FERRET: Fall-back to LTE Microservices for Low Latency Data Access

Muhammad Taqi Raza
University of Arizona

Fatima Muhammad Anwar
UMASS Amherst

Dongho Kim
AT&T Research Lab

Kyu-Han Kim
HP Enterprise Research Lab

ABSTRACT

Motivated from Software Defined Networking (SDN), LTE standard body (3GPP) has recently proposed splitting monolithic LTE Network Functions (NFs) into their control-plane and data-plane modules for better performance, flexibility and agility. The data-plane logic is pushed at the edge of the network, while retaining control-plane functionality at the core. However, both network edge and the core modules involve in executing LTE control-plane procedures (e.g. *device registration/deregistration*, and *mobility* etc.) as well as LTE data-plane services (e.g. voice over LTE, and video streaming, etc.). We discover that these decoupled modules, being part of the same LTE network function, interact frequently and cause deadlocks and races. In this paper, we argue that SDN style approach may not work for LTE NFs due to their monolithic design. We reason to retain LTE legacy design by not splitting its NFs. Our idea is to keep all types of LTE control-plane procedures handling at the core; while moving the execution of LTE data-plane services to the edge as microservices. We propose $FERRET$ that is inspired from the success of the Circuit Switch Fall Back (CSFB) procedure, and falls-back to specific LTE microservice for the user requesting a particular LTE service. It first records signaling messages exchange as part of LTE service establishment phase at the core and then replays these messages at dedicated microservice to enable that service handling. $FERRET$ lets microservice to facilitate LTE service execution that provides data forwarding at the edge.

1 Introduction

In Fourth Generation (4G) LTE network, packet processing Network Functions (NFs) are implemented at the backend of cellular network. These NFs being far away from radio base station introduce latencies for both control and user planes (also referred as data-planes in the paper) packets at both uplink and downlink directions. All packets originated from the source first traverse cellular backend NFs before they are forwarded to the destination. Such a cellular design choice was made to optimize backend NFs processing and management where all these NFs are implemented over carrier grade boxes that provide strong coupling between software and hardware implementation [1]. Recently, Network Function Virtualization (NFV) concept has emerged that breaks the software dependency on underlying hardware. Although, NFV idea was envisioned to reduce operators capital and operation expenditures by implementing network packet processing logic on commodity off-the-shelf boxes, it also provides an opportunity to redesign cellular architecture. The idea of redesigning LTE architecture is well received by Fifth Generation Network (5G) project whose aim is to reduce end-to-end latencies and increase throughput. 5G introduces Mobile Edge Computing (MEC) paradigm in cellular space by bringing user-plane traffic processing and forwarding modules closer to mobile edge, whereas keeping the control-plane operational modules at the core. In this direction, LTE standardization body (3GPP), inspiring from software defined networking (SDN) concept, has recently proposed mechanisms to split control and user planes NFs logic [2]. The user-plane functional modules are brought together and implemented at the edge (as shown in Figure 1). They communicate with control-plane functional modules, which are implemented at the backend, through newly introduced network interfaces by 3GPP [2]. In our preliminary study, we find that SDN like distributed implementation of these decomposed functional modules cause deadlock and introduce race-conditions. We show these two issues happening in LTE Tracking Area Update (TAU) procedure.

Deadlock condition arises when both control-plane and user-plane functional modules of same NF lock their respective local resources and want to acquire a lock on each others locked local resource to complete their task. To take an example, when TAU procedure starts, user-plane traffic path and charging rules are modified at Charging Function's control-plane module. These modifications are required to be reflected at Charging Function's user-plane module. Assume, Charging Function's user-plane module has already processing changes required by relocation of some other NF, and wants to update its changes to Charging Function's control-plane module as well. As a result both Charging Function's user-plane and control-plane modules wait on each other to update their changes.

