Fast EPC: A Low Latency Cellular Control Plane

Mukhtiar Ahmad, Wasiq Noor Ahmad Qasmi, Syed Usman Jafri, Ridah Naseem, Muhammad Ali Nawazish, Muhammed Azam Ikram, Zartash Uzmi, Zafar Ayyub Qazi
LUMS, Pakistan

ABSTRACT

Timely completion of control plane operations is critical to providing fast data access in cellular networks. In this work, we design a new edge-based cellular control plane, Fast EPC, which reduces control plane latency through fast serialization of control messages and rapid failure recovery.

CCS CONCEPTS

• **Networks** → *Middle boxes / network appliances*; *Wireless access points, base stations and infrastructure*; *Control path algorithms.*

KEYWORDS

Cellular Core, Control Plane, MME

ACM Reference Format:
Mukhtiar Ahmad, Wasiq Noor Ahmad Qasmi, Syed Usman Jafri, Ridah Naseem, Muhammad Ali Nawazish, Muhammed Azam Ikram, Zartash Uzmi, Zafar Ayyub Qazi. 2019. Fast EPC: A Low Latency Cellular Control Plane. In *SIGCOMM '19: ACM SIGCOMM 2019 Conference (SIGCOMM Posters and Demos '19), August 19–23, 2019, Beijing, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3342280.3342324

1 INTRODUCTION

5G networks promise to provide ultra-low latency to support applications such as virtual reality, remote surgery, self-driving cars, and to improve the responsiveness of web browsing and other applications. However, the existing 4G/LTE network deployments suffer from path inflation as all user traffic has to be routed to one of the few data centers hosting cellular packet core functions. This can result in web traffic experiencing delays of hundreds of milliseconds [7]. In order to meet latency demands, 5G cellular networks are expected to move the core functionality closer to the edge.

The cellular packet core[1] is an important component of cellular networks that connects the IP backbone with the base stations and implements cellular-specific processing on user's control and data traffic. A key component of cellular core is the control plane, which provides services including, coordinating device mobility, authentication and managing sessions. With 5G, control traffic is expected to increase rapidly because of (i) a shift to small cell sizes, which will likely cause more mobility handoffs and (ii) the proliferation of IoT devices with high control to data traffic ratio. The time taken by the cellular control plane to process control traffic can have a direct impact on the delay experienced by user applications [4]. In addition, failures of control plane functions can exacerbate these delays [2]. As a result, timely completion of control plane procedures is vital to provide low latency and reliable data access to user applications.

In this work, we design a new edge-based cellular control plane, Fast EPC. We redesign the key cellular control plane element, Mobility Management Entity (MME), and load balancer for control traffic. Fast EPC significantly reduces control plane latency by (i) speeding up the processing of control updates between base stations and MME by using a FlatBuffers-based serialization technique and (ii) providing fast failure recovery through a synchronization protocol which uses procedure-level check pointing at MME (reducing synchronization overhead) and message logging at the load balancer (while ensuring small messages log). We implement Fast EPC and evaluate its performance on real control traffic traces. Our results show an improvement in control procedure completion times by upto 6× without failures, and upto 8× under MME failure.

2 BACKGROUND AND MOTIVATION

A typical end-to-end communication path for control messages in cellular networks is as follows: a User Equipment (UE) generates a control request/update, sends it to the base station which then forwards it to the packet core. Inside the packet core, the traffic may go through a load balancer, which then routes it to the appropriate MME. The MME decodes the request/update and sends a response back to base station/UE. Events like `attach` to the network and `handover` to a different base station, lead to a sequence of request/response messages between base station/UE and MME. In 4G/LTE, the S1AP [1] protocol is used to exchange these control messages between the base station and the MME. These messages are serialized using ASN.1 [1]. Our analysis of these messages based on the 3GPP standard and real control traffic show that (i) a single message can consist of multiple information elements (objects), (ii) data in these messages is organized in a hierarchical fashion, with potentially multiple nested elements. Our benchmarking of these control messages shows that processing them can take significant time (see Figure 3). As the core functions move towards the edge to reduce the propagation delays, the processing delays can become the bottleneck.

