Highly Available Service Access Through Proactive Events Execution in LTE NFV

Muhammad Taqi Raza, *Member, IEEE*, Fatima Muhammad Anwar, *Member, IEEE*, Dongho Kim, and Kyu-Han Kim

***Abstract*—The explosion of mobile applications and phenomenal adoption of mobile connectivity by end users make all-IP based 4G LTE as an ideal choice for providing Internet access on the go. LTE core network which handles device control-plane and data-plane traffic becomes susceptible to network resource constraints. To ease these constraints, Network Function Virtualization (NFV) provides high scalability and flexibility by enabling dynamic allocation of LTE core network resources. NFV achieves this by decomposing LTE Network Functions (NF) into multiple instances. However, LTE core network architecture which is designed considering fewer NF boxes does not fit well where decomposed NF instances add delays in network event execution. Certain control-plane events being time critical hurt data-plane traffic requirements defined by LTE standard. This paper proposes Fat-proxy which acts as a stand-alone execution engine of these critical network events. Through space uncoupling, we execute several signalling messages in parallel while skipping unnecessary messages to reduce event execution time and signalling overhead while ensuring highly available service access. We build our system prototype of open source LTE core network over virtualized platform. Our results show that we can reduce event execution time and signalling overhead upto 50% and 40%, respectively.**

***Index Terms*—Availability, 4G mobile communication, network function virtualization.**

I. Introduction

ONE OF the leading use cases for carrier network virtualization is the LTE core network (also known as LTE Evolved Packet Core – EPC) [1], [2], [3]. Service providers, particularly mobile operators planning for voice over LTE (VoLTE), Evolved Multimedia Broadcast/Multicast Service (eMBMS), and numerous Internet of Things (IoT) applications support, are looking towards Network Functions Virtualization (NFV) to scale services up and down quickly and to better align costs with network usage [4], [5], [6].

In this paper, we analyze the impact of virtualization on EPC functionality and service provisioning. We find that current LTE EPC architecture which is designed for fewer powerful dedicated NFs does not fit well when thousands or even hundreds of NFs are chained within one EPC.

Challenges and impact: We discover following two major challenges on virtualizing EPC.

Virtualized network is not designed for LTE: LTE EPC is virtualized over data-center (DC) network which suffers from long queueing delays in switches [7], [8], packet losses [9], [10], timed out retransmissions [11], [12], and out of order packets delivery [13]. Because of these characteristics, virtualized NFs (VNFs) implemented over commodity DC network only provide flow level guarantees [14], whereas LTE standard requires packet level guarantees (100ms and 300ms delays for voice and data packets, respectively) [15]. This significantly degrades user quality of service (QoS) and even causes temporary service unavailability when active sessions drop.

EPC is not designed for virtualized network: In legacy EPC, there are fewer NF boxes, which are connected through dedicated fiber links. The round-trip-time (RTT) over one-hop link is stable, and determines NF reachability and packet retransmission counters [16], [17]. In virtualized EPC implementation, some network signalling packets take longer and congested path triggering unnecessary packet retransmission at sender. Further, it causes domino effect by triggering time out at other chained NFs. This higher signalling failure rate while executing certain network events [18] have direct impact on user traffic (e.g., voice and data) continuity. We regard these events as mission critical events.

Goals: Our goal is to ensure high availability of LTE services. To achieve this goal, we want to protect mission critical events from delay and failure. We categorize these events being *handover* event during device mobility, *paging* event during device idle mode, and *service request* for gaining network resources; because these 3 events cause 50% of all network signalling [19], therefore, we aim to isolate them to reduce networking signalling load at EPC. We also want our solution design to be applicable in 5G cellular network.

Design: Our design is motivated by the fact that during execution of mission critical events, no other event will be handled by EPC. Also, their operations are mutual exclusive even within one NF [20]. Therefore, we first decompose these events from EPC and implement them separately as a

Manuscript received September 26, 2020; revised January 30, 2021 and May 4, 2021; accepted June 23, 2021. Date of publication August 9, 2021; date of current version September 9, 2021. The associate editor coordinating the review of this article and approving it for publication was E. Oki. *(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).

Fatima Muhammad Anwar is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01002 USA (e-mail: fanwar@umass.edu).

Dongho Kim is with the Research Department, AT&T Labs, Palo Alto, CA 94301 USA.

Kyu-Han Kim is with the Networking and Mobility Research Group, Hewlett Packard Labs, Palo Alto, CA 94304 USA.

Digital Object Identifier 10.1109/TNSM.2021.3103160

Fat-proxy. As the name suggests, the Fat-proxy acts as an execution engine to an event. When an EPC receives event request, it forwards the request to particular Fat-proxy – that takes responsibility of executing the event and finally flushes the updated event status and device session information to EPC. In other words, all event execution logic being local to one virtual machine (VM) does not only address above mentioned challenges but also keeps greater number of signalling messages flow away from EPC.

In our design we also address several challenges such as identifying event specify logic from chained NFs, and resolving functional dependencies while implementing Fat-proxy, etc.

Moreover, we use space uncoupling and further decompose mutual exclusive modules within one NF. Then we execute those signalling messages in parallel that communicate with mutual exclusive modules. However, we merge the results from these parallel signalling execution in consistent manner and support transaction roll back in case of failure. Further, the concept of Fat-proxy, that acts as a stand-alone execution engine of critical messages, is also applicable in 5G core network.

Results: We gather results from our OpenEPC implementation over virtualized platform. We use network traces to match actual capacity of NF in real operator network. Our result shows that (1) Fat-proxy reduces more than 40% signalling load, (2) reduces upto 50% event execution time, and (3) generates upto 40% less signalling messages by skipping and parallelizing these messages.

Related Work: Our work is in contrast to other efforts in EPC virtualization or in similar other directions. Reference [21], [22] and [23] decomposes edge router functions, gateway functions, and network services, respectively into control-plane and data-plane. However, our work addresses those key control-plane functions that affects user services. References [24], [25] make use of software and hardware choices to meet specific service demand. Our work instead aims to provide solution under dynamic control-messages flow. It is achieved by delegating event execution responsibility to the fat-proxy, which is transient in nature, i.e., on successful event execution the user states are deleted.

Summary: In summary our paper makes following contributions.

- We identify key limitations in EPC deployment over commodity DC network to ensure high LTE service availability. To best of our knowledge, this is first paper on EPC virtualization that highlights EPC implementation-specific issues.
- We identify and address only those events which affect user traffic, which is first effort in this direction, as best of our knowledge.
- Our design works purely for commodity DC network – not requiring express routes and non-commodity powerful servers. Moreover, our design acts as plug and play, where Fat-proxy VM being LTE standard compliant can be installed with any network operator's virtualized network.
- Our design controls exponentially growing signalling messages [26] by keeping them away from EPC.

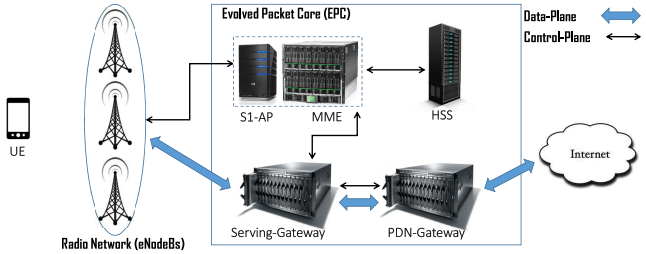

Fig. 1. LTE architecture: an overview.

- Our results show an improvement of upto 50% in number of signalling messages, event execution time, and EPC traffic load without introducing any performance bottlenecks.

Organization: Section II provides background on LTE architecture, followed by Section III which motivates the problem and defines the problem scope. Section IV introduces challenges in virtualizing EPC core. Section V talks about design rationale and explains our system design in detail. Section VI discusses implementation, Section VII provides evaluation results, and Section VIII stimulates discussion. Section IX compares our work with the related work, and Section X concludes the paper.

II. Background: LTE Architecture

LTE network consists of three main components, which are User Equipment (UE), Evolved Node Base-station (eNodeB), and Evolved Packet Core (EPC), as shown by 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, private corporate networks or the IP Multimedia Subsystem (IMS) and facilitates user communication. LTE EPC comprises over a number of LTE Network Functions (NFs), that include Mobility Management Entity (MME), HSS (Home Subscriber Server), Serving Gateway (SGW), Packet Data Network Gateway (PGW), Policy and Charging Rules Function (PCRF), and few others. These NFs handle control-plane and data-plane traffic through separate network interfaces. As shown in Figure 1, 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 authenticate and authorizes UE, handles network events (such as device Attach, Handover, Service provisioning, and Paging events), and maintains SGW and PGW connections for data-plane traffic.

EPC NFs are static in nature and are connected, or chained, in a certain way that achieve desired overall functionality or service that LTE network is designed to provide. These NFs exchange a number of control messages to execute a specific network event. For example, during device Attach event, MME obtains device security keys from HSS, authenticates the device, creates device session information at SGW and PGW. Then SGW and PGW establish data bearer connection with the device and configure specific QoS profile. Thereafter, the device is said to be registered with LTE network. The delay or failure in one control-message results into complete event

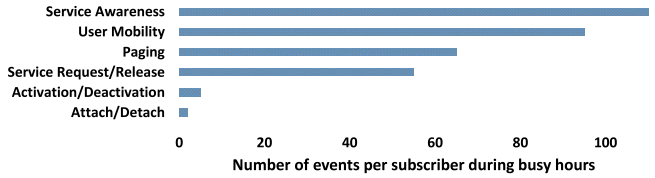

Fig. 2. Number of events per user during busy hours.