Race condition occurs, when control-plane signaling message arriving from two different NFs have timing constraint. The outcome depends in the arriving order of signaling message from two different NFs. For example, during TAU procedure, it is determined that user has moved to a different location and its gateway NF needs to be relocated. As a result, mobility management NF sends *relocation request* message to that gateway's control-plane module. However, this *relocation request* message has a race condition with another *relocation request* message coming from gateway's user-plane module that has reached its serving capacity. The order of these signaling messages result into two different behavior. The latest *relocation request* message will override changes made by the former one, and results into anomalous behavior.

In this paper, we argue against SDN like approach that splits LTE NFs into their control-plane and user-plane operations. Instead, we are in favor of retaining legacy LTE monolithic design for MEC implementation. Our idea is to process LTE control-plane procedures (such as *registration/de-registration*, *mobility*, and *location update* etc.) at the core; while moving the execution of LTE services (such as voice over LTE (VoLTE), multimedia, streaming, web, and gaming etc.) to the edge. It means user-plane traffic bypasses LTE core and directly gets forwarded to the Internet. This ultimately reduces the end-to-end user-plane latencies. We propose of creating a number of microservices where each microservice is tailored to handle a specific LTE service. Such as VoLTE-microservice and web-microservice handle VoLTE and web data traffic, respectively. To efficiently coordinate between LTE core and microservice, we propose our design $FERRET$. It is inspired from the concept of Circuit Switch Fallback (CSFB) in which LTE falls-back to legacy radio network (such as 3G/2G) for circuit switch voice call. Similarly, $FERRET$ falls back to specific microservice to facilitate particular user data service. When the subscriber requests a particular LTE service, the service is first established at the core. $FERRET$ records complete signaling messages (such as bearer setup, filtering rule, etc.) execution information during service establishment phase as first-in first-out (FIFO) order queue. It then re-establishes LTE service at edge microservice by replaying the recorded messages. The edge microservice reconfigures its interfaces with LTE base station and device and starts forwarding uplink and downlink data traffic. We prototype $FERRET$ over OpenEPC and consider VoLTE application. $FERRET$ reduces voice traffic latency upto 10X by introducing one-time delay of 0.5 second.

2 Background

LTE network consists of three main components, which are LTE device, LTE base-station, and Evolved Packet Core (EPC). LTE base-station anchors as a radio interface between device and EPC. EPC communicates with packet data networks in the outside world such as the Internet and facilitates user communication. LTE EPC comprises over a number of LTE NFs, that include Mobility Management Entity (MME), Serving Gateway (SGW),

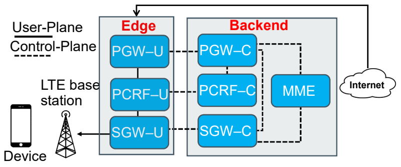


Figure 1: LTE MEC overview: User-plane functionalities (i.e. PGW-U, PCRF-U and SGW-U functional modules) of LTE core network are implemented at the edge (closer to the user); whereas, control-plane (i.e. PGW-C, PCRF-C and SGW-C functional modules) of LTE core network still resides at the back-end.

Packet Data Network Gateway (PGW), Policy and Charging Rules Function (PCRF), and few others. These NFs handle control-plane and user-plane traffic through separate network interfaces. Up till now, LTE NFs are implemented over carrier grade boxes which provide strong coupling between software and hardware as well as strong association between different NFs. Such contemporary LTE implementation brings user-plane latencies in which data packets need to traverse backend of LTE infrastructure before they reach to device. Recently, LTE NFV concept emerges that breaks software dependency on vendor specific platforms and paves the way to reduce user-plane latency at LTE network through MEC. In MEC, LTE NFs are first split into control-plane and user-plane functional modules and then user-plane modules are implemented closer to device, i.e. at mobile edge, as shown in Figure 1. Figure 1 shows PDN, SGW and PCRF NFs are split between their control and user planes functional modules. User-plane modules are moved at mobile edge, whereas control-plane operational modules still reside at the back-end.

