ (Accessed: May 17, 2022).

[19] "NF Virtualization Using VMware's vSphere: The Efficient and Secure Platform." [Online]. Available: https://www.vmware.com/products/vsphere/features/ (Accessed: May 17, 2022).

[20] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[21] "Bringing Network Function Virtualization to LTE." Nov. 2014. [Online]. Available: http://www.4gamericas.org/files/1014/1653/1309/4G_Americas_-_NFV_to_LTE_-_November_2014_-_FINAL.pdf

[22] "Boost Agility with Virtual Networks that Deliver Quickly on Demand." [Online]. Available: http://www.verizonenterprise.com/products/networking/sdn-nfv/virtual-network-services/ (Accessed: May 28, 2022).

[23] "Exploring the Opportunities in NFV." [Online]. Available: http://www.vodafone.com/business/group-enterprise/software-defined-networking-and-network-function-virtual (Accessed: May 28, 2022).

[24] "SK Telecom Creates Its Own in-House NFV MANO." [Online]. Available: https://techblog.comsoc.org/tag/sk-telecom/ (Accessed: May 28, 2022).

[25] B. H. Nguyen *et al.*, "A reliable distributed cellular core network for public clouds," Microsoft Res., Redmond, WA, USA, Rep. MSR-TR-2018-4, Feb. 2018.

[26] "High Availability is More Than Five Nines." [Online]. Available: https://archive.ericsson.net/service/internet/picov/get?DocNo=10/28701-FGB101256&Lang=EN&HighestFree=Y (Accessed: May 28, 2022).

[27] A. Elmokashfi, D. Zhou, and D. Baltrunas, "Adding the next nine: An investigation of mobile broadband networks availability," in *Proc. ACM Mobicom*, 2017, pp. 88–100.

[28] "Alcatel-Lucent 5620 SAM." [Online]. Available: http://www.pexx.net/pdfs/whitepapers/alcatel_lucent/mpr9500/EPC_Solution_wp_0309.pdf (Accessed: May 28, 2022).

[29] "Huawei eCNS300 Solution for EPC." [Online]. Available: http://m.huawei.com/enmobile/enterprise/products/wireless/gsm-r/gsm-r/hw-200203.htm (Accessed: May 28, 2022).

[30] J. Sherry *et al.*, "Rollback-recovery for middleboxes," in *Proc. ACM SIGCOMM*, Aug. 2015, pp. 227–240.

[31] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford, "Heading off correlated failures through independence-as-a-service," in *Proc. OSDI*, 2014, pp. 1–19.

[32] X. Foukas, M. K. Marina, and K. Kontovasilis, "Orion: RAN slicing for a flexible and cost-effective multi-service mobile network architecture," in *Proc. ACM Mobicom*, 2017, pp. 127–140.

[33] "Alcatel-Lucent Virtualized EPC Delivering on the Promise of NFV and SDN." [Online]. Available: http://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2014/10743-alcatel-lucent-virtualized-epc-delivering-the-promise-nfv.pdf (Accessed: May 17, 2022).

[34] "Ericsson Virtual Router." [Online]. Available: https://archive.ericsson.net/service/internet/picov/get?DocNo=1/28701-FGB1010557&Lang=EN&HighestFree=Y (Accessed: May 28, 2022).

[35] "OPNFV—Building Fault Management into NFV Deployments." [Online]. Available: https://www.opnfv.org/wp-content/uploads/sites/12/2016/11/opnfv_faultmgt_final.pdf (Accessed: May 28, 2022).

[36] M. Kablan, A. Alsudais, E. Keller, and F. Le, "Stateless network functions: Breaking the tight coupling of state and processing." in *Proc. NSDI*, 2017, pp. 97–112.

[37] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico replication: A high availability framework for middleboxes," in *Proc. ACM Symp. Cloud Comput.*, 2013, p. 1–15.

[38] "Configuring the Compute Node—Work is in Progress." [Online]. Available: https://docs.openstack.org/ha-guide/compute-node-ha.html (Accessed: May 28, 2022).

[39] "Understanding VM Snapshots." [Online]. Available: https://kb.vmware.com/s/article/1015180 (Accessed: May 28, 2022).

[40] "VMWare Virtual Machines Become Unresponsive for Over 30 Minutes." [Online]. Available: https://kb.vmware.com/s/article/2039754 (Accessed: May 28, 2022).

[41] J. Kim, K. Salem, K. Daudjee, A. Aboulnaga, and X. Pan, "Database high availability using SHADOW systems," in *Proc. ACM Symp. Cloud Comput.*, 2015, pp. 1–13.

[42] "Report on scalable architectures for reliability management," ETSI, Sophia Antipolis, France, document GS NFV-REL 002, 2017.

[43] "Network functions virtualisation (NFV); reliability; report on models and features for end-to-end reliability," ETSI, Sophia Antipolis, France, document GS NFV-REL 003, 2017.

[44] *Non-Access-Stratum (NAS) Protocol for Evolved Packet System (EPS); Stage 3*, 3GPP Standard TS 24.301, Jun. 2013. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/24301.htm

[45] C. Guidi, I. Lanese, F. Montesi, and G. Zavattaro, "Dynamic error handling in service oriented applications," *Fundamenta Informaticae*, vol. 95, no. 1, pp. 73–102, 2009.

[46] *General Packet Radio Service (GPRS) Enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Access*, 3GPP Standard TS 23.401, 2011. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23401.htm

[47] "Study of evolved packet core (EPC) nodes restoration," 3GPP, Sophia Antipolis, France, Rep. TR 23.857, 2012.

[48] *Tunnelling Protocol for Control Plane (GTPv2-C)*, 3GPP Standard TS 29.274, 2014.

[49] "CISCO—Troubleshooting GTPC and GTPU and Associated Path Failures." [Online]. Available: https://www.cisco.com/c/en/us/support/docs/wireless/asr-5000-series/200026-ASR-5000-Series-Troubleshooting-GTPC-and.html (Accessed: May 28, 2022).

[50] *GPRS Enhancements for E-UTRAN Access*, 3GPP Standard TS 23.401, 2011.

[51] "Making Nova Database Highly Available with Pacemaker." [Online]. Available: https://wiki.openstack.org/wiki/HAforNovaDB (Accessed: May 28, 2022).

[52] R. Aranha, P. Tuck, J. E. Miller, C.-P. Wang, M.-A. Neimat, and S. S. Cheung, "Database system with active standby and nodes." U.S. Patent 20 080 222 159 A1, 2008.

[53] W. Lang, F. Bertsch, D. J. DeWitt, and N. Ellis, "Microsoft azure SQL database telemetry," in *Proc. ACM Symp. Cloud Comput.*, 2015, pp. 189–194.

[54] W. Lin, "StreamScope: Continuous reliable distributed processing of big data streams," in *Proc. USENIX NSDI*, vol. 16, 2016, pp. 439–453.

[55] Y. Zhang, J. Yang, A. Memaripour, and S. Swanson, "Mojim: A reliable and highly-available non-volatile memory system," *ACM SIGARCH Comput. Archit. News*, vol. 43, no. 1, pp. 3–18, 2015.

[56] R. Koo and S. Toueg, "Checkpointing and rollback-recovery for distributed systems," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 1, pp. 23–31, Jan. 1987.

[57] J. L. Kim and T. Park, "An efficient protocol for checkpointing recovery in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 8, pp. 955–960, Aug. 1993.

[58] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn, "Diameter base protocol," Internet Eng. Task Force, RFC 6733, 2012.