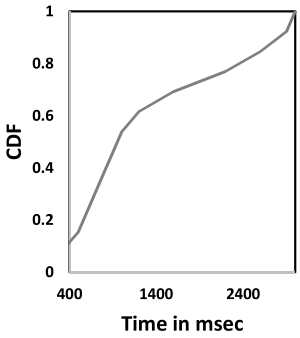

Fig. 3. Event completion time (average) during busy hours in operational EPC network.

failure [20]. Therefore, NFs which are implemented over vendor specific software and hardware guarantee per signalling message level reliability and NFs high availability support.

III. Motivation and Problem Scope

vEPC has already been embraced by a number of operators, such as SK telecom in Korea [27] and Spark telecom in New Zealand [28]. In other countries, it is currently being rolled out, such as Docomo in Japan [29], and Telefonica in Spain [30]. Yet others are considering to deploy vEPC in near future, such as AT&T [31] in USA and China mobile in China [32]. vEPC has potential benefits: (1) NFV based EPC can achieve high scalability and high flexibility to quickly scale services up and down [3], [33], and (2) reduce network expenses (both capital expenditures (CAPEX) and operational expenditures (OPEX)) [34], [35]. However, we argue that EPC implementation over general purpose boxes expose a number of issues which otherwise are suppressed by carrier grade NF boxes (e.g., Ericsson Blade System (EBS) [36], Alcatel-Lucent Wireless Mobility Manager [37] and others). We identify two major issues, i.e., (1) controlling signalling storm during peak hours, and (2) timely execution of mission critical events, that require immediate attention in vEPC implementation.

Controlling network signalling storm: LTE devices frequently interact with LTE network to execute their events. These events are *Device Attach, Service Request and Release, Handover, Paging, Bearer Activation, Modification and Deactivation, Detach Request*, and many others. Out of these, some events are executed more frequently than others. As shown in Figure 2, *Handover* event is executed at least 50 times more than device *Attach* incident during busy hours [19]. Moreover, to execute one such event, different NF components interact with each other and generate a greater number of signalling messages. Some events produce more number of signalling message than others. For example, one *handover* event generates 32 signalling messages compared to *paging* event that produces only 6 signalling messages. When all events are combined from all devices during busy hours, a signalling storm is generated at EPC NFs.

Therefore, we are motivated to provide a solution that controls the signalling storm at LTE core without restricting devices network access (the solution operational LTE networks use to control signalling storm [38]).

Administrating mission critical events: Although, delay or failure of one signalling message can jeopardize whole network event, all network events are not equal. Some network events have stronger execution constraint than others. For example, failure of one signalling message during device *Attach* operation triggers *Re-Attach* that does not impact user quality of experience. Whereas, failure of one signalling message during device *Handover* operation aborts mobile connection and terminates any device voice and data sessions. Our preliminary study on LTE operational network discloses that during rush hours average events completion time at EPC is significantly high, reaching upto 3 seconds (as shown in Figure 3).[1] This higher latency directly affects user QoS experience, resulting from Voice over LTE (VoLTE) call drop and voice jitter, to affecting TCP based services. Table I shows how event completion delays exceeding LTE standard defined QoS deadlines [15] affect user services. Therefore, in this work, we are also motivated to provide timely execution of mission critical events, even during higher service load at LTE NFs.

Defining mission critical events: We categorize those events being mission critical whose delay or failure has direct impact on ongoing user services (i.e., voice, data, and multimedia services). These events are:

- *Handover event* that ensures seamless user traffic flow during user mobility.
- *Paging event* wakes device from idle state when voice/data traffic is pending at LTE network.
- *Service Request event* provides on-demand network resources to device.

Interestingly, these 3 events make-up 50% of all network signalling traffic [19]. Therefore, by addressing these events, we not only ensure timely execution of user service sessions but also address highly occurring network signalling messages.

Assumptions: This work neither assumes special DC network topology and high performance server boxes nor requires changes in LTE standard. We address timely execution of important events on commodity DC network (with no dedicated/express links) while obeying LTE standard to provide plug and play solution for any carrier network. We do not assume overprovisioning approach of hot standby – an approach taken by most of the network operators to ensure high availability [39].

[1]We gather LTE traces at device and ignore radio retransmissions (at both MAC and RLC LTE layers) and also excluded device and radio RTT from results.

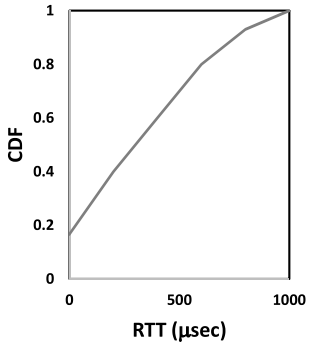

Fig. 4. RTT fluctuation (average) in virtualized network.

TABLE I
LTE PACKET DEADLINES FOR QOS AND IMPACT WHEN SUCH DEADLINES ARE NOT MET

QCI	Priority	Packet Delay	Packet Loss	Traffic Example	Observations
5	1	100ms	10^{-5}	IMS signalling	Call drop. Device switches to 3G for voice call
1	2	100ms	10^{-2}	VoLTE	Call drop. Voice Jitter
8	8	300ms	10^{-5}	TCP based services	TCP retransmission timer time-out

IV. CHALLENGES IN VIRTUALIZING LTE-EPC

We discover multiple challenges from our implementation experience of LTE-EPC virtualization, and from our study on LTE standard documents and virtualized network infrastructure.

A. On Data-Center Network Characteristics

NFV envisions the implementation of NFs as software-only entities that run over NFV infrastructure. NFV infrastructure consists of commodity servers that run Virtualized Network Function (VNF) over cloud platforms, such as OpenStack [40], OPNFV [41], CloudStack [42], OpenNebula [43] and others. In contrast to legacy LTE NFs implementation, NFV implementation introduces a number of changes. First, unlike traditional NFs which are connected over single hop, these VNFs may be located over multiple hops. Therefore, long queueing delay in switches introduces high latency [7], [8], [44]. Second, during high DC utilization, packet loss probability increases [9], [10] that can adversely affect traffic flows where the loss of an ACK may cause TCP to perform a timed out retransmission [11], [12]. Third, DC network traffic exploits the inherent multi-path nature of DC networks [44], [45] that causes out of order packet delivery [13]. Fourth, DC network designed to meet application deadlines which provide mechanisms to meet traffic flow deadlines (e.g., mice flows), rather than per packet guarantees [14], [46].

In short, DC network is designed to meet Service Level Agreement (SLA) by protecting execution bounds on traffic flows. However, in LTE, service guarantees are made by timely execution of mission critical events.

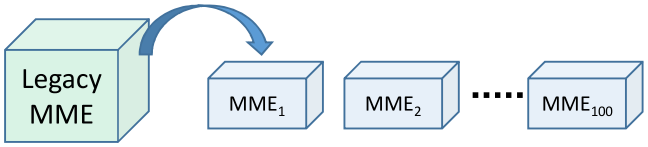

Fig. 5. One legacy MME is decomposed into multiple virtualized MME instances.

B. On Inter-VNF Delay

LTE-NFV framework provides the flexibility and network scalability by decomposing original NFs into multiple VNFs [3], [33]. However, in order to ramp up the original capacity of NF, multiple VNF instances are desirable. For example, we need hundreds (if not thousands) of MME-VNF instances implemented over commodity servers in order to facilitate 10 millions subscribers as supported by conventional MME function [47], [48]. As shown in Figure 5, legacy MME is decomposed into multiple MME instances, where each MME holds the profiles of subset of customers. These VNF instances are distributed within data center. Ideally, related EPC VNF instances (e.g., MME, SGW, PGW etc.) are placed within the same rack that eliminates network delays between two EPC NFs. However, during mobility, device switches to target eNodeB – connected to different MME instance. As a result, the device session migrates from its source MME to target MME during handover. Thereafter, new serving MME and rest of old serving EPC NFs end up residing at different racks. Now network delays play important role on timely execution of network signalling messages. We find that LTE-NFV framework is not able to cope with varying delays among different VNF instances because of following reasons.

Expiry of a timer at any NF may lead to event failure: LTE was designed for fewer EPC NFs which are directly connected over dedicated fiber link. Therefore, in legacy LTE network, the variation in RTT values is negligible. This motivates LTE network designer to use RTT for two purposes (1) path management, and (2) calculating message retransmission timer, as given in equation 1, between a pair of EPC NFs.

$$RetransmissionTimer = RTT * \alpha$$
$$where: \alpha = anoffset < RTT. \quad (1)$$

Path management: As a matter of fact, all EPC NFs and the connection between them must always be active to serve users. To determine that a peer NF is active, the NFs exchange echo-request and echo-response messages [16], [17]. This exchange of the echo-request and echo-response messages between two NFs allows for quick detection if a path failure occurs [16].

Retransmission timer: Moreover, echo-request and echo-response also helps calculating packet retransmission time at EPC NF. Retransmission timer is calculated based on RTT measurements (i.e., time between echo-request and echo-response) [49]. Although, such timer value incorporates arbitrary RTT value delays, it does not include larger RTT value variations because network communication delay does not occur for one-hop legacy LTE NFs.