Tracking Area Update Procedure The network keeps track of device location by mapping it to a particular location area (known as tracking area in LTE). Once the user moves from one tracking area to the other, it must inform MME for its new tracking area. Figure 2 explains TAU procedure which is initiated by a device where it sends *tracking area update request* message to LTE base station (step 1). LTE base station then forwards this requests to MME (step 2). Before processing TAU request message, MME authenticates the device (step 3). Once the device is authenticated, MME informs SGW about user location change and asks SGW to modify its bearer with device (step 4). SGW modifies device bearer and forwards the *modify bearer request* to PGW (step 5). PGW modifies device session at PCRF (step 6) and sends *modify bearer response* back to SGW (step 7). SGW informs successful bearer modifications to MME by sending *modify bearer response* message (step 8). Once bearers are modified at SGW and PGW, MME commits device new location at LTE network's database (known as Home Service Subscriber (HSS)) (step 9). Thereafter, MME confirms successful execution of tracking area update procedure at network by sending *tracking area update accept* message to device (step 10). The device then acknowledges the message by sending *tracking area update complete* message.

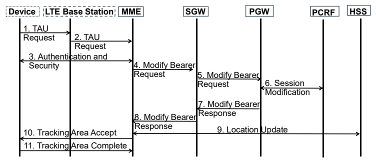


Figure 2: TAU procedure kicks-in when device moves from one tracking area (location) to an other. The LTE network updates device's new location in its database so that device remains reachable even during device idle periods.

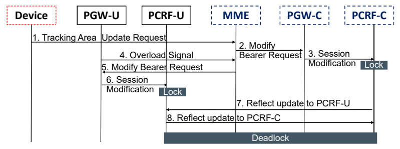


Figure 3: Deadlock happens when PCRF-C and PCRF-U have locked their respective processes and want to reflect their update by requesting lock on each other's locked processes.

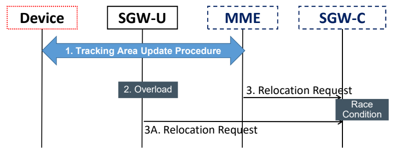


Figure 4: Race condition happens when SGW-C receives relocation request signaling message from MME and SGW-U at the same time.

3 Issues Arising from LTE Functional Split

We use TAU procedure to discuss that implementation complying LTE standard can cause deadlock and races when its NFs are split between control and user planes functional modules.

3.1 Deadlock

Figure 3 shows potential deadlock scenario in TAU procedure. TAU request being a control-plane procedure first arrives at MME (step 1). MME triggers TAU procedure by sending *modify bearer request* to PGW-C (via SGW-C) that forwards it to PCRF-C (steps 2 and 3). PCRF-C locks the process handling *session modification request* and makes changes into Traffic Handling Rules (THR) that control how the user-plane treats a certain piece of traffic. The THR contains packet flow description (IP filters, Application ID) as well as parameters that describe how that traffic matching the packet flow descriptor shall be treated. The PCRF-C also generates triggering conditions and thresholds based on which PCRF-U reports traffic usage information. Before such information is passed-on to PCRF-U, MME receives an overload signal from PGW-U(step 4). This overload signal informs MME that PGW-U is about to reach its capacity so load balancing is required. MME being a central entity, prepares new PGW-U resource and asks current PGW-U to switch to newly prepared PGW-U instance by sending *modify bearer request* message (step 5). PGW-U generates *session modification* message towards PCRF-U so that PCRF-U should account traffic coming from/to new PGW-U instance (step 6). On receiving *session modification* message, PCRF-U locks its process and session level reporting entries per PDN connection and applies THR rules. PCRF-U is required to update such changes to its distributed control-plane counterpart (PCRF-C) to complete *session modification* request. At this point, both PCRF-C and PCRF-U are required to reflect the changes they made to each other (steps 7 and 8). To achieve this, they seek a lock on *session modification* process at distributed instance, while locally holding a lock on the same process. This creates a deadlock scenario in which both PCRF-U and PCRF-C keep waiting on each other, resulting not only TAU procedure failure but also making any user traffic unaccountable at PCRF-U.