3 FAST EPC DESIGN

Figure 1 shows the overall design of Fast EPC. Below we describe the Fast EPC's key mechanisms:

Control Message Serialization Engine: For speeding up the processing of control messages, in Fast EPC these messages are encoded in a flat binary buffer by MMEs and base stations, using the

[1] Cellular packet core for 4G/LTE is called Evolved Packet Core (EPC).

SIGCOMM Posters and Demos '19, August 19–23, 2019, Beijing, China

ACM ISBN 978-1-4503-6886-5/19/08…$15.00
https://doi.org/10.1145/3342280.3342324

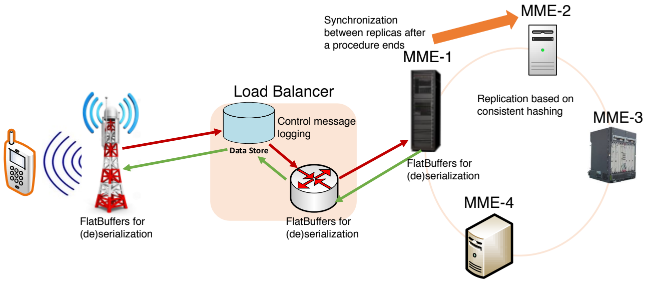
Synchronization between replicas after a procedure ends
MME-2
MME-1
Load Balancer
Control message logging
Data Store
Replication based on consistent hashing
MME-3
FlatBuffers for (de)serialization
FlatBuffers for (de)serialization
FlatBuffers for (de)serialization
MME-4

Figure 1: Fast EPC Design.

FlatBuffers serialization scheme [3]. FlatBuffers represent hierarchical data in a flat binary buffer in such a way that it can still be accessed directly without parsing/unpacking. The only memory needed to access the control data is that of the buffer and it requires 0 additional allocations, saving processing time. When MME and base stations receive a control message, the message is decoded using the FlatBuffers. Load balancers selectively access a small subset of fields in the message (e.g., user ids such as GUTI) to decide the appropriate MME.

Failure Recovery Mechanism: Existing EPCs require the UE to re-attach to the network in case of an MME failure. One key reason is that even if MME replicas exist, they may be out-of-sync (not having the updated user state) because they are not always kept consistent [2]. To avoid re-attaching to the network, Fast EPC requires MME replicas to sync UE state after the completion of each control plane procedure. While the procedure is in-progress the load balancer logs the control messages. The load balancer stores these messages in RAM and prunes the message log as soon as the procedure is completed, ensuring the log does not grow unbounded. Our implementation consists of (i) an MME based on PEPC [6], (ii) MME replication scheme that uses consistent hashing, (iii) MME replica synchronization protocol, and (iv) load balancer for control traffic with support for message logging.

4 EVALUATION

Testbed: Our test setup consists of two servers, with one server running custom DPDK-based traffic generator, while the other server runs a load balancer and MME instances. For testing control traffic, we implement the handling of request and response messages between the UE/eNodeB and EPC for `attach` and `handover` procedures based on S1AP protocol. The experiments are run using real signaling traces from a commercial traffic generator and RAN emulator [5]. Using these traces, we create a schema of all the messages in both ASN.1 and FlatBuffers.

Results: In our evaluation, we compare Fast EPC with Existing EPC, which uses ASN.1 and requires UEs to reattach on MME failure. Below we discuss key evaluation results.

Procedure completion times without failures: Figure 2 shows the median and 99 percentile of completion times for an `attach` procedure with varying number of active UEs, with Fast EPC and Existing EPC. We observe an improvement of up to 6× in median completion times when using Fast EPC. This is primarily because of the fast serialization engine for control messages. Figure 3 shows the speedup in encoding/decoding with FlatBuffers as well as the corresponding increase in message size. We observe with FlatBuffers up to 180× decrease in decoding times and up to 10× decrease in message encoding times. This speedup does come at a cost; the encoded message in FlatBuffers can be between 1-2× larger than the encoded message in ASN.1. However, as we observe in Figure 2 this cost is offset by the decrease in encoding/decoding times.

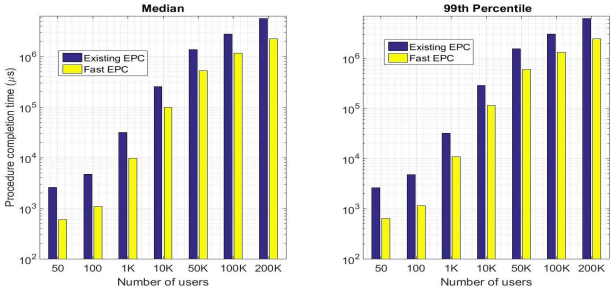

Figure 2: `attach` procedure completion times with varying number of active users.