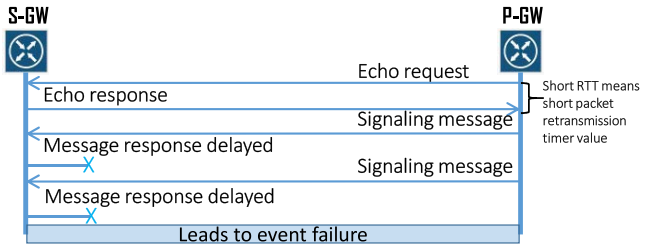

Fig. 6. Signalling messages expire pre-maturely when timer based on RTT value is too short.

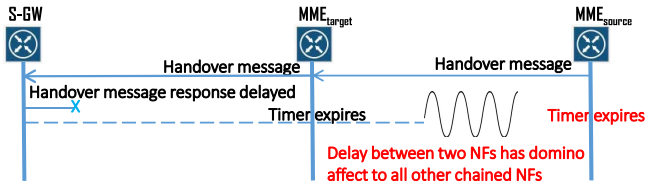

Fig. 7. Expiry of timer between two NFs has a domino effect that leads to an event failure.

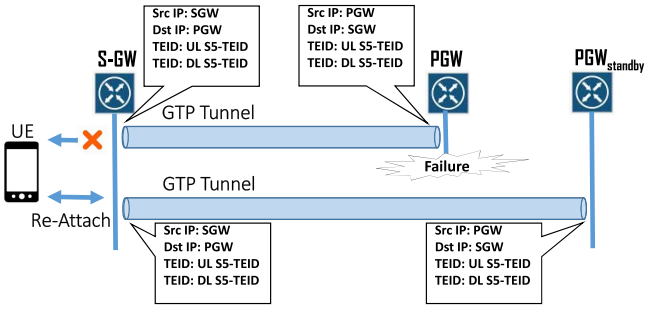

Fig. 8. GTP Tunnel is established between two NFs and remains active throughout device registration lifetime. NF failure can tear down GTP tunnel that results into device de-registration from the network.

However, virtualized EPC implementation needs to address significant RTT variations. DC topologies' network link redundancy provides multiple paths for each pair of NF [44], [45], [50]. That mean, for each RTT calculation echo-request and echo-response packets may traverse through two different paths. This can potentially cause a significant variation between two subsequent RTT measurement readings. To make things worse, DC network congestion can cause RTT spikes up to tens of milliseconds [45], [51] that makes EPC retransmission timer calculation even harder. Figure 4 shows variation of RTT values in a virtualized network [44]. The RTT varies from few microseconds to 1000 microseconds under normal network load. This 1000X RTT difference converts into 1000 different timer values. When a NF selects smaller timer value based on smaller RTT value, the signalling messages from that NF are unnecessarily retransmitted, as shown in Figure 6. This unnecessary signalling messages retransmission lead to overall delay in event execution, and at times expiry of event-timer running at device that results into event failure.

Expiration of a timer has a domino effect: For one event execution, multiple EPC NFs are chained such that one NF output is an input of second NF, and so on. For example, in *handover* event execution, signalling messages are exchanged between 5 different NFs (i.e., source MME, source SGW, target MME, target SGW, and PGW). Each pair of NF is running a different retransmission timer value. When one timer value expires, it produces a domino effect that causes expiration of timer to preceding NF. This has been shown in Figure 7, where source MME sends handover signalling message (e.g., *Forward Relocation Request message*) to a target MME. Target MME sends another handover message (e.g., *Create Session Request message*) to SGW. But even in the presence of mild network congestion, the response from SGW is delayed that results into expiration of timer at target MME. Because source MME is waiting for a response from target MME, eventually the timer value at source MME also expires. This can potentially create a domino effect to chained NFs for *handover* event execution.

In short, EPC by design is not only sensitive to network delays but also does not tolerate any delay variance. However, it is challenging to provide both constant and smaller network delay in virtualized DC network, where packets may face network congestion and take multiple packet traversal paths.

C. On Fault Tolerance

On device registration, MME establishes GTP (GPRS Tunneling protocol) tunnel with SGW which thereafter creates GTP tunnel with PGW. GTP tunnel encapsulates user data and carries bearer specific signalling traffic between pair of core network NFs, and maintains device connectivity during mobility. It recognizes each user by assigning unique UL and DL Tunnel Endpoint Identifiers (TEIDs) at both end of tunnel. As shown in Figure 8, GTP tunnel is first established between SGW and PGW for a particular UE. When sending user traffic to PGW, SGW inserts a GTP header with source address = SGW, destination address = PGW, and TEID = S5 TEID (UL), and forwards the packet to PGW. Similarly, for DL user traffic, PGW inserts a GTP header with source address = PGW, destination address = SGW, and TEID = S5 TEID (DL), and forwards the packet to SGW. GTP tunnel remains active throughout the device registration with the network, and does not change until LTE S1-handover is initiated towards other EPC NF; otherwise device is deregistered from the network [52]. Such LTE design implies NFs failures must be handled locally without breaking the tunnel end-points.

Virtualized NFs break user connection during faults: LTE-NFV which is implemented using cloud platform over DC Network relies on cloud-platform recovery procedures to provide LTE fault tolerance. In all cloud-platforms, such as OpenStack [40], OPNFV [41], CloudStack [42], OpenNebula [43] and others, high availability is implemented with redundant hardware running redundant instances of each service. If one piece of hardware running one instance of a service fails, the system can then fail-over to use another instance of a service that is running on hardware that did not fail [53]. Such fault tolerance procedure does not work for LTE that requires faults must be handled locally by maintaining tunnel end-points between two NFs. Moreover, any engineering effort to address this issue is futile where current cloud systems being Infrastructure as a Service (IaaS) do not provide instance-level

fault tolerance anyway. Therefore, standby servers do not help to keep GTP tunnel in-tact during faults and result into failure propagation to user (where user is required to re-attach with LTE network).

V. System Design

At high level our design first identifies those execution modules (i.e., functions) located at different EPC NFs, that are executed in response to a critical network event. Then it decouples these modules from their respective NFs and chain them together as a separate NF. Now this new event specific NF becomes a stand-alone execution engine without requiring interactions between different EPC NFs. When a critical event triggers, MME labels that event critical and directs its execution to that event specific NF. MME also sends device session information to facilitate the event execution. Thereafter, the event specific NF executes the event and populates the updated user session information to MME after successful event execution. Although our proposed design is explained in the context of LTE, the proposed design will also work in 5G cellular network that share the same mission critical events as handled by 4G LTE.

We describe the design rationale, its working and challenges as follow.

A. Decoupling Event Based Logic From NFs

Our design is based on following design insights that we make after studying LTE protocol functioning as defined in 3GPP standard documents [20], [54], [55], [56], and their implementation in operational LTE network.

Design insight 1 (NFs are device's event facilitators): LTE network is designed to facilitate both control-plane and data-plane traffic. A closer look into EPC system design architecture and its implementation reveal that much more efforts are spent for addressing control-plane traffic compared to data-plane. For data-plane traffic, i.e., voice and data, fewer NFs (i.e., SGW, PGW and Charging function) are involved. Whereas, control-plane traffic consists of a wide range of different events, executed by greater number of NFs (e.g., MME, SGW, PGW, Charging function, HSS, IMS servers, and DNS servers, etc.) Control-plane events can be categorized into device Attach/Detach, Handover, Location Update, Paging, Service Request/Release, Session Activation/Modification, Security Mode events (i.e., Authentication and Ciphering operations), and many more others. These events, which are handled at LTE control-plane, are also critical to execution of data-plane traffic. The delay or failure of any of the above mentioned event either degrades the user experience, terminates data-plane traffic (i.e., voice and data), or even can cause the device to re-attach.

In short, we view NFs as network events facilitator.

Design insight 2 (Network events execute in blocking-mode): Because these events are critical to device operations, therefore, 3GPP standard requires them to be executed in blocking-mode. In blocking-mode operation, when one event is being executed, no other event can execute. To take an example, if during the *handover* operation, MME detects that the SGW or/and the MME needs be relocated, the MME rejects any such request received since handover operation has started and all other control-plane requests are temporarily rejected due to handover operation in progress [20].

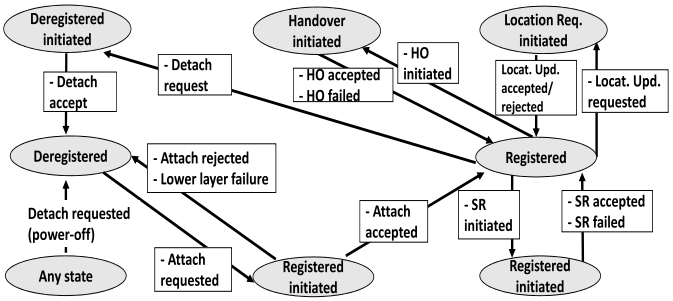

Fig. 9. State transition diagram shows that only one event at a time can be executed.

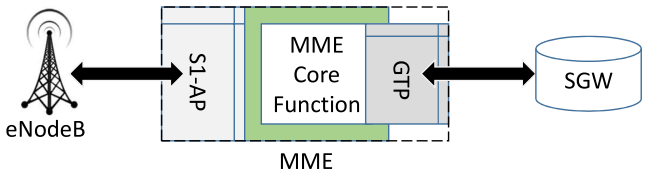

Fig. 10. MME NF can support mutual exclusive operations.

Similarly, the device cannot report multiple events at the same time. In case the device has to report multiple events (e.g., *Handover, Service Request* (*SR*), *or Location Update*), then these events should be executed one by one. Figure 9 shows state transition diagram of different network events. Once a particular network event is initiated, no other event can be originated.