3.2 Race Condition

Figure 4 shows potential race-condition scenario. The TAU procedure is initiated by device when it moves to a new location which is served by a different LTE base-station and SGW. During TAU procedure execution, MME determines that the user has moved to an area served by a different SGW and the old SGW cannot continue to serve the user. MME decides to relocate the SGW by sending *relocation request message* to SGW-C (step 3). Note that, during TAU procedure, the MME may also decide to relocate the SGW if a new SGW is expected to serve the UE longer and/or with a more optimal UE to PGW path, or if a new SGW can be co-located with the PGW. Meanwhile (while TAU procedure is on-going), SGW-U finds itself overloaded and requires new instance of SGW-U to be initiated. SGW-U does so by sending *relocation request message* to SGW-C (step 3A) and SGW-C prepares new SGW-U instance. However *relocation request message* sent by MME and SGW-U have race-condition in which the changes requested by earlier NF (MME or SGW-U) are lost and causes anomalous behavior.

4 Our Position

Caution over LTE functional split We argue that LTE architecture which is monolithic by design should not be split into its control-plane and data-plane operations. LTE NFs are logically separated where they facilitate each other in executing certain tasks. One such NF is PCRF that takes subscribers charging rules and policy information from MME and HSS NFs, and applies these rules at uplink and downlink data traffic by communicating with SGW and PGW NFs. Other NFs, although, can be split into their control and data planes functional modules, they may cause deadlocks and races (as discussed in Section 3). Therefore, we call for caution over LTE functional split and argue that LTE monolithic design should be steered to achieve desired goals.

Fall-back as microservice To achieve our goal of reducing LTE data-plane latencies, we do not call for radical changes into LTE architectures. Rather, our position is let all types of LTE control-plane procedures (such as *registration, deregistration*, *mobility*, and *location update* etc.) be executed at the core, and only LTE services (such as voice, data, multimedia, and gaming etc.) execution logic is moved at the edge that reduces data-plane latencies. To achieve this, a number of dedicated microservices should be created to execute individual LTE services. For example, VoLTE-microservice and web-microservice should only handle voice calls and web services access, respectively.

5 Ferret Design

Design overview We propose $FERRET$ that provides fall-back to particular microservice instance in case of LTE service access. First, it captures all signaling messages in FIFO order as they execute for establishing LTE service at the core. Once the LTE service connection has been established at EPC, $FERRET$ replays these messages at dedicated microservice for execution. Figure 5 shows that $FERRET$ captures messages (M_1 and M_2) as they execute between different NFs (NF_1 and NF_2). Finally, it replays these messages at the edge. The microservice NFs receive messages in-order and execute them accordingly. They also reconfigure their interfaces with LTE base station and device to assume the role of legacy EPC. Once LTE service connection falls back to microservice, the data-plane traffic can start. Now the data-plane packets are directly forwarded to the Internet from edge mircorservice without going to legacy EPC.
We next explain $FERRET$ design in detail.

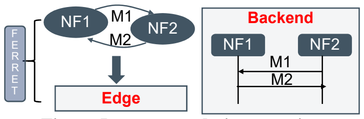
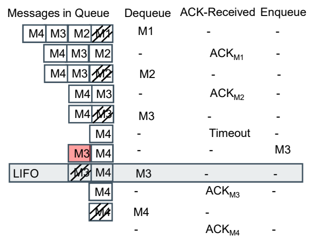