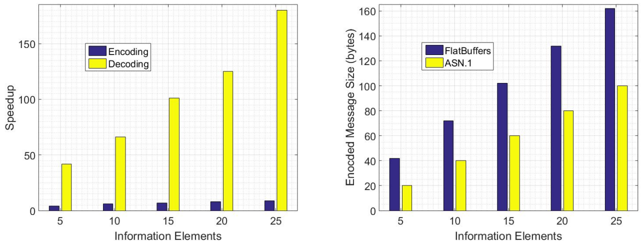

Figure 3: Speedup through FlatBuffers and its overhead.

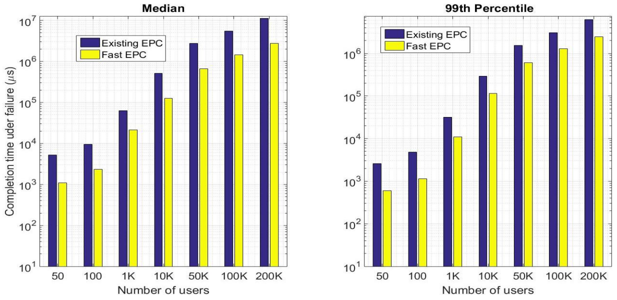

Figure 4: `attach` procedure completion times under failure.

Procedure completion times under failure: Figure 4 shows the median and 99 percentile of procedure completion times under an MME failure for an `attach` procedure. We observe an improvement of upto 8× in median completion times when using Fast EPC. In addition to faster serialization, this increase is attributed to faster failure recovery in Fast EPC. Instead of re-attaching the UE on an MME failure, load balancer sends logged messages to the replica MME, which then replays them to reconstruct the state updates from the last checkpoint. This saves multiple RTTs for UEs.

5 CONCLUSION AND FUTURE WORK

In this work, we (i) present the preliminary design of a cellular control plane, Fast EPC, that speeds up the processing of control messages and improves failure recovery times, and (ii) implement and evaluate its performance on a testbed using real control traffic traces. Our ongoing and future work includes implementation of other control procedures and evaluation of Fast EPC under different failure scenarios and traffic workloads.

REFERENCES

[1] 3GPP Ref #: 36.413. 2016. S1AP. https://portal.3gpp.org/.

[2] Arijit Banerjee, Rajesh Mahindra, Karthik Sundaresan, Sneha Kasera, Kobus Van der Merwe, and Sampath Rangarajan. 2015. Scaling the LTE Control-plane for Future Mobile Access. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT ’15)*. ACM, New York, NY, USA, Article 19, 13 pages. https://doi.org/10.1145/2716281.2836104

[3] Google/GitHub. 2019. *FlatBuffers*. https://google.github.io/flatbuffers/

[4] Yuanjie Li, Zengwen Yuan, and Chunyi Peng. 2017. A Control-Plane Perspective on Reducing Data Access Latency in LTE Networks. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom ’17)*. ACM, New York, NY, USA, 56–69. https://doi.org/10.1145/3117811.3117838

[5] ng4T. 2016. *Traces*. https://www.ng4t.com/wireshark.html

[6] Zafar Ayyub Qazi, Melvin Walls, Aurojit Panda, Vyas Sekar, Sylvia Ratnasamy, and Scott Shenker. 2017. A High Performance Packet Core for Next Generation Cellular Networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM ’17)*. ACM, New York, NY, USA, 348–361. https://doi.org/10.1145/3098822.3098848

[7] Kyriakos Zarifis, Tobias Flach, Srikanth Nori, David Choffnes, Ramesh Govindan, Ethan Katz-Bassett, Z. Morley Mao, and Matt Welsh. 2014. Diagnosing Path Inflation of Mobile Client Traffic. In *Proceedings of the 15th International Conference on Passive and Active Measurement - Volume 8362 (PAM 2014)*. Springer-Verlag, Berlin, Heidelberg, 23–33. https://doi.org/10.1007/978-3-319-04918-2_3