Design insight 3 (Network operations can be mutual exclusive): We find that some network operations are mutual exclusive or disjoint. This is possible because one network function is designed by combining multiple protocols/module. As shown in Figure 10, MME is made of S1AP module, MME core function module, and GTP module. Therefore, MME interaction with eNodeB via S1AP moduke is mutual exclusive to its GTP module's communication to SGW. Therefore, communication between MME and eNodeB and between MME and SGW can be carried out in parallel.

Design insight 4 (Exploiting data center redundancy): Current DC networks typically provide 1:1 redundancy to allow traffic to flow along an alternate route when a device or link becomes unavailable [57]. However, these redundant servers do not take charge until primary server fails, and remain underutilized when primary server is functioning. Therefore, we exploit the underutilized capacity of these redundant servers in our advantage and use them to host event specific logic to facilitate on-demand event execution.

B. Decomposing Network Function as Fat-Proxy

We leverage above observations and decompose event specific execution modules from EPC NFs. The decomposed modules are placed into their respective event category (i.e., handover, service request, and paging), where each module

contains the implementation of executing specific event. Each such module is implemented as a separate functional module, named as Fat-proxy. This gives us three Fat-proxies for three critical events (i.e., handover, service request, and paging) which are hosted over DC network's redundant servers. Now whenever a critical event comes, MME delegates that event's execution to appropriate Fat-proxy. The Fat-proxy executes the event by taking user session information from MME and coordinate with device for follow-up signalling message flow. At the end of execution, it informs MME about the event outcome and updated user session information.

To understand the procedure, we take *handover* event execution example (as explained in Figures 5.5.1.2.2-1: S1-based handover in [58]). A *handover* event occurs when a mobile user travels from one area of coverage or eNodeB to another eNodeB. When an *handover* event triggers, it is possible that user's voice and data services are running, that needs to be transferred to the new eNodeB as well. Therefore user context transfer is not merely towards a new radio base station, but also user's session information and DL/UL traffic migrates from serving NFs to target NFs of EPC. In *handover* event execution, serving NFs (i.e., current MME, SGW and PGW) communicate with target NFs (i.e., new MME, SGW and PGW). This *handover* procedure requires device session transfer, bearer modification, transfer of UL and DL data from source NFs to target NFs, and device location update operation. In short, in one *handover* event execution, a number of control-plane and data-plane messages are exchanged between a number of EPC NFs. Therefore, it requires a considerable effort to implement Fat-proxy that can replace EPC during event execution. Following we explain the steps to implement Fat-proxy, that requires execution logic identification, decomposition, addressing dependencies, and finally coupling different modules into running Fat-proxy.

Identifying event execution logic: We identify event execution logic by looking 3GPP specification documents [20], [55] and LTE EPC source code [59]. In our *handover* event example, the source eNodeB initiates a handover (on behalf of device) by sending *Handover Required* message to source MME. In this procedure, MME and/or the Serving GW may be relocated. Then *Packet forwarding* takes place via the source and target SGWs. Meanwhile, other necessary signalling messages (e.g., dedicated bearer modification; location reporting control; NAS message transfer; etc.) are executed and exchanged between NFs. To trace complete execution path and NF modules for handover, we embed debugging message inside OpenEPC [59] and identify *handover* event execution logic across a number of EPC NFs. We repeat the same procedure for *paging* and *service request* events to identify their execution logic.

Event-based decomposition: Once we identify event execution logic and the message flow for an event, we *decompose* this event from rest of EPC functionality. In our *handover* event example context, we find that there are two types of modules that kicks in on *handover* event, one that are local to *handover* event execution, and the others which have dependency on different other events (e.g., device *location update* procedure, etc.). Therefore, we migrate *handover* specific

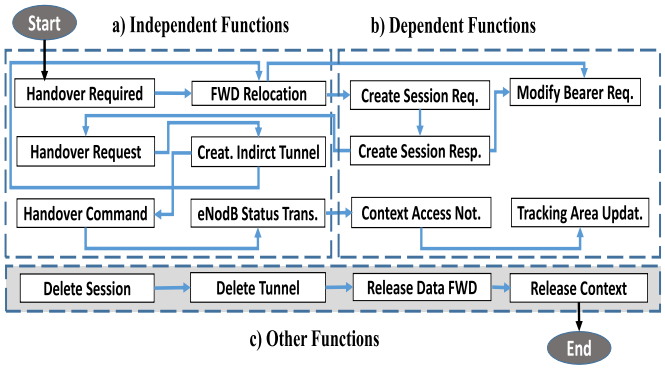

Fig. 11. Functional dependency and non-dependency graph for handover event.

modules to Fat-proxy function, but provides the replica of those modules on which other events have dependency.

Resolving dependencies: One of the challenging task is identifying all those EPC functional modules on which more than one event has dependency. We find dependency among different modules through functional dependency graph, that we generate by implanting debug messages in our source code. Figure 11 shows functional dependency graph of handover event, where handover request, handover command, forward relocation request, and indirect tunneling does not have any dependency, but *bearer modification, session request, and location update* procedures have dependency on different other events; whereas other functions in Figure 11 are not part of *handover event* execution but might trigger when abnormal event comes (such as device detach request, etc.)

Developing Fat-Proxy: We develop Fat-proxy in which we couple all event specific modules from different EPC NFs. To better understand, we explain it through our example of *handover* event – for which we develop Handover Management (HoM) Fat-proxy, as shown in Figure 12. HoM encompasses all those modules which are required to handle message exchange between eNodeB and EPC, between UE and EPC, and between different NFs of EPC. HoM is divided into two phases, i.e., active and passive phases.

Passive phase: In passive phase, we set-up HoM server VM. We pull decomposed *handover* event implementation from MME, SGW and PGW and fused it into HoM. Now HoM becomes execution engine of *handover* event. The HoM server is then connected to MME which delegates handover related execution to it.

Active phase: Because HoM functionality depends upon real-time user information, therefore, during active phase, we ensure all such user information is fed into HoM as soon as *handover* event kicks-in. Thereafter, HoM mimics a full-fledge EPC to outside world (i.e., to serving and target eNodeBs). The handover decisions are independently made by HoM Fat-proxy and final result (i.e., handover successful/failure) along with updated user session information is reported to respective EPC entities.

We should clarify that handling real-time bear-modification is challenging. This is mainly because during *handover* event processing, device bearers are to be modified at actual SGWs

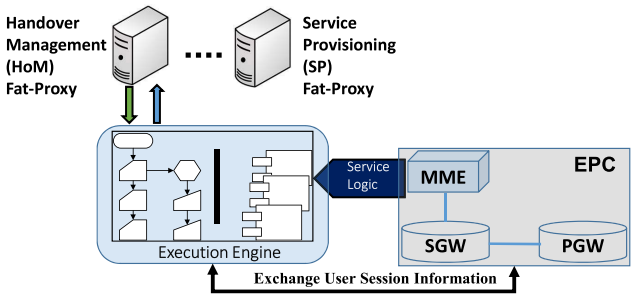

Fig. 12. Decomposing event based service logic from legacy network functions to Fat-Proxy.

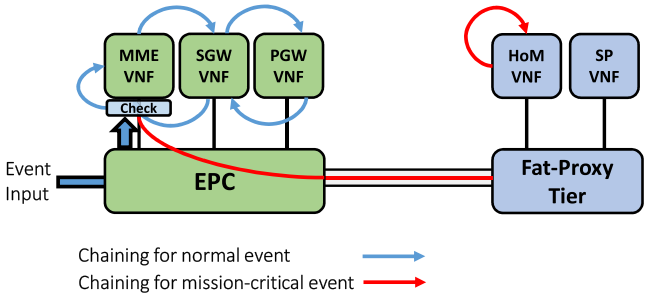

Fig. 13. Service chaining of NFV helps to skip certain NFs for mission critical events.

and PGW of EPC. We provide a work-around where HoM acknowledges eNodeBs that device bearers have been modified by executing the next *handover* procedural steps; but in the background HoM is still negotiating bearer modification step with actual SGWs and PGW. It is possible that such bearer modification step could fail (e.g., one of SGW or PGW fails), but in that case HoM will send *handover* failure message to eNodeBs. We believe handover failure message is common irrespective at which step failure occurs, so artificially proceeding to next execution step, while previous step might fail, does not generate any abnormal result.

C. Service Chaining

In legacy LTE network, when an event arrives at MME, other EPC NFs coordinate back and forth in a fixed loop. In our design, we use service chaining to make distinction with mission critical events than others. We develop a software module, i.e., service agent (shown as "check" in Figure 13, which sits in between S1AP and MME function and encounters all the network events. This service agent is tightly coupled with MME and has access to all device states that MME holds. If the device originated event arriving at service agent is categorized as mission-critical event then this network event along with device states are forwarded to Fat-proxy, whereas non-critical events are directly passed to MME for processing.

Figure 13 shows an example of service chaining where handover event skips execution at MME, SGW and PGW of EPC. As depicted in Figure 13, when the handover request event arrives at service agent of source MME, it is forwarded to HoM Fat-proxy. After that HoM communicates with the radio network and device to facilitate the handover operation. Because the handover service logic of different NFs (such as source MME, target MME, source SGW, target SGW, source PGW, and target PGW) is local to HoM Fat-proxy function, therefore, the handover procedure is not only faster but also excludes any network failure cases between different NFs.