Figure 5: *FERRET* design overview

Capturing signaling messages LTE NFs always exchange a number of signaling messages for every LTE service request. For example, for VoLTE service request from device, MME authenticates and authorizes the device, SGW and PGW establish dedicated voice bearer, PCRF enables charging rule, and IP Multimedia Subsystem (IMS) finally establishes the call. The call establishment procedure also involves signaling messages exchange between a number of IMS NFs (such as P-CSCF, S-CSCF and Media Gateway). $FERRET$ deploys a $FERRET$ agent at every NF that records complete message execution information. Such information include, the message name, message type, input and output parameters, number of state transitions, and name of two NFs among which the message is exchanged. These messages are placed in the queue (to be later replayed at the edge NFs) in exactly same order as they are executed at first place. $FERRET$ can achieve this by exploiting sequential execution of LTE messages, and POSIX message queues implementation. LTE standard requires sequential execution of LTE signaling messages, where the following message shall not execute until former message is successfully executed. To ensure this, LTE standard explicitly enables sequence number field in different messages (such as NAS sequence number indicating the sequential number of the NAS protocol message) [3]. As a result, $FERRET$ can safely assume that all LTE messages arrive in-order. To ensure different $FERRET$ agents do not interleave, it adopts POSIX message queues implementation approach. POSIX message queues allow for an efficient, priority-driven mechanism with multiple readers and writers. Whenever a $FERRET$ agent sends a message to a queue, a priority (first-in first-out) is specified for that message. The queue remains sorted such that the oldest message will always be at the front.

Replaying signaling messages at edge Once $FERRET$ constructs signaling messages queue, it next forwards these messages to particular LTE microservice located at the edge. The edge microservice then executes these signaling messages and re-establishes the service. We explain this procedure through VoLTE call example. Once VoLTE call has been established at the core, $FERRET$ requests IMS to place call on hold (by sending *call_hold* signal to IMS) while it is replaying recorded control-plane messages at edge. $FERRET$ knows all signaling messages are queued as first-in first-our order, where it dequeues the front most message and sends it to particular edge NF. The edge NF executes the message, compare its produced output parameters with that of it has received from $FERRET$ and forwards it to next NF (if it needs to). For example, during VoLTE call setup the first two messages are create session request (M_1) from SGW $\rightarrow$ PGW, and create session response (M_2) from PGW $\rightarrow$ SGW. Both these messages are responsible for establishing VoLTE dedicated bearer. $FERRET$ sends M_1 and M_2 to SGW_{edge} and PGW_{edge}, respectively. The SGW_{edge} forwards M_1 to PGW_{edge} that executes the message. PGW_{edge} then ensures that it has generated same output parameters as provided by $FERRET$ in M_2. These include, *voice PDN IP address*, *transport layer address*, *protocol ID*, and *quality of service class*. It should be noted that the exact parameter values (such as value of IP address parameter) may differ from the ones received in M_2.

Reconfiguring interfaces The LTE service session which has already been setup between EPC, LTE base station, and device needs to be updated with respect to edge NFs. This is achieved by reconfiguring EPC interfaces with base station and device. The edge NFs (MME_{edge} and SGW_{edge}) reconfigure their interfaces with base station as soon as they receive first message from $FERRET$. These NFs ask the base station to update its interfaces towards MME and SGW by sending *S1 Application Protocol (S1-AP) request* message. This message indicates that the subscriber uplink and downlink packets should be sent to new MME and SGW addresses. Moreover, edge microservice asks device to update its connection from EPC to micrservice. This is taken



(a) Transition between FIFO and LIFO

```
FIFO_Dequeu ()
    if (message is never dequed)
        message = Deque_message_FIFO ( Queue )
        sendto_edgeNF (message)
        wait_for_ACK () /*ACK or timeout*/
        if (message is ACKed)
            goto ( FIFO_Deque() )
        else
            Enqueu (message)
            goto ( LIFO_Deque() )

LIFO_Dequeu ()
    message = Deque_message_LIFO ( Queue )
    sendto_edgeNF (message)
    wait_for_ACK () /*ACK or timeout*/
    if (message is ACKed)
        goto ( FIFO_Deque() )
    else
        Enqueu (message)
        goto ( LIFO_Deque() )
```

(b) FIFO and LIFO transition algo

Figure 6: All queued messages are sent to respective edge NFs as first-in first-out. Those messages which are not ACKed are enqueued again. $FERRET$ dequeues unacked message(s) as last-in first-out and re-sends them to respective edge NF. The Figure has been discussed in Discussion section (after conclusion)

care while edge NFs are executing the messages as received from $FERRET$. We make changes in the protocol implementation of the edge NFs where instead of sending *create-dedicated-bearer-request* message (as requested by $FERRET$), we send *modify-dedicated-bearer-request* message to device (via MME). This message asks the device to modify its dedicated bearer's IP address and access point network address. Thereafter, the device is ready to carry out its data-plane communication through edge mircoservice.

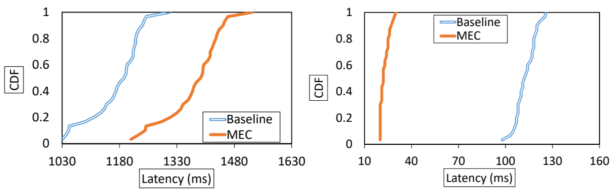
6 Prototyping and Evaluation

Implementation Our system implementation consists of OpenEPC [4] LTE implementation and virtualization of LTE EPC NFs. 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. We virtualize EPC NFs over vMware vSphere, which is a server virtualization platform with consistent management. We first decompose 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.

IMS as microservice We consider IP Multimedia Subsystem (IMS) as a microservice and supports VoLTE call. For IMS implementation, we use OpenIMS that provides basic implementation of IMS NFs (such as S-CSCF and P-CSCF). We deploy these IMS NFs (S-CSCF and P-CSCF) and LTE NFs (SGW, PGW and PCRF) as IMS microservice.

Preliminary results

We provide VoLTE service control-pane and data-plane latencies CDF. Figure 7a provides call setup delay (control-plane) results for proposed Mobile Edge Compute (MEC) design and baseline (without MEC). We can see that our proposed MEC design takes on average 500ms more compared to baseline design. The reason MEC incurs higher latency is that $FERRET$ is required to replay all signaling messages (that were executed at the core) again at the edge. Because baseline does not require



(a) Control plane latencies (b) Data plane latencies

Figure 7: (a) VoLTE call setup, and (b) voice packets delay: The baseline (legacy without edge mircorservice) incurs less voice call setup signaling (control-plane) latencies. In mobile edge compute (MEC) design, *FERRET* needs to replay recorded packets and reconfigure interfaces and takes roughly 500 miliseconds more. The real gain in MEC comes in reduction of VoLTE speech packets (data-plane) delay, where they are forwarded directly from the edge – reducing latency upto 10X compared to baseline.

any extra overhead, therefore it incurs lower latency. The real benefit of our MEC design comes in terms of user-plane traffic. Figure 7b shows that MEC incurs lower latency compared to baseline design for user plane traffic. In case of MEC, user-plane traffic is directly forwarded to the Internet without going to EPC; this reduces latency an order of 10 times.

7 Related Work

Closest to our work are Rethinking LTE NFV [5], ACACIA [6], DPCM [7], LTE-Xtend [8], Contain-ed [9], OpenNF [10], and Scaling LTE control plane [11]. Rethinking LTE NFV work [5] argues in favor of logic based decomposition of LTE NFs instead of their instance based decomposition. The logical modules are implemented as Fat-Proxy and reduces LTE critical procedures, such as handover, service access, and paging. Our work does not consider modifying LTE procedures execution. ACACIA [6] proposes a framework that enables interactive applications on edge clouds of mobile networks. It leverages SDN/NFV principles with LTE/EPC QoS mechanisms to steer interactive application traffic to closely located mobile edge clouds. It also splits LTE NFs (as shown in Figure 5 [6]) to develop MEC. Contrary to ACACIA, we argue against SDN based approach for mobile edge computing and do not split NFs. DPCM [7] proposes low latency LTE data access approach for service request, handover and roaming scenarios. However, we do not try to reduce latencies for LTE control-plane signaling, rather focus on LTE services. Contain-ed [9] splits LTE user and control plane and implements user plane functions at the edge. However, it fails to point out any issue that may emerge from SDN style functional split.

8 Conclusion

This paper calls for caution over LTE functional separation in next generation LTE network. It argues that SDN approach for splitting monolthic LTE design is not right. When a NF is split into its control and user planes operations, their distributed implementation may cause deadlocks and race conditions. Instead, this paper advocates for microservice-fall-back concept in which only LTE services are executed at mobile edge, while keeping LTE control-plane procedures at the core.