D. Space Uncoupling

A network event triggers coordination between a series of NFs. Such coordination takes place between different NFs within EPC, between EPC and the radio network. The network events produce significant messages flow between MME and eNodeB compared to MME with any other EPC NF. MME request eNodeB to establish secure radio connection with UE, instruct eNodeB to establish device context, initiate the connection between SGW for user UL/DL traffic, and many more. For example, one *paging* event generates 5 messages between MME and eNodeB compared to only 1 message between MME and the serving GW.

Such signalling message exchange between MME and eNodeB are carried out by S1AP module. The S1AP module is a stand-alone NF with a number of unique functions, such as radio bearer management function, mobility function, NAS Signalling transport function, location reporting function, network flow control and congestion control function, and many more. Although, S1AP sits between MME and eNodeB where its role has been uniquely defined by 3GPP, but legacy EPC infrastructure tightly couples it with MME. Such an implementation has its justification that not only renders faster communication between MME and S1AP but also helps S1AP to exploit MME hardware level fault tolerance [36], [60], [61]. But in virtualized EPC, we are addressing different goals, where we are interested in not only avoiding intra data-center latencies, but also wants to enable faster response between EPC and eNodeB through NFd decoupling and employing service chaining.

In order to satisfy above goals, we decompose S1AP module from MME and install it at the edge of the cloud. Because S1AP function also carries those signalling messages which need to be forwarded to MME core module, therefore, we extend a direct interface between S1AP and MME function. As shown in Figure 10, S1AP module is decomposed from MME core function. Such design choice results faster communication between eNodeB and EPC even during data-center network congestion.

1) Concurrent Execution of Signalling Messages: In order to facilitate one network event, different EPC NFs communicate with each other. LTE standard mandates such signalling exchange between these NFs are carried out serially [62], [63]. That is, subsequent message in the chain of signalling messages will not be sent if the previous message has been successfully executed. In case of any failure, the whole procedure is terminated and an error is sent back to the UE. On receiving an error message, the device makes an other attempt.

For example, successful *handover* event requires 32 messages to be executed between different EPC NFs. The delay

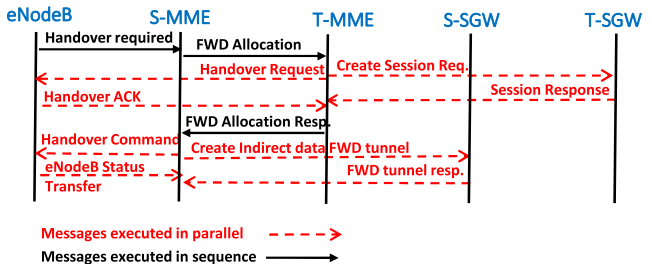

Fig. 14. Concurrent Execution of Messages.

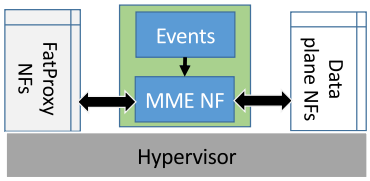

Fig. 15. Implementation schema.

in executing any message will add to handover delay, and any message execution failure will abort the whole handover procedure. Figure 14 shows subset of messages exchange between EPC NFs during *handover* event. Serving MME receives *Handover Required* message from eNodeB and triggers *Forward Allocation* message to target MME. On receiving the device session information, target MME creates device session with target SGW. On successful session response, target MME sends *Handover Acknowledgement* to eNodeB. It should be noted that the direction of "Create Session Request" and "Handover Request" messages are opposite, where former is sent to serving SGW within EPC, and the later is sent out of EPC to the eNodeB. Moreover, the contents of both messages are mutually exclusive. Therefore, we accelerate handover event by executing such mutual exclusive messages in parallel (as shown in dotted line of Figure 14).

We found that in most network events, there exists 40% to 60% messages that can be executed concurrently, that significantly improves the network performance.

Skipping some messages: In standardized event execution, temporary device states are first created, and later deleted on completion of an event. For example, in *handover* event execution (Figure 11c – mixed functions), device session information and context are modified – whereas old device states and context are held temporarily. Also, temporary indirect tunnel is established to relay packets to target NFs. When *handover* event is successfully concluded, these temporary states are deleted through special messages exchange between NFs. In our design, because states in Fat-proxy are transient in nature anyway (where MME delegates the event execution to Fat-proxy and after event completion all device information is cleared from Fat-proxy), we do not require to execute "delete session" related steps. In our *handover* event example, we reduce 25% signalling messages.

Transaction rollback on failure message: We understand that concurrent message execution failure may provide inconsistent view of network states. For example, on receiving the "Handover Request" message from target MME, the eNodeB believes that the target MME has successfully established the device connection with target serving SGW. But such eNodeB's view of network state may become false, in case target MME is failed to establish device session with target SGW. Therefore, in order to handle such network state inconsistencies, we propose transaction rollback by sending network failure message. As stated earlier, message execution failure at any step will terminate whole handover process, therefore, by sending a failure message (even at later step) to eNodeB will address any inconsistency previously caused by concurrent message execution.

VI. System Implementation

Our system implementation consists of open source LTE implementation and virtualization of LTE EPC NFs.

Open source LTE deployment: Our test-bed consists of a LTE eNodeB (nanoLTE Access Point [64]), OpenEPC software EPC platform [59], and Samsung S6 smartphones. The eNodeB is a 3GPP Release 10 compliant LTE small cell on 700 MHz band. Considerable effort, involving code modifications to OpenEPC components, was spent to integrate eNodeB (closed-source) with EPC to ensure interoperability with commercially available LTE clients (i.e., Samsung S6 smartphones). Our EPC network consists of MME, HSS, PCRF for control plane and SGW and PGW for data plane functions. In addition, the Internet gateway provides connectivity to the Internet. Samsung S6 smartphones use USIM cards programmed with the appropriate identification name and secret code to connect with eNodeB. Since eNodeB and the device communicate on T-Mobile's licensed band, we use custom built frequency converters. These converters convert the frequency in both downlink and uplink from 700 MHz to 2.6 GHz, where we have an experimental license to conduct over the air experiments.

User emulation: For evaluation of space uncoupling, where S1AP-MME simultaneously communicates with S1AP-eNodeB and SGW, we require changes at eNodeB-S1AP side. Because our eNodeB is closed-source, therefore, we use device emulation provided by OpenEPCP. OpenEPC provides *client-Alice* module that emulate user device and eNodeB and interacts with EPC NFs. The client-Alice module has basic S1-AP functionality, enough to show performance improvement when space-coupling is used.

Virtualizing LTE EPC: After LTE testbed deployment, we virtualize EPC NFs. EPC virtualization includes deployment of decomposed EPC NFs over VMs, and exposing them to real LTE traffic load. High level implementation schema is described in Figure 15.

NF decomposition and placement: We virtualize EPC NFs over vMware vSphere [65], which is a server virtualization platform with consistent management. We first decomposed OpenEPC into a number of LTE NFs (i.e., MME, SGW, PGW, HSS, and PCRF). Then we treat these NFs as VNFs running as separate VMs. We implement virtualized interfaces in order to relay packets to and from these VNFs. We gather results by changing testbed configuration in different settings: (a) each

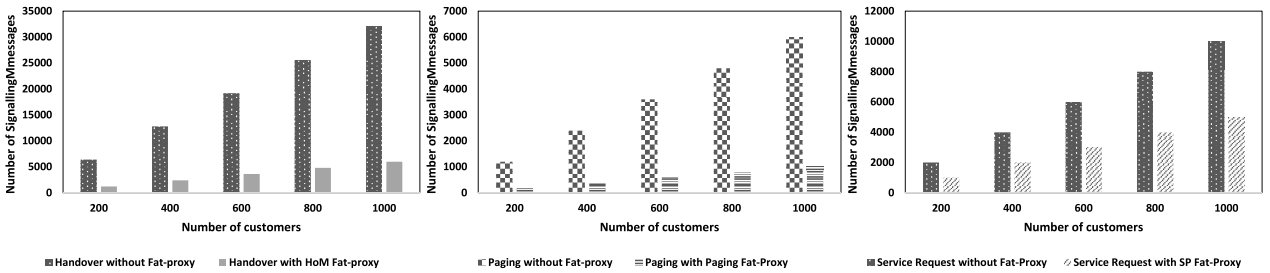

Fig. 16. Signalling load EPC is exposed during peak hours with and without Fat-proxy.

NF is decomposed and then mapped to a VNF, (b) placing these VNF over different servers, which are then connected through network tunnel, and (c) installing the Fa-proxy VNF that combines event-specific NF components.

Considering real data-center network loads: Because lab testbed environment does not (a) add round trip time to DC network (b) consider dynamic loads at servers (c) take DC network congestion into account, we consider DC network performance metrics while compiling our results. We parsed system logs provided by HP Helion cloud infrastructure [66] and gather inter-data-center network latency metrics. We measure round trip time from query entering and exiting the DC network. Our results are consistent with previous studies on data-center network performance [67], [68].

Note: in order to provide fair comparison, we neglected CPU processing time, delays because of congestion or queueing at switches. Also note that we conduct experiments over nanoLTE Access Point to show a proof of concept system. The detailed experiments as discussed in this paper are conducted under emulation mode where we can generate high traffic load scenarios.

VII. Evaluation and Results

We evaluate our design from three major aspects: (1) controlling signalling storm, (2) timely execution of mission critical events, and (3) performance impact. We ran our tests on a local network of servers with 10-core Intel Xeon E5 - 2650 v3 processors at 2.3Ghz, 25MB cache size, and 16GB memory. We build our prototype and tested it using real smartphones (Samsung S6 smartphones) and device emulation mode of OpenEPC. To consider real operator network scenario, we use a network trace as our input packet stream; results are representative of tests we ran on other traces.

Signalling load at EPC: As mentioned earlier in the paper, during busy hours, operational LTE core is exposed to signalling storm. First, we show that our design reduces signalling storm by diverting highly occurring network events to Fat-proxy. Although, Fat-proxy is part of LTE core, all execution remains local to Fat-proxy module. In this way, LTE core NFs (such as MME, SGW ad PGW) are not exposed to high signalling messages exchange and remain functional at all time. Figure 16 shows that for *handover*, *paging* and *service request* events, LTE core is exposed 5X, 6X and 2X respectively less signalling traffic compared to the case when Fat-proxy is not used. We see *paging* event benefits most from our design

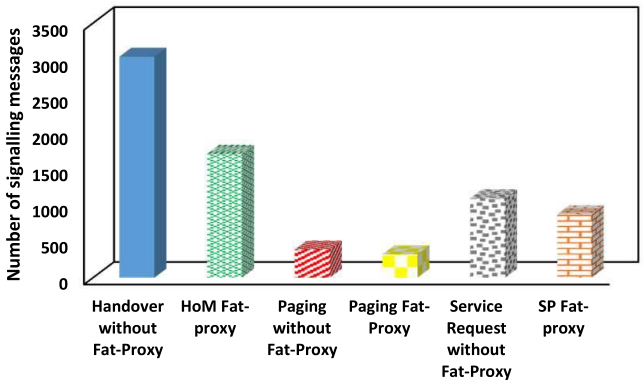

Fig. 17. Total number of signalling messages per subscriber during busy hour with and without Fat-Proxy.

which is due to the fact that paging Fat-proxy communicates all the paging signalling with eNodeB when MME delegates paging execution to Fat-proxy. Whereas *service request* event requires bearer modifications at actual SGW and PGW which relatively increases EPC signalling load even in case of SP Fat-proxy.

Total number of signalling messages: We show that less number of signalling messages are generated by each event with Fat-proxy compared to the case when Fat-proxy is not used. This is mostly because of the space uncoupling concept (Section V-D) where we can not only execute some messages in parallel, but can also skip few messages to be even executed. Note that, because two messages are executed in parallel, therefore, we count them as one, but in actual implementation exactly two messages are generated. The rationale of treating pair of parallel message as one is that these two messages are traveling in opposite direction, i.e., one out of EPC and the other towards EPC NF. Therefore, both of these messages are independent to each other execution. Figure 17 shows *handover* event produces around 40% less signalling messages, when *handover* event is handled by Fat-proxy. Whereas, *paging* event can only skip one message of *Uplink-Nas-Transport*. This NAS message carries the information about the service that device wants to receive from LTE network.

Event execution time: Figure 18 shows CDF of event execution time for *handover*, *paging* and *service request* events with and without Fat-proxy implementation. We see that on average with HoM Fat-proxy *handover* event latency decreases by the factor of 6X. This improvement is observed because HoM

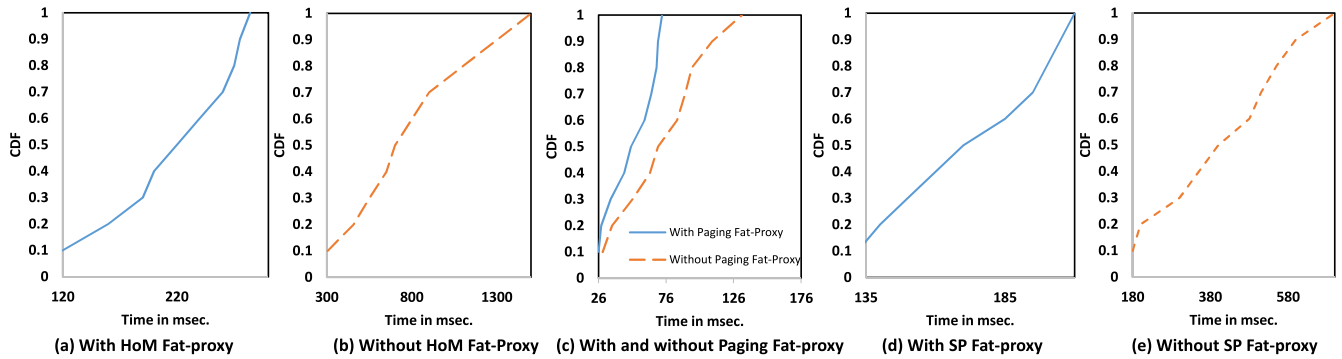

Fig. 18. Event execution time during peak hours with and without fat-proxy concept.

executes 6 signalling messages in parallel and skips total of 8 messages. Moreover, HoM event logic local to one VM does not suffer any network delays. We note that even with HoM Fat-proxy, handover latency is higher than 100 ms. This is because in our experiment we handle worst case handover scenario in which both MME and SGW are relocated. Although event execution time does not meet QoS time-bounds shown in Table I, it does not affect user QoS experience where users' data packets are tunneled from old serving SGW to target SGW and then delivered to user.

Paging event execution time with and without Fat-proxy is not significantly high. We observe in *paging* event, all of the signalling messages are exchanged between S1AP of MME and eNodeB which diminishes intra-EPC NFs delay. The improvement we see in Figure 18c is mainly achieved by pushing S1AP to the edge of cloud and executing one pair of message in parallel.

SP Fat-proxy, on average can reduce only upto 50% *service request* event execution time mainly because at the end of *service request* event execution, SP Fat-proxy needs to update device bearers with EPC.

VIII. Discussion

We discuss how Fat-proxy can achieves service-level resilience during mission critical event execution, and does not introduce any performance overhead.

On voice service resilience: We use most common LTE service (VoLTE) to test its resilience during *handover* mission critical event. We deploy openIMS [69] and make calls request to IMS server during *handover* event. The standard LTE voice call feature requires, the IMS peer must respond to call establishing control-plane messages within a time, defined by a timer T1 [70], [71]. T1 timer is calculated based on RTT value between two IMS servers. In case the client device does not receive an answer from IMS server, the timer T1 times-out and client device makes a retry attempt. The call establishment request aborts when client does not receive an answer to its control message within five consecutive retries. Our goal was to observe number of call drops with maximum possible timer T1 value observed in our handover event experiment. As high the T1 timer value goes, the call-drop probability becomes low – but at the cost of service provisioning delay [72]. Therefore,

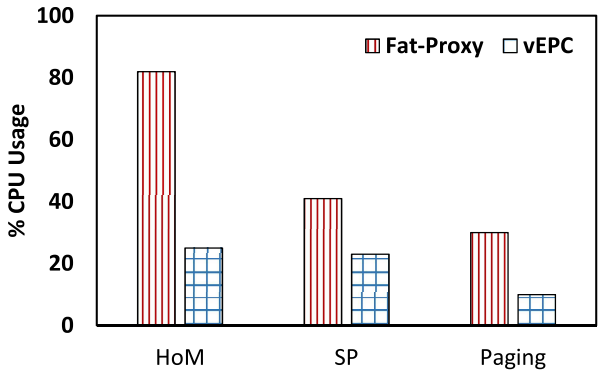

Fig. 19. CPU usage for end-to-end event execution where vEPC intercepts all events, delegates event execution to Fat-Proxy, and finally update results. The Fat-Proxy executes the event.

we consider the conservative timer T1 value of 30 ms (including the radio delay). We observe that for 20%-30% of all *handover* events request, there is a call drop. But with handover proxy implementation, the call drop rate drops to zero. This is because HoM Fat-proxy event executes before the call timeout value of 930 ms. In short, Fat-proxy achieve service resilience during mission critical events, which otherwise not possible.

Note that, call drop can only be observed during *handover* event. Because for *paging* and *service request* events, the device has not obtained the network resources and as a result call request cannot be made. Moreover, forwarding tunnel in *handover* event which is used to forward only data-plane messages from source NF to destination NF does not carry voice control-plane messages. That is control-plane messages are blocked until *handover* event is completed. Therefore, *handover* event delay has direct impact on calls failure.

On overhead: We show that our design does not introduce any performance overhead, rather it increases the vEPC capacity to accept more number of requests. Figure 19 and Figure 20 show CPU and memory usage with Fat-proxy. Handover event which is computation intensive takes most of the CPU resources compared to paging and service request events. As shown in Figure 19, Fat proxy performs most of the event execution and let vEPC to accept more user requests. Our design enhance vEPC capacity upto 5 times, in which 5X

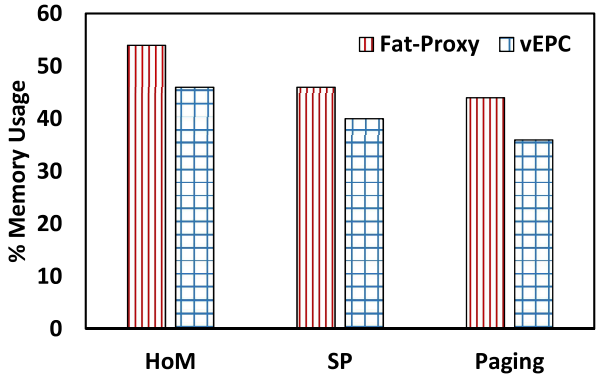

Fig. 20. Memory usage for end-to-end event execution where vEPC keeps all device states in memory which are updated when Fat-Proxy returns the updated states. Fat-Proxy modifies device states during the event execution.

more customers can be registered with single vEPC. Therefore, the network operator does not need to configure more vEPC (i.e., MME, SGW, PGW, and HSS), rather it need to spin readily available Fat-proxies (i.e., 5 fat-proxies in our case). We observe the memory usage in all cases are almost same (as shown in Figure 20). This is mainly because both vEPC and HoM need to keep all in-service users information in memory. Although vEPC delegates the execution to fat-proxy, it still keeps copy of users session in memory to quickly recover in case of some failure. Furthermore, more number of events occurrence does not significantly increase the memory usage due to the fact that event execution related functionality has already been loaded into memory when fat-proxy start serving first customer.