Discussion

This paper stimulates the discussion in mainly three aspects: (1) how $FERRET$ handles failures? (2) how the device switches back from the microservice to its associated EPC?, and (3) what are existing limitations of $FERRET$ and how we will extend this work?

(1) Recovering failures through FIFO and LIFO transitions Unlike signaling messages exchange in legacy EPC, $FERRET$ wants each of its messages to be ACKed by edge microservice. In case of failure, we argue that $FERRET$ can retransmit the message by simply switching the message dequeue order from first-in first-out (FIFO) to last-in first-out (LIFO). It implements FIFO and LIFO transition algorithm (Figure 6b) to ensure in-order delivery of messages even during failures. At the start, it dequeues the message as FIFO and sends this to respective edge NF for execution. The edge NF should send an ACK on receiving the message. In case $FERRET$ does not receive the ACK, it puts back the message into the queue. It then dequeues the message as LIFO and retries. Figure 6a explains the process. It has been show that $FERRET$ has received ACKs for M_1 and M_2 messages. It dequeues next message (M_3) as FIFO and sends it to edge NF. M_3 was not ACKed by edge NF and *ACK_receive* timer has timed-out. On timeout event, $FERRET$ first enqueues M_3 and changes its dequeuing order from FIFO to LIFO. This means on next round of dequeuing, the failed message (M_3) is dequeued and resent to respective edge NF. This time $FERRET$ has received an ACK for M_3 and switches back to FIFO for dequeuing next message (M_4). In short, by transitioning between FIFO and LIFO, $FERRET$ ensures that messages are sent in-order even during transmission failures.

(2) Releasing connection Once LTE service concludes at the subscriber (such as voice call hangs-up), the device releases the radio connection. The base station sends this release signal to in-service microservice that finally falls the subscriber connection back to EPC core. Thereafter, EPC re-configures its interfaces with base station and registers the device status as *Idle*.

(3) Limitations One key limitation of $FERRET$ is that it can only handle one application at a time. This is because, we are using standardized base station interfaces that can only extend to one EPC instance (i.e. core or microservice). Once the base station is connected to in-service microservice (say VoLTE microservice), it cannot switch to any other microservice (say web microservice). The other limitation is that our microservice cannot function during handover procedure as we have not implemented location update procedure at microservice architecture for simplicity.
In future work, we aim to address both these issues by placing network address translator like agent at LTE base station that allows it to associate with multiple EPC instances (i.e. legacy EPC and a number of microservices).

References

[1] Ericsson's Integrated Site concept. https://www.ericsson.com/ericsson/corpinfo/publications/review/2005_01/files/2005014.pdf.

[2] 3GPP. TR 23.714: Study on control and user plane separation of EPC nodes, September, 2016.

[3] 3GPP. TS23.401: GPRS Enhancements for E-UTRAN Access, 2011.

[4] Open EPC - open source LTE implementation. http://www.openepc.net/.

[5] M. T. Raza and et al. Rethinking lte network function virtualization. In *IEEE ICNP*, 2017.

[6] J. Cho and et al. Acacia: Context-aware edge computing for continuous interactive applications over mobile networks. In *ACM CoNeXT*, 2016.

[7] Y. Li and et al. A control-plane perspective on reducing data access latency in lte networks. In *ACM MobiCom*, 2017.

[8] V. Nagendra and et al. LTE-Xtend: scalable support of M2M devices in cellular packet core. In *ACM Workshop on All Things Cellular*, 2016.

[9] A. Sheoran and et al. Contain-ed: An NFV Micro-Service System for Containing e2e Latency. In *ACM workshop HotConNet*, 2017.

[10] A. Gember-Jacobson and et al. Opennf: Enabling innovation in network function control. In *ACM SIGCOMM*, 2014.

[11] A. Banerjee and et al. Scaling the LTE control-plane for future mobile access. In *ACM CoNEXT*, 2015.