We should also highlight that the delay between EPC NF and FatProxy for a given user is negligible. Take an example of handover event. When the handover critical event occurs, the source eNodeB sends a signalling message “Handover Required" to source MME. On receiving this message, the source MME forwards the request to Handover FatProxy along with the user’s session states. Thereafter, the Handover FatProxy takes control of the handover event and executes rest of handover procedure signaling messages.

IX. Related Work

We elaborate on related efforts, which are categorized as following.

Industry and academic efforts: The ETSI has provided several documents discussing guidelines and requirements for LTE-NFV. The open source NFV platform (OPNFV) [41] is designed to accelerate deployment efforts of LTE-NFV. There are several white papers provided by technology giants [2], [73], [74], [75], but none of them has demonstrated any (1) system design of LTE-NFV that solves LTE specific issues in virtualized environment (2) prototype that clearly shows the merit of LTE-NFV over legacy LTE EPC design. Our work stimulates discussion on system design that clearly shows the merit over naive NF decomposition approach discussed in industrial white papers.

End-to-end NF management: Closest to our work is [76], which discusses the NF placement problem in LTE core. While both, our work and [76], address NF decomposition issues but our work significantly differ from their design goals and problem statement. The [76] aims to use SDN for control-plane traffic whereas enhances data-plane performance through NFV decomposition. In contrast, we try to solve issues mainly stem from control-plane traffic explosion where time-critical events miss their tight execution deadline and render affect on data-plane traffic. For example, the delay in lengthy handover event (that consists of upto 32 signalling message exchanges between various NFs) cause call or data-packets drop or even causes the deregistration of the device from the network.

NF Fat-proxy: The concept of Fat-proxies or thick clients is very useful in distributed community, where server delegates some processing logic to the client [77], [78]. We motivate our design from similar goal but under different setting, where we want to make proxy a powerful component. We delegate full event execution control to Fat-proxy by providing complete user session information. As a result, we only seek updated user session information and the final outcome of delegated event from these Fat-proxies. While the key Fat-proxy based idea in LTE NFV was first discussed in our previous work [79], we have accessed the key ideas from different perspectives. In this work, we provide a complete solution by incorporating all systems level details that enable high availability in LTE NFV.

LTE NFV: There have been recent efforts on LTE NFV. Reference [80] refactors LTE NFs to reduce latencies for latency sensitive applications like IMS. However, this work concerns about high availability of critical LTE procedures. Reference [81] discloses that existing LTE design causes bottlenecks to service availability. This work strengthen the arguments in our paper that LTE architecture is not designed for virtualization. Our work solves the problem of service availability which is not addressed in [81]. Reference [82] focuses on the performance of LTE core in NFV. In contrast, we aim to provide high service availability under heavy traffic load scenarios that also improves the performance. Reference [83] proposes four different LTE architecture alternatives to enable split LTE control and data planes, that is adopting SDN-style architecture. Reference [84] proposes a method to determine the binding of eNodeBs to the traffic aggregation nodes and their amount in virtualized networks. Different to both these works, our work proposes a separate execution engine that we call FatProxy to make LTE NFV a reality.

X. Conclusion

In this paper, we make an effort to highlight major issues in virtualizing LTE core. We address those control-plane messages which make-up 50% of all EPC signalling and have direct impact on data-plane traffic. Our proposed Fat-proxy takes signal intensive events away from the core network and enjoys parallel execution of messages through space uncoupling, which otherwise not possible. Results show that our

design can not only maintain event execution time-bounds, but also reduces signalling traffic by skipping unnecessary signalling messages.

REFERENCES

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

[2] *Carrier NFV Project—One Year Milestone*. [Online]. Available: http://www.layer123.com/download&doc=ETSI-1013-Lopez-NFV_One_Year_Milestone

[3] *GS NFV-INF 001: Network Functions Virtualisation (NFV); Infrastructure Overview*, ETSI, Sophia Antipolis, France, 2015.

[4] *LTE NFV Deployment for IoT Services*. Accessed: Aug. 28, 2020. [Online]. Available: http://www.businesscloudnews.com/2015/02/05/samsung-and-sk-to-deploy-nfv-in-live-iot-lte-network/

[5] *NFV Use Cases Emerge as IoT Evolves*. Accessed: Aug. 28, 2020. [Online]. Available: http://internetofthingsagenda.techtarget.com/tip/NFV-use-cases-emerge-as-IoT-evolves/

[6] *LTE Signaling on the Rise With VoLTE, LTE Growth*. Accessed: Aug. 28, 2020. [Online]. Available: http://www.rcrwireless.com/20151118/network-function-virtualization-nfv/ltena-2015-lte-signaling-tag6/

[7] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2011, pp. 63–74.

[8] T. Benson, A. Akella, and D. Maltz, "Network traffic characteristics of data centers in the wilds," in *Proc. ACM IMC*, 2010, pp. 267–280.

[9] M. Calder, R. Miao, K. Zarifis, E. Katz-Bassett, M. Yu, and J. Padhye, "Don't drop, detour!" in *ACM SIGCOMM Comput. Commun. Rev.*, 2013, pp. 503–504.

[10] M. Alizadeh, S. Yang, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "Deconstructing datacenter packet transport," in *Proc. ACM Workshop Hot Topics Netw.*, 2012, pp. 133–138.

[11] Y. Xia, T. Wang, Z. Su, and M. Hamdi, "Preventing passive TCP timeouts in data center networks with packet drop notification," in *Proc. CloudNet*, Oct. 2014, pp. 173–178.

[12] D. Zats, A. P. Iyer, R. H. Katz, I. Stoica, and A. Vahdat, "FastLane: An agile congestion signaling mechanism for improving datacenter performance," EECS Dept., Univ. California at Berkeley, Berkeley, CA, USA, Rep. UCB/EECS-2013-113, 2013.

[13] N. Farrington, "Multipath TCP under massive packet reordering," EECS Dept., Univ. California at San Diego, San Diego, CA, USA, Rep. TR 01-ATL-062917, 2001.

[14] *Of Mice and Elephants*. Accessed: Jan. 11, 2020. [Online]. Available: http://networkheresy.com/2013/11/01/of-mice-and-elephants/

[15] *Policy and Charging Control Architecture*, 3GPP Standard TS 23.203, 2013.

[16] *General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)*, 3GPP Standard TS 29.281, 2013.

[17] J. Networks, "GPRS tunneling protocol (GTP) for serving gateway," Juiper, Sunnyvale, CA, USA, Rep. 11.2, 2014.

[18] A. Lucent, "LTE subscriber service restoration," ALU, Romano d'Ezzelino, Italy, Rep. NP2014045334EN, 2014.

[19] "Signaling is growing 50% faster than data traffic," Nokia Siemens Networks, Espoo, Finland, Rep. C401-00780-WP-201212-1-EN, 2012. Accessed: Aug. 9, 2021.

[20] *Non-Access-Stratum (NAS) Protocol for Evolved Packet System (EPS); Stage 3*, 3GPP Standard TS 24.301, Jun. 2013.

[21] A. L. Chiu *et al.*, "EdgePlex: decomposing the provider edge for flexibilty and reliability," in *Proc. 1st ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, 2015, pp. 1–6.

[22] W. Hahn and B. Gajic, "GW elasticity in data centers: Options to adapt to changing traffic profiles in control and user plane," in *Proc. Intell. Next Gener. Netw. (ICIN)*, 2015, pp. 16–22.

[23] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet, "Network service chaining with efficient network function mapping based on service decompositions," in *Proc. 1st IEEE Conf. Netw. Softw. (NetSoft)*, 2015, pp. 1–5.

[24] R. Mahindra, H. Viswanathan, K. Sundaresan, M. Y. Arslan, and S. Rangarajan, "A practical traffic management system for integrated LTE-WiFi networks," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw. (Mobicom)*, 2014, pp. 189–200.

[25] L. Osmani *et al.*, "Building blocks for an elastic mobile core," in *Proc. CoNEXT Student Workshop*, 2014, pp. 43–45.

[26] "Charting the signalling storm, v3.1," Stoke, Bedford, MA, USA, Rep. 3.1, 2012.

[27] *SK Telecom Deploys Samsung's Virtual IMS*. Accessed: Mar. 1, 2020. [Online]. Available: https://www.sdxcentral.com/articles/news/sk-telecom-deploys-samsungs-virtual-ims-epc/2015/09/

[28] *Spark New Zealand Deployed Metaswitch's vIMS Platform*. [Online]. Available: http://www.telecompaper.com/news/spark-new-zealand-deployed-metaswitchs-vims-platform–1088686/

[29] *DoCoMo to Offer NFV-Based LTE in 2016*. [Online]. Available: http://www.lightreading.com/carrier-sdn/nfv-(network-functions-virtualization)/docomo-to-offer-nfv-based-lte-in-2016-/d/d-id/709223/

[30] *Telefonica: Evolving Towards a Fully Virtualized Network*. [Online]. Available: https://www.telefonica.com/documents/26188/318131/Stand_Virtualization.pdf/c72ce5f9-7a57-4a85-8848-a3916781f213/

[31] *AT&T Shows How Serious It Is About SDN, NFV and Open Source*. [Online]. Available: http://www.fiercewireless.com/tech/story/att-shows-how-serious-it-about-sdn-nfv-and-open-source/2015-08-20/

[32] *Huawei Shows vEPC, Virtual IMS With China Mobile*. Accessed: Mar. 1, 2020. [Online]. Available: http://www.telecompaper.com/news/huawei-shows-vepc-virtual-ims-with-china-mobile–998836/

[33] *Network Functions Virtualisation (NFV); Virtual Network Functions Architecture*, document GS-NFV-SWA-001, ETSI, Sophia Antipolis, France, 2015.

[34] *Survey: NFV Expected to Deliver Opex, Capex Benefits*. Accessed: Mar. 1, 2020. [Online]. Available: http://www.fiercewireless.com/tech/story/survey-nfv-expected-deliver-opex-capex-benefits/2013-12-07/

[35] *NFV, Initiative to Reduce Network Expenses (CAPEX & OPEX Both)*. Accessed: Mar. 1, 2020. [Online]. Available: https://www.linkedin.com/pulse/nfv-initiative-reduce-expenses-capex-opex-both-apoorv-gupta/

[36] *Ericsson Blade System (EBS) for EPC*. [Online]. Available: http://archive.ericsson.net/service/internet/picov/get?DocNo=266/03819-FAP130506&Lang=AE&HighestFree=Y

[37] *Alcatel-Lucent 9471 Wireless Mobility Manager*. Accessed: Mar. 1, 2020. [Online]. Available: https://www.alcatel-lucent.com/products/9471-wireless-mobility-manager

[38] *MME Trigger OVERLOAD START Towards ENodeB*. Accessed: Mar. 1, 2020. [Online]. Available: http://tech.queryhome.com/103073/different-situations-trigger-overload-start-towards-enodeb

[39] *Ensuring High-Availability and Resiliency for NFV*. Accessed: Mar. 1, 2020. [Online]. Available: https://www.viavisolutions.com/pt-br/literature/ensuring-high-availability-and-resiliency-nfv-white-papers-books-en.pdf

[40] *OpenStack Open Source Cloud Computing Software*. Accessed: Mar. 1, 2020. [Online]. Available: https://www.openstack.org/software/

[41] *Open Platform for NFV (OPNFV)*. Accessed: Mar. 1, 2020. [Online]. Available: https://www.opnfv.org/

[42] *System Reliability and High Availability in CloudStack*. Accessed: Mar. 1, 2020. [Online]. Available: http://docs.cloudstack.apache.org/projects/cloudstack-administration/en/4.6/reliability.html/

[43] *OpenNebula High Availability 4.2*. Accessed: Mar. 1, 2020. [Online]. Available: http://docs.opennebula.org/4.12/advanced_administration/high_availability/oneha.html

[44] R. Mittal *et al.*, "TIMELY: RTT-based congestion control for the datacenter," in *Proc. ACM SIGCOMM*, Aug. 2015, pp. 537–550.

[45] L. Henrique, G. Ferraz, O. Carlos, and M. B. Duarte, "Analysis of multipathing techniques in data center networks," Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, Rep. TS2-01, 2013.

[46] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based load balancing for fast datacenter networks," in *Proc. ACM Sigcomm*, 2015, pp. 465–478.

[47] *Ericsson SGSN-MME*. Accessed: Jan. 11, 2020. [Online]. Available: http://www.ericsson.com/ourportfolio/products/sgsn-mme?nav=productcategory004

[48] "Cisco ASR 5000 mobility management entity," Cisco, San Jose, CA, USA, Rep. OL-22987-01, 2014.

[49] "Configuring GGSN GTP services, v3.1," Cisco, San Jose, CA, USA, Rep. OL-19936-03, 2016.

[50] C. Yu, M. Xu, X. Fu, and E. Dong, "Explicit multipath congestion control for data center networks," in *Proc. ACM CoNEXT*, Dec. 2013, pp. 73–84.

[51] S. M. Rumble, D. Ongaro, R. Stutsman, M. Rosenblum, and J. K. Ousterhout, "It's time for low latency," in *Proc. USENIX Conf. Hot Topics Oper. Syst.*, 2011, pp. 129–140.

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

[53] *High Availability Concepts in OpenStack Platform*. Accessed: Aug. 9, 2020. [Online]. Available: http://docs.openstack.org/ha-guide/

[54] *Radio Interface Protocol Architecture*, 3GPP Standard TS 25.301, 2008.

[55] *Radio Resource Control (RRC)*, 3GPP Standard TS 36.331, 2012.
[56] *General Packet Radio Service (GPRS); Service Description; Stage 2*, 3GPP Standard TS 23.060, Dec. 2006.
[57] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2011, pp. 350–361.
[58] *GPRS Enhancements for E-UTRAN Access*, 3GPP Standard TS 23.401, 2011.
[59] *Open EPC—Open Source LTE Implementation*. Accessed: Sep. 10, 2020. [Online]. Available: http://www.openepc.net/
[60] *Build Massively Scalable Soft Real-Time Systems*. Accessed: Aug. 19, 2020. [Online]. Available: http://www.erlang.org/
[61] *Alcatel-Lucent 5620—Service Aware Manager*. Accessed: Aug. 19, 2020. [Online]. Available: https://infoproducts.alcatel-lucent.com/cgi-bin/dbaccessfilename.cgi/3HE03415AAAA01_V1_Alcatel-Lucent/
[62] *3GPP Specification: TS23.060, TS23.401, TS23.228, TS23.272, TS24.008, TS36.331, TS36.304*. Accessed: Aug. 19, 2020. [Online]. Available: http://www.3gpp.org
[63] H. Holma and A. Toskala, *LTE for UMTS: Evolution to LTE-Advanced*. Hoboken, NJ, USA: Wiley, 2011.
[64] *NanoLTE Access Points*. Accessed: Aug. 28, 2020. [Online]. Available: http://www.ipaccess.com/en/lte/
[65] *NF Virtualization Using VMware's vSphere*. Accessed: Aug. 28, 2020. [Online]. Available: https://www.vmware.com/products/vsphere/features/
[66] *HP Helion: Hybrid Cloud Solutions*. Accessed: Aug. 28, 2020. [Online]. Available: http://www8.hp.com/us/en/cloud/helion-overview.html/
[67] T. Kraska, G. Pang, M. J. Franklin, S. Madden, and A. Fekete, "MDCC: Multi-data center consistency," in *Proc. 8th ACM Eur. Conf. Comput. Syst. (EuroSys)*, 2013, pp. 113–126.
[68] *Understanding I/O: Random vs Sequential*. Accessed: Sep. 10, 2020. [Online]. Available: http://flashdba.com/2013/04/15/understanding-io-random-vs-sequential/
[69] *Open Source IMS Core*. Accessed: Aug. 9, 2020. [Online]. Available: http://www.openimscore.org//
[70] *IP Multimedia Subsystem (IMS); Stage 2*, 3GPP Standard TS 23.228, 2012.
[71] *3GPP IMS Management Object (MO); Stage 3*, 3GPP Standard TS 24.167, 2012.
[72] W. Jiang, K. Koguchi, and H. Schulzrinne, "QoS evaluation of VoIP end-points," in *Proc. ICC*, 2003, pp. 1917–1921.
[73] *Network Functions Virtualization: Challenges and Opportunities for Innovations*. Accessed: Jul. 18, 2020. [Online]. Available: http://web2-clone.research.att.com/export/sites/att_labs/techdocs/TD_101400.pdf
[74] *Network Functions Virtualization and Software Management*. Accessed: Jul. 18, 2020. [Online]. Available: http://www.ericsson.com/res/docs/whitepapers/network-functions-virtualization-and-software-management.pdf
[75] *Wireless Network Virtualization: Ensuring Carrier Grade Availability*. Accessed: Jul. 18, 2020. [Online]. Available: https://www.sdxcentral.com/wp-content/uploads/2014/10/Wind-River_CRAN-white-paper.pdf
[76] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proc. All Things Cellular Oper. Appl. Challenges*, 2014, pp. 33–38.
[77] J. H. Howard *et al.*, "Scale and performance in a distributed file system," *ACM Trans. Comput. Syst.*, vol. 6, no. 1, pp. 51–81, 1988.
[78] P. P. Hung and E.-N. Huh, "An adaptive procedure for task scheduling optimization in mobile cloud computing," *Math. Probl. Eng.*, vol. 2015, May 2015, Art. no. 969027.
[79] 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)*, 2017, pp. 1–10.
[80] M. T. Raza, S. Lu, M. Gerla, and X. Li, "Refactoring network functions modules to reduce latencies and improve fault tolerance in NFV," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2275–2287, Oct. 2018.
[81] A. S. Rajan *et al.*, "Understanding the bottlenecks in virtualizing cellular core network functions," in *Proc. 21st IEEE Int. Workshop Local Metropolitan Area Netw.*, 2015, pp. 1–6.
[82] 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. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 348–361.
[83] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E.-D. Schmidt, "A virtual SDN-enabled LTE EPC architecture: A case study for S-/P-gateways functions," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, 2013, pp. 1–7.
[84] M. Skulysh, S. Sulima, and G. Grynkevych, "Traffic aggregation nodes placement for virtual EPC," in *Proc. IEEE Int. Conf. Inf. Telecommun. Technol. Radio Electron. (UkrMiCo)*, 2019, pp. 1